Using Task-oriented Requirements Engineering in Different Domains – Experience of Application in Research and Industry

Sebastian Adam, Joerg Doerr, Michael Eisenbarth, Anne Gross
Fraunhofer IESE, Fraunhofer Platz 1, 67663 Kaiserslautern

Abstract
The early orientation on tasks of the application domain to be supported by software systems has been proposed as a fruitful means during requirements engineering for achieving more appropriate and usable systems as well as to focus the requirements engineering process. Besides goal-oriented approaches, task orientation has therefore been recognized as a complementary and promising concept to assure a higher completeness and correctness of requirements specifications and, in particular, a better integration of requirements and usability engineering issues. In this paper, we present experience made with the task-oriented requirements engineering framework “TORE” in four different case studies. The case studies for this paper were selected based on their contextual specifics like service orientation or the ambient domain that might impose certain challenges on task-oriented RE. As a lesson learned, we experienced TORE to be highly beneficial even in systems that do not seem to be “traditional interactive systems” at a first glance. However, we also identified challenges that pose necessary adaptations to be made.

1. Introduction
The traditional approach of just eliciting software requirements has been experienced as not sufficient to provide adequate stakeholder support and a fulfillment of certain goals related with a software system [1]. Especially the achievement of complete and correct requirements specifications has been experienced to be hard if the purpose and the application context are not explicitly considered.

In the last decade, there has therefore been a shift towards a stronger orientation on the tasks and goals to be supported. Several state-of-the-art requirements engineering approaches such as goal-modeling [2], scenarios [3], or Use Cases [4] therefore reflect this orientation on more problem-oriented than solution-oriented requirements at the beginning of each development project. Thus, the application environment and domain requirements, i.e., the business tasks and goals to be supported, are to be analyzed first in order to make the actual product requirements, and thus, the resulting systems more usable and appropriate for their intended purpose and audience, at least in the information system domain.

At the beginning of this decade, we introduced task orientation at our institute as the fundamental principle for our requirements engineering work, and we developed a conceptual framework for task-oriented requirements engineering (called TORE) [6] that now proposes a set of 18 decisions that have to be made (explicitly or implicitly) every time the requirements for an information system are developed. The framework is based on the observation that systems will probably not support the users in an adequate manner, if the more application-oriented decisions on a high level of abstraction have not all been made before lower level requirement decision are addressed. Thus, TORE aims at making the conceptual dependencies between different decisions explicit, allowing a better guidance during the entire requirements process.

In more than ten projects with partners from research, industry, and the public domain, we applied the TORE framework as the conceptual basis for our requirements engineering activities. We have found out that an orientation on the decisions to be made during requirements engineering rather than on certain artifacts, techniques or processes enables, basically, a highly flexible application of the task-oriented paradigm in different domains and project settings.

However, TORE is not intended to reflect domain-specific needs. Besides the limitation of being intended for (traditional) information systems, TORE does not incorporate the specificities of other domains. Hence, the advent of, e.g., service-based systems, ambient intelligence, or hybrid applications, necessitates the qualitative evaluation whether these specific contexts impose new challenges to the task-oriented approach.

In the remainder of this paper, we therefore describe our experiences with TORE made in four case studies, and we highlight the benefits but also unsolved challenges when applying the approach outside the (traditional) information system domain.
2. TORE at a Glance

TORE is a decision framework that encapsulates 18 decisions on four different levels of abstraction that have typically to be made during requirements engineering for interactive (information) systems (see Figure 1). The benefit of thinking in these decisions is that it can serve as a conceptual model independent of concretely used processes or notations allowing a high applicability in many different contexts.

Figure 1. Decision points in the TORE framework

At the Goal & Task Level, the first decision point is “Supported Stakeholders”. Deciding which stakeholders should be supported by a system to be developed is usually one of the initial decisions to be made. Typical notations used to make this decision explicit are stakeholder maps as used in [10], stereotypical user descriptions like personas [14] or simple role descriptions. The second decision point is the capturing which Stakeholder’s Goals exist and shall be supported by the system. With TORE we support goals of organizations (business goals) as well as goals of users (individual goals). Typical notations used for documenting goals are notations used in methods like KAOS [12], i* [13], or simple AND/OR goal refinement trees. Typically, the functional goals are refined to Stakeholder’s Tasks. In simple information systems, Stakeholder’s Tasks include the tasks of the users, while in complex business information systems this decision point is rather the hierarchy of business processes. In the Domain Level, each Stakeholder’s Task in the list is refined into its As-Is Activities, i.e., the description of how tasks and processes are performed today without the system to be developed. In contrast to that, the To-Be Activities describe the tasks or business processes as they should be carried out when the system to be developed is in place. The typical notation used to describe the As-Is and To-Be Activities is process modeling notations like EPCs [8] or UML Activity diagrams [7]. With System Responsibilities decisions, one determines which of the To-Be Activities are performed automatically, which are performed by humans and which are system-supported (i.e., humans and the system interact). The Domain Data decisions then determine which data is visible on the Domain Level.

Typical notations are ER-Diagrams or UML class and object diagrams. In the Interactions decision point at the Interaction Level, the concrete usage of a system by a human is determined for all system-supported To-Be Activities. A typical notation used for this decision point is Use Cases or other scenario techniques. For all System Functions that were identified during the To-Be Activities and Interactions, the System Functions decision point determines the details of the system functions (visible behavior, input, output, etc.). The Interaction Data decision point determines the data used on the interaction level. This is typically a refinement of the Domain Data, using similar notations. The UI-Structure is a first logical grouping of functions and data. It is neither a detailed UI, nor is the modality determined on this level. Typical notations used to document these decisions are workspaces as in [15]. The aforementioned decision points (sometimes also the ones on the subsequent System Level) are the typical decision points that we use in order to determine the decision in our requirements engineering activities. More detailed information of TORE in general can be found in [6].

3. Applying TORE in Research & Industry

We have applied TORE as a conceptual basis in more than 10 projects including those that have not been from the traditional interactive systems domain. To illustrate the wide applicability of TORE, we subsequently present four case studies from different domains, which have been selected based on their context specifics. Table 1 summarizes the specifics, benefits and challenges for the four case studies.

3.1. Social Network Application

Project Context. This industrial project exhibits mostly the characteristics TORE was originally designed for. Hence, we successfully applied TORE in the context of a social network application, featuring the basic characteristics of the initially addressed application domain. The goal of the project was to identify easy to use interaction activities for non-professional IT-users based on typical user tasks.

TORE Application. We started our TORE application by identifying the Supported Stakeholders and their Goals, related to the specific business goals of the customer. This step was followed by an in depth analysis of the concrete Stakeholder’s Tasks and the current As-Is Activities. The reason for an in depth analysis of the As-Is Activities was motivated by the current statements and problems related to the state of the practice regarding these tasks. The next step was an analysis of the existing platform and System Functions towards To-Be Activities that can support the envisioned Stakeholder’s Tasks in the future. Based on
already existing System Functions and ones to be implemented, it was easy for our customer to clearly separate among the System and User Responsibilities. In a final step, the concrete Use Cases were identified and specified in order to understand and model the envisioned Interactions between user and system.

Benefits: TORE offered a systematic way to derive and decompose the functional requirements. Usability issues were addressed from the beginning due to the task-orientation.

Challenges. A major challenge was to apply TORE in our customer’s agile development context. Our customer used the SCRUM [5] approach to manage the project development process. Therefore, we had to map the individual TORE decisions points to the SCRUM phases and artifacts. We experienced that the decision points on the Goal & Task Level and Domain Level as well as the System Functions and Interactions could be handled within the product backlog of the SCRUM approach, describing both functional and non-functional requirements, as well as technical team-generated requirements. Within the individual sprint backlogs, which define the set of work tasks for a sprint within SCRUM, we were able to address especially the System Level decision points, detailing the individual Use Cases.

3.2. Ambient Assisted Living

Project Context. Another context where we applied the TORE approach was in a research project within the field of Ambient Assisted Living, a quite new research field focusing on systems that enable elderly or disabled persons to live longer in their preferred environments instead of living in a nursing home, for instance. One of the special characteristics of ambient systems is that they rely on intelligent networked systems that receive their information via numerous, unobtrusive sensors that are able to analyze the situation (for example health status) of assisted persons and take appropriate measures, for example by activating emergency reactions as it was also the goal of the research project considered in this case study.

TORE Application. Starting with a stakeholder analysis where we identified and described relevant stakeholders to be supported by the system (Supported Stakeholders) we did a detailed goal and task analysis comprising the identification of Stakeholder’s Goals and to understand current Stakeholder’s Tasks. In several interview sessions with stakeholder representatives (in particular, dispatchers, emergency physicians, or in-patient caregivers) we derived hierarchically structured and prioritized goals they currently want to achieve when handling an emergency situation. Besides that, we also asked the stakeholders to indicate which of the goals are currently difficult to reach, what kind of problems they currently face, what kind of information needs they have as well as improvement suggestions they have regarding their current processes. We assigned all this information to the concrete activities included in their current workflows that we also elicited in the interviews (As-Is Activities) to finally come up with a process model.

Afterwards, we identified To-Be Activities and modeled to-be processes that have been validated with stakeholder representatives in further interview sessions. We then identified System Responsibilities and modeled the Domain Data. For those activities that require interactions between stakeholders and the system we specified the Interactions and Interaction Data in form of detailed use case descriptions that have again been validated with stakeholder representatives before systematically deriving and describing System Functions that have finally been used as input to the design of user interfaces.

Benefits. The systematic TORE approach worked very well in case of elicitation and specification of requirements related to that part of the system that relies on information and communication flows (in particular that part of the system that reacts on detected emergency situations by alarming dispatchers or caregivers). Getting a deep understanding of current processes was in fact a very important activity as one of the main objectives within this project was a close integration into existing emergency medical systems (EMS) whose processes are nowadays quite efficient and rely on lots of human decisions that we did not want to influence in a negative way. Nevertheless, the detailed analysis of goals and As-Is Activities supported us to identify activities in these efficient processes that could be improved by our system. The usage of results that we derived at the various decision points in validation interviews showed that the stakeholders who were not experts in software development were able to understand the concepts of the system in order to prioritize and validate its functionalities proposed in the to-be process models and use case descriptions. Furthermore, the set of Use Cases and system functions served as valuable input for the design of the user interface.

Challenges. We faced the challenge to systematically follow the TORE decision points for specifying requirements related to the more “intelligent” parts of the system, such as the detection of activities of daily living or emergency situations based on reasoning on information derived from sensors. Due to the complexity of situations that indicate deteriorations of the health status of an assisted person, it was quite difficult to derive a to-be process model in a systematic way. However, in several workshops with experts form the medical
3.3. Services for the Office Domain  

*Project Context.* Even if TORE has originally been intended for application engineering in single system development, we also applied TORE successfully in the context of a service-oriented “design for reuse”.

In this project, a company was interested to enter the service market by providing single, self-contained (web) services for the office domain. TORE was used as a systematic approach for the identification and specification of services providing innovative functionality for supporting daily office tasks.

TORE Application. As the office domain was quite fuzzy, the company first told us specific market segments in which it was interested to sell the services, e.g., human resource departments. We then involved representative stakeholders from different organizations within these segments and asked them for their responsibilities and tasks within their organizations. In the human resource case, for instance, we got a list of task such as creating job advertisements, or preparing contracts. Together with the stakeholders, we then analyzed how these tasks/workflows are today done, and we identified shortcomings, respectively unsatisfied goals that hamper a more efficient and satisfactory work in day-to-day business. Based on a classification of these shortcomings, we developed innovative solution ideas together with a development team from the company and proposed improved workflows to the stakeholders. After incorporating their feedback, we determined the workflow steps to be supported by software.

Based on a detailed description of these steps using Use Cases, we systematically transformed each system step within the Use Cases into descriptions of System Functions. These descriptions were then analyzed, negotiated, clustered and consolidated according to our service derivation method described in [9]. Hence, we finally achieved a set of 13 services, which the company was interested to provide in order to support daily tasks within the considered market segments.

**Benefits.** First, the TORE approach provided us a systematic way to decompose completely and traceable workflows of a certain segment into service functions. Second, the different levels of TORE helped us to find an appropriate level of abstraction on which service functions could be provided in order to be highly reusable on the one hand, but also immediately beneficial for the intended audience on the other hand. If we had identified service functions on a lower level of abstraction, we would probably not have provided a support that was considered useful by the business users. But, if we had identified service functions on a higher level of abstraction, this would not have enabled an efficient reuse, because the business process flows to be supported were largely individual and often not reusable in another context.

**Challenges.** TORE did not provide us a systematic support for transforming the As-Is Activities into improved To-Be Activities. As the company was explicitly interested in generating innovations, we therefore had to incorporate creativity techniques [11] into the entire project method in order to increase the degree of innovation significantly.

3.4. Transport Automation System  

*Project Context.* In another project, a customer from the foundry industry asked us for supporting him in writing a system definition for an innovative transport automation system (TAS) in his factory, which should be delivered by an external supplier.

TORE Application. After a kick-off meeting in which we were informed about the general objectives, we started analyzing the actual tasks to be supported, mainly, “transporting material between different buildings of the plant”. However, we also identified a list of administrative tasks, such as “manage transport wagons” which only appeared due to the introduction of the TAS itself and that therefore had to be supported too, of course. However, as the primary task could not be seen independently of the entire production of large iron castings, we had to model the complete to-be process on how the production should be done when using the TAS in the future. A critical point during this modeling activity was to determine which process steps should be supported by the TAS and which steps should be supported by another machine such as a crane. The same held true for the interactive steps, i.e., steps that required the user to read or write data. Thus, we had to determine whether these steps should be performed by the user interface of the TAS or whether they could be performed via the user interfaces of connected information systems which already existed. Then, a Domain Data model was designed in order to illustrate the logical dependencies between the different entities that should be handled by the TAS (e.g., transport wagons, casting boxes, etc). By using this model, we could clearly define rules that must always hold when using the TAS later.

Finally, we wrote Use Case notation in order to analyze the desired usage and interactions, and hence, the exact scope of the TAS. However, due to the highly integrative nature of the factory, it was necessary to specify also Use Cases for the process steps that were not supported by the TAS directly, because many TAS
functions were planned to be triggered implicitly through events in the connected systems.

Hence, the Use Cases on a more TAS-independent level were very important to understand the desired usage from the end users point of view as well as the seamless interplay between the TAS and its connected systems. All results of the requirements analysis were then incorporated into the system definition. However, we intentionally ignored the other decision points of TORE as we suggest that the supplier should describe all these missing aspects, mainly the System Functions, in his requirements specification.

Benefits. First of all, we experienced TORE as useful also in hybrid systems where both, hardware and software is required to make something happen. Second, we experienced the orientation on the plant’s tasks as inevitable for getting the “big picture” of the entire plant, especially across all systems involved. In particular, the identification of interfaces between the TAS and its connected external systems would not have been possible without this superior view. Furthermore, TORE enabled us not only to derive the requirements for the actual TAS but also to determine the required adaptations for the connected systems.

Third, the TORE approach facilitated the clear separation between system definition and system specification, and the communication between our customer and his supplier. Furthermore, the systematic elicitation and specification along the TORE decision points enabled our customer to find the right level of abstraction for the system definition.

Challenges. Constraints due to the building dimensions, the physical environment, the occupational safety regulations, or the technical infrastructure were quite important and had also to be incorporated into the system definition. Furthermore, non-functional requirements were also considered as essential. However, we could not elicit and specify them in an integrated manner with TORE, but we had to gather and describe them with separate approaches.

4. General Benefits and Drawbacks

Considering the above-mentioned benefits and challenges but also those recognized in other projects, we identified the following benefits of TORE:
Support for a systematic way of thinking without prescribed processes and notations. By focusing first on these decisions, TORE becomes widely applicable also in agile, market-driven, or reuse-oriented development settings. In particular, the decision points serve as the rationale to explain why specific process activities and artifacts are needed. Hence, there is a higher degree to follow a process or use a notation when it’s clear why to do the process step. Furthermore one can discuss which decision points need to be documented explicitly in the specification and which will be kept implicit.

Increased understanding of the domain, the tasks to be supported, current problems and thus facilitates the detection of improvement potential.

Systematic functional decomposition starting from a system-independent view on the domain. Thus, more consistent and complete requirements specifications can be achieved, in particular as the decision points are explicitly linkable by selecting appropriate notations. For instance, if the To-Be Activities are modeled with UML, and the System Responsibilities are depicted as stereotypes, one would recommend that for each a textual description must exist with the same name.

Separation of concerns by making abstraction levels and their dependencies clearer. In particular, TORE allows an abstraction from system issues allowing the system definition also for hybrid HW/SW-systems.

Integration of Usability / UI aspects into the requirements phase by connecting usability decisions with requirements decisions.

Supports for RE education and technology transfer through the conceptual definition of which information should be described in a requirements specifications. Furthermore, decision points are a good means to prevent the misuse of techniques, as they allow making clear which decision should be documented, respectively analyzed by a certain notation.

However, TORE is not a silver bullet, it faces some challenges. As the decision points are more abstract than process activities, or notations, one still has to determine the corresponding techniques, as well as to incorporate these choices into a coherent requirements process. The instantiations described in the case study might help in this regard. Furthermore, the decisions how As-Is Activities have to be optimized towards To-Be Activities remains still a creative issue. Finally, constraints and non-functional requirements are not (yet) reflected and have to be handled separately.

5. Conclusion

Based on our described experience with TORE, we are convinced that a goal and task-oriented requirement engineering approach is a promising and highly beneficial approach even in systems that do not seem to be traditional interactive systems at first glance. Especially the abstraction from concrete notations and processes has been experienced as a fruitful basis for systematic and highly applicable requirements engineering. Hence, we share the opinion that task-oriented requirements engineering approaches like TORE should receive more attention also in research. This includes research on how to address the challenges described in this paper.

References