

Use Case Modeling @ Fraunhofer IESE

| | |
|----------------|-----------------------------------|
| Sebastian Adam | Sebastian.Adam@iese.fraunhofer.de |
| Joerg Doerr | Joerg.Doerr@iese.fraunhofer.de |
| Anne Gross | Anne.Gross@iese.fraunhofer.de |
| Kizito Mukasa | Kizito.Mukasa@iese.fraunhofer.de |

1) Introductory Information on Deployed Modelling Process and Produced Artefacts

The Use Cases for this experiment were created by using an instance of the TORE approach as used at Fraunhofer IESE in Kaiserslautern, Germany. Here, we shortly summarize the main information that is needed to understand why we created the resulting artefacts in Chapter 2. Some further explanation on our usage of TORE is given in Appendix A. One should understand that the requirements specification according to TORE uses Use Cases as an essential specification artefact, but additional models are provided to fully describe the requirements. Furthermore, the requirements specification is aimed at describing an intended solution, making decisions on various levels of abstraction (i.e., it is not intended to be a purely problem oriented requirements specification).

In a first step, Stakeholders were identified based on the given problem description (PD). An exemplary stakeholder role is described using a basic role-definition template. Then, a goal model was produced from the PD: stakeholders specific problems were identified in the PD and translated and attached to goals. First solution concepts/ideas emerged and were attached to the goalmodel. For the To-Be Activities, we created a future (intended) workflow that was derived from the activity information (tasks) in the PD and the problems/goals as well as the first solution ideas in the goal model. The system-responsibilities were determined for the activities in the workflow model. A Use Case diagram was produced from the workflow and goal model. The classification used in EPCs and UC-Diagram is:

- UC in Systemboundary: functionality executed automatically by the system=System Activity (denoted “SA” in UC-Diagram and EPC)
- UC at border: Interaction of system with human=Human System Activity (denoted “HSA” in UC-Diagram and EPC)
- UC outside system boundary: pure human activity without system support = Human Activity (denoted “HA” in UC-Diagram and EPC)

In our methodology, we only create textual Use Cases for Human-System activities (i.e, a subset of the Use Cases in the Use Case Diagram) to determine how the user shall interact with the system. The Use Case Diagram was reworked after the textual use case description (introducing new use cases for making use of reuse opportunities). The Human Activities are not further refined. System Activities from the EPCs / Use Case Diagram as well as from the textual use cases are usually described with a system function template. As this was out of scope for this experiment, we only created a list of system functions. An Interaction Data Model was produced from PD and Textual Use Cases.

All (resulting) artefacts were not created in a waterfall like manner, but iteratively. This means that a redesign of higher level artefacts (EPCs, UC-Diagram) took place after a more detailed modelling (textual Use Case Description) took place.

As the effort for this experiment was limited, we did not produce a complete specification for all TORE artefacts. The following lists the artefacts produced and whether the artefacts can be seen as complete specification or exemplary specification:

- Stakeholder description (exemplary), in case of 1st line supporter: complete specification
- Goal Model (aimed for completeness)
- Workflow model (aimed for completeness)
- UC-Diagram (aimed for completeness)
- Textual Use Case Description (exemplary), HSA2, HSA3, HSA5 are completely specified Use Cases; HSA1, HSA4, HSA6, HSA7, HSA8, HSA9 are only working documents (not finalized). They are included in the submission as they can give a more clear idea of what these use cases will do.
- Interaction Data Model (exemplary)
- System Function list (exemplary)

We did not make use of the following TORE decisions and artefacts that are usually deployed:

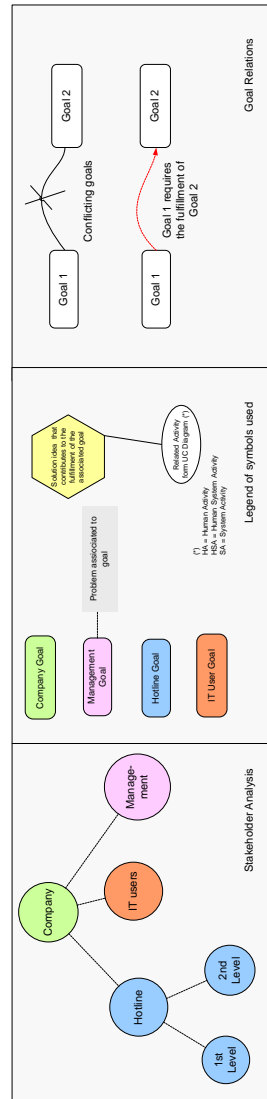
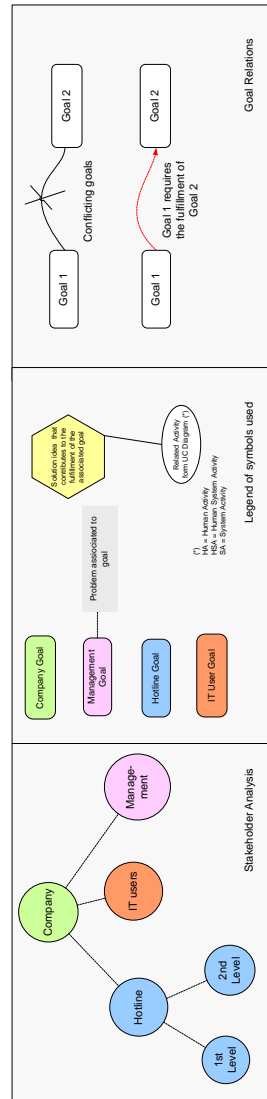
- As-Is Activities: not enough information was given in the PD
- Domain Data: as the data-model is so small, we decided to only use one model (Interaction Data)
- UI-Structure: out of scope for the experiment
- All system level decisions: out of scope for the experiment

2) Resulting Artefacts

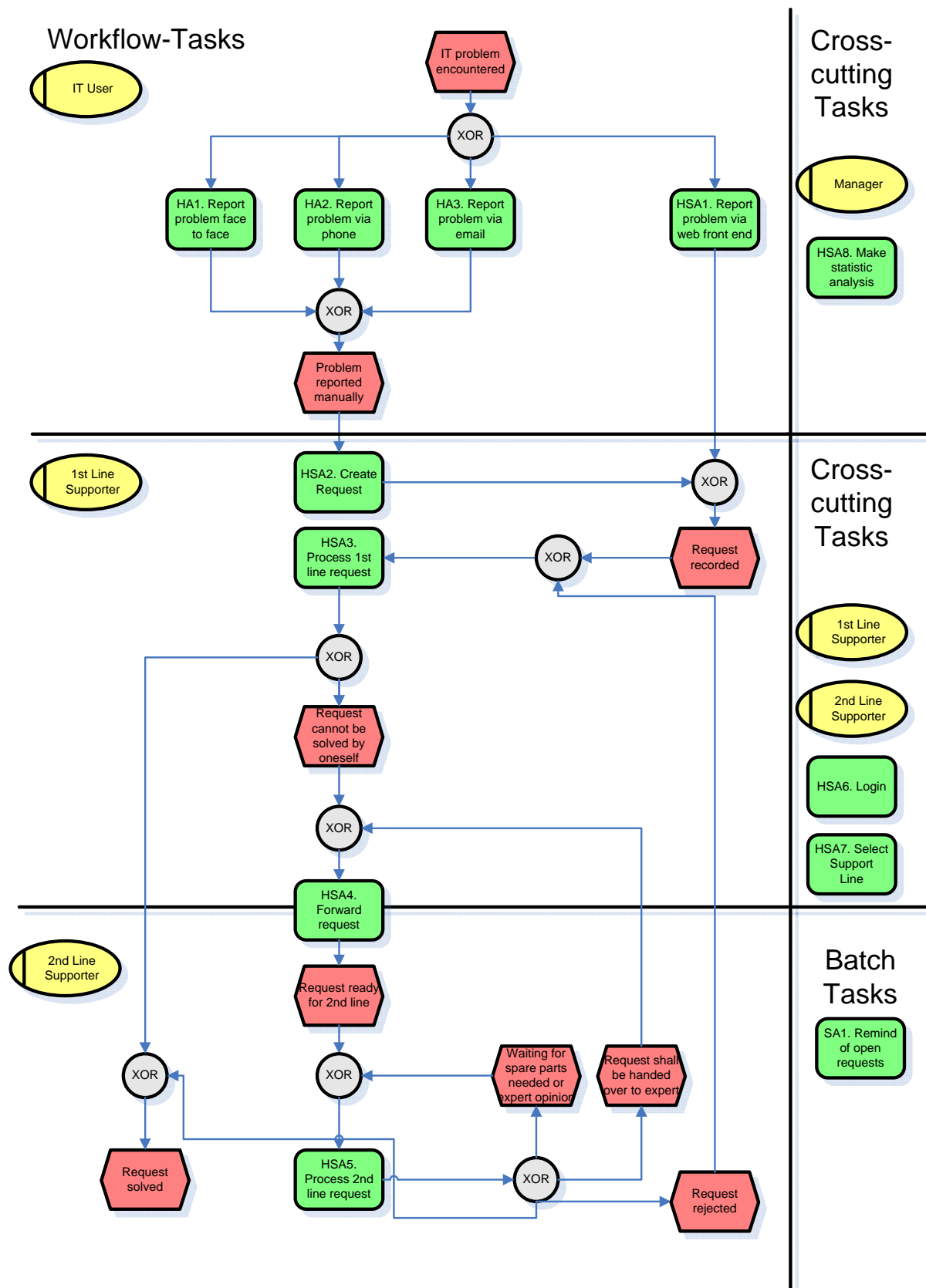
2.1) Stakeholder Role description

| RD1. First-line Supporter | |
|-------------------------------------|---|
| Responsibility | <p>The first-line supporter is a member of the hotline support team, receiving all incoming problems (by phone, via email, in person or via web-front end). For each incoming problem he / she has to create an appropriate request.</p> <p>In case that the first-line supporter is not able to process a request right away he/she forwards the request to a second-line supporter for further processing.</p> <p>Furthermore, the first-line supporter has the responsibility to reassign requests in case that a second-line supporter rejects a request due to overload, vacation or the second-line supporter is absent for a longer period of time.</p> <p>A first-line supporter can always switch to 2nd line support team, assumed that there is at least one person available in the 1st level support team.</p> |
| Success criteria | <ul style="list-style-type: none"> • Assure that for each problem a request is created in the system for statistical purpose. • Assure that there is at least one person in 1st line support team. • Avoid losing problems by reassigning request. • Efficient and effective transfer of problems to internal experts. |
| Tasks | <p>HSA2. Create request HSA3. Process 1st level requests HSA4. Forward request HSA6. Login HSA7. Select support line.</p> |
| Communication partner | <p>IT Users Second-line supporter</p> |
| Degree of innovation | Low |
| Existing Knowledge wrt. Tasks | <p>High knowledge on treating customers adequately. Not very detailed knowledge on solving problems.</p> |
| Existing knowledge wrt. Software/PC | Used to work with PCs and helpdesk support software. |

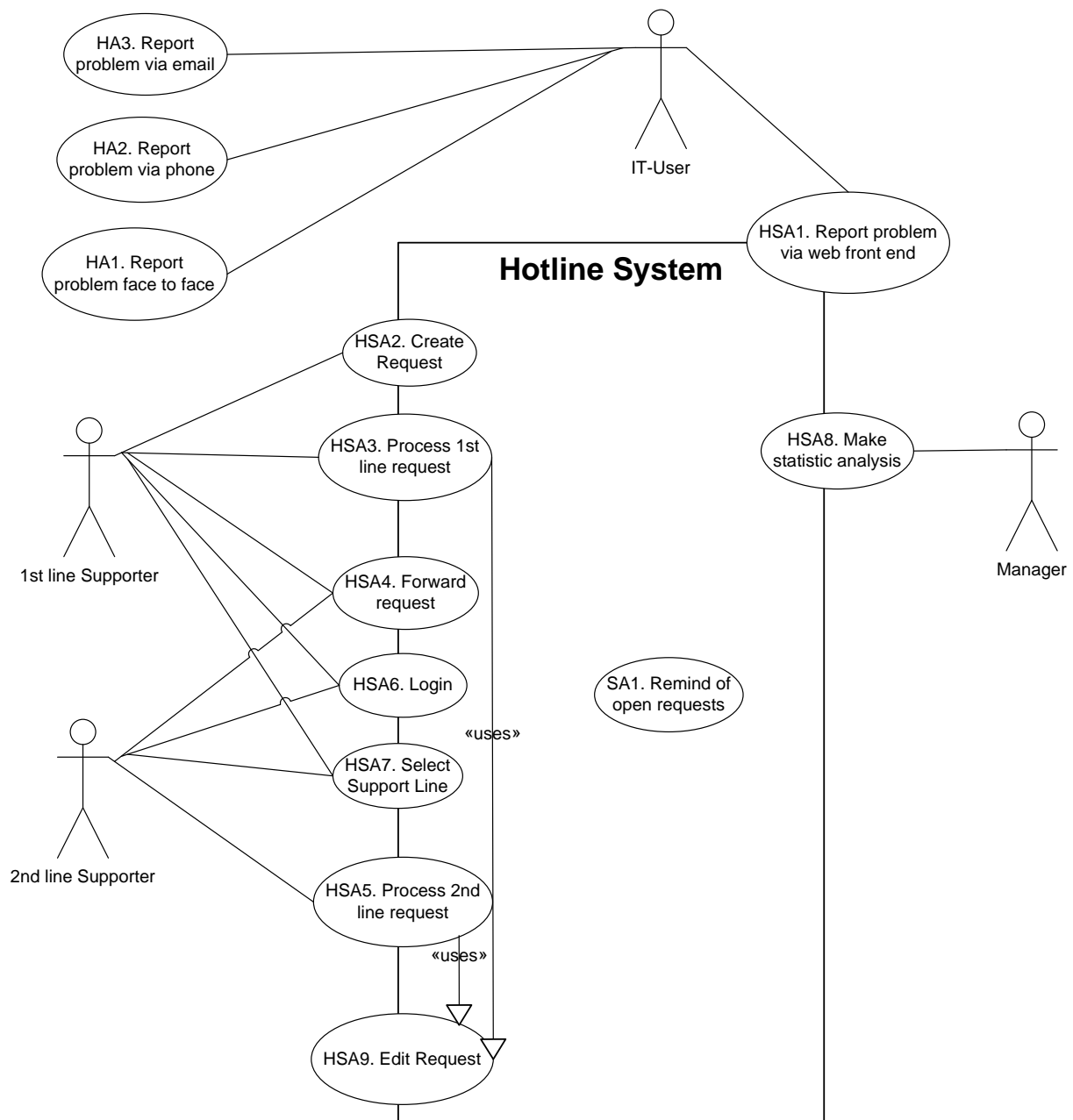
2.2) Goal Model



2.3) To-Be Workflow Model



2.4) Use Case Diagram



2.5) Textual Use Case Description

| | |
|----------------------|--|
| HSA1 | |
| Use Case Name | Report problem via web front end |
| Goal | Create and send problem request to hotline for processing. |
| Actor | IT-User |
| Preconditions | |
| Description | <i>This use case describes the functionality provided to the IT-User to manually create a new request, to specify relevant data (problem description, priority, etc) and send request to hotline support team.</i> |
| Exceptions | |
| Business Rules | |
| Quality requirements | <i>(Not in scope of experiment)</i> |
| Data (Data model) | |
| System Functions | SA6. Automatically create request |
| Postcondition | |

| | |
|----------------------|--|
| HSA2 | |
| Use Case Name | Create request |
| Goal | Record all incoming requests in the system. |
| Actor | First-line supporter |
| Preconditions | The Actor is logged in as First-Line Supporter (See Use Case HSA6. Login) |
| Description | <ol style="list-style-type: none"> 1. The Actor <ol style="list-style-type: none"> 1.1 receives request via phone or in person or via email 1.2 triggers a new request 2. The system requests the actor to specify data associated to the request. 3. The Actor <ol style="list-style-type: none"> 3.1 specifies data associated to the request 3.2 triggers the system to save the request 2. The system <ol style="list-style-type: none"> 2.1 saves the new request 2.2 records the request 2.3 displays the new request in the list of hotline requests 2.4 marks request as "first line" |
| Exceptions | - |
| Business Rules | <ul style="list-style-type: none"> • Only problems with high priority are allowed to be requested via phone or in person. • For statistical purpose it is not allowed to create a request for more than one problem. • The Actor must use the existing employee database to select the user during step 3.1. |
| Quality requirements | <i>(Not in scope of experiment)</i> |
| Data (Data model) | <ul style="list-style-type: none"> • User data (UserID) • Problem (Description, Category, Cause) • Request (RequestNumber, Priority, Creator (UserID of supporter), SubmissionTime) |
| System Functions | SA2. Provide template SA3. Save request SA4. Record request and processing information SA5. Update list of hotline requests SA9: Set state |
| Postcondition | The problem request is created in the system and can be further processed. |

| | |
|----------------------|---|
| HSA3 | |
| Use Case Name | Process 1 st line request |
| Goal | Process request as first-line supporter. |
| Actor | First-line supporter |
| Preconditions | The Actor is logged in as First-Line Supporter (See Use Case HSA6. Login). The hotline request list is not empty. |
| Description | <ol style="list-style-type: none"> 1. The Actor selects request in the list of hotline requests [rejected request] [absence of owner] [Actor wants to edit request] 2. The system <ol style="list-style-type: none"> 2.1 marks the request as taken 2.2 displays all details (attributes) of the request. 2.3 displays possible solutions 3. The Actor <ol style="list-style-type: none"> 3.1 decides on appropriate solutions [solution not available] 3.2 solves the problem [problem can not be solved right away] 3.3 assigns problem solution to request 3.4 adds additional comments / information to requests if needed 3.5 triggers system to save the request 4. The system <ol style="list-style-type: none"> 4.1 saves the request 4.2 records the request 4.3 marks the request as closed 4.4 updates the hotline request list 4.5 notifies the user that the problem has been solved |
| Exceptions | <p>[rejected request] or [absence of owner]</p> <ol style="list-style-type: none"> 1. The system displays all details (attributes) of the request 2. The Actor forwards request to second-line supporter (use Use Case HSA4.Forward request). <p>(end of Use Case)</p> <p>[Actor want to edit request]</p> <ol style="list-style-type: none"> 1. The Actors edits the request (see Use Case HAS 9. Edit request) <p>(end of Use Case)</p> <p>[problem can not be solved right away]</p> <ol style="list-style-type: none"> 1. The system displays all details (attributes) of the problem 2. The Actor forwards request to second-line supporter (use Use Case HSA4.Forward request). <p>(end of Use Case)</p> <p>[solution not available]</p> <ol style="list-style-type: none"> 1. The Actor adds new solution. 2. The system saves new solution. <p>Continue with step 3.2</p> |
| Business Rules | N/A |
| Quality requirements | <i>(Not in scope of experiment)</i> |
| Data (Data model) | <ul style="list-style-type: none"> • Request (Status, Owner (UserID of Supporter), AdditionalInformation) • Problem (All attributes) • Solutions (Category, Description) • Solution List |
| System Functions | SA4: Record request and processing information SA5: Update list of hotline requests SA7. Display request details |

| | |
|---------------|---|
| | SA8: Notify user SA9: Set state SA10: Assign solution SA11: Add new solution |
| Postcondition | A problem request has been successfully processed by the first-line supporter. |

| | |
|----------------------|---|
| HSA4 | |
| Use Case Name | Forward request |
| Goal | Forward request to expert. |
| Actor | First-line supporter Second-line supporter |
| Preconditions | |
| Description | <p><i>In case that a problem request can not be solved right away by a first-line supporter, the first-line supporter can forward the request to a second-line supporter who will then be the owner of the request.</i></p> <p><i>In case that a problem request has been rejected by a second-line supporter or in case that a second-line supporter is not available to further process a request, the first-line supporter can reassign the request to another second-line supporter.</i></p> <p><i>In case that a second line supporter can not solve the problem on his/her own, he/she can forward the request to another second-line supporter who will be the new owner of the request.</i></p> <p><i>In all cases relevant information is transmitted to the new owner and the changes are recorded for statistical purpose.</i></p> |
| Exceptions | |
| Business Rules | |
| Quality requirements | <i>(Not in scope of experiment)</i> |
| Data (Data model) | |
| System Functions | |
| Postcondition | |

| HSA5 | |
|---------------|---|
| Use Case Name | Process 2 nd line request |
| Goal | Process request as second-line supporter |
| Actor | Second-line supporter |
| Preconditions | The Actor is logged in as Second-Line Supporter (See Use Case HSA6. Login). At least one request is assigned to this second-line supporter. |
| Description | <ol style="list-style-type: none"> The Actor selects an assigned request (parked or new) in the list of hotline requests [vacation] [overload] [Actor wants to edit request] The system <ol style="list-style-type: none"> marks the request as taken displays all details (attributes) of the request. displays possible solutions The Actor <ol style="list-style-type: none"> decides on appropriate solutions [solution not available] solves the problem [problem can not be solved] [external expert is required] [help of internal expert required] assigns problem solution to request adds additional comments / information to requests triggers system to save the request The system <ol style="list-style-type: none"> saves the request records the request removes the request from the list of hotline requests displays the hotline request list notifies the user that the problem has been solved |
| Exceptions | <p>[vacation] or [overload]</p> <ol style="list-style-type: none"> The Actor <ol style="list-style-type: none"> rejects the request because of vacation optionally adds comments to the request The system <ol style="list-style-type: none"> Marks request as rejected Notifies first-line supporter Updates list of hotline requests Records rejection <p>(end of Use Case)</p> <p>[Actor want to edit request]</p> <ol style="list-style-type: none"> The Actors edits the request (use Use Case HSA 9. Edit request) <p>(end of Use Case)</p> <p>[solution not available]</p> <ol style="list-style-type: none"> The Actor adds new solution. The system saves new solution. <p>Continue with step 3.2</p> <p>[problem can not be solved]</p> <ol style="list-style-type: none"> The system displays all details (attributes) of the problem The Actor forwards request to another second-line supporter (use Use Case HSA4.Forward request). <p>(end of Use Case)</p> <p>[external expert is required]</p> <ol style="list-style-type: none"> The Actor <ol style="list-style-type: none"> contacts external expert optionally changes estimated time to solution (use Use Case HSA9. Edit request) |

| | |
|----------------------|--|
| | <p>1.3 sets status to “parked” (end of Use Case)</p> <p>[help of internal expert required]</p> <ol style="list-style-type: none"> The Actor <ol style="list-style-type: none"> contacts internal expert optionally sends relevant information to expert via email The system transmits the information to internal expert. The Actor receives response from internal expert <p>Continue with Step 3.2.</p> |
| Business Rules | N/A |
| Quality requirements | <i>(Not in scope of experiment)</i> |
| Data (Data model) | <ul style="list-style-type: none"> Request (Status, Owner (UserID of Supporter), AdditionalInformation) Problem (All attributes) Solutions (Category, Description) Solution List |
| System Functions | <p>SA4: Record request and processing information</p> <p>SA5: Update list of hotline requests</p> <p>SA7: Display request details</p> <p>SA8: Notify user</p> <p>SA9: Set state</p> <p>SA10: Assign solution</p> <p>SA11: Add new solution</p> <p>SA12: Reject request</p> <p>SA13: Transmit information to internal expert</p> |
| Postcondition | A problem request has been successfully processed by the second-line supporter. |

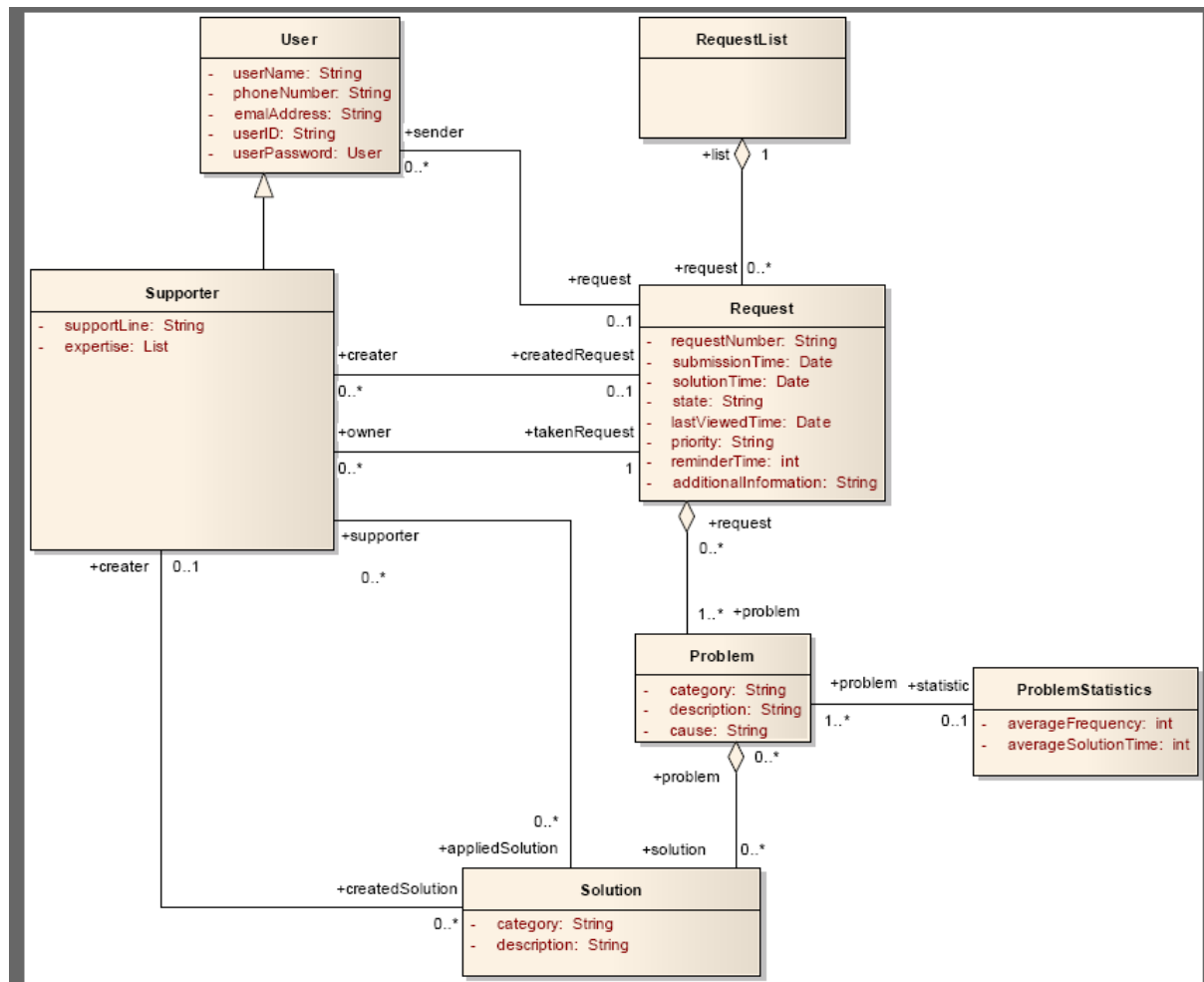
| HSA6 | |
|---------------------------------------|---|
| Use Case Name | Login |
| Goal | |
| Actor | |
| Supported goal from goal model | |
| Addressed problem(s) (see goal model) | |
| Preconditions | |
| Description | <p><i>This use case describes the login-functionality.</i></p> <p><i>Upon login, the user can decide whether he/she want to take over the role of a first-line supporter or a second-line supporter.</i></p> <p><i>Depending on the chosen role, the system provides the user with the respective view and functionality to process requests.</i></p> <p><i>Furthermore, the system provides the functionality to assure that there is at least 1 person in the first-line support team by automatically assigning the first person that logs into the system the role of first-line supporter.</i></p> |
| Exceptions | |
| Business Rules | |
| Quality requirements | |
| Data (Data model) | <ul style="list-style-type: none"> • Supporter (Support-Level) |
| System Functions | |
| Postcondition | |

| | |
|---------------------------------------|---|
| HSA7 | |
| Use Case Name | Select support level |
| Goal | |
| Actor | |
| Supported goal from goal model | |
| Addressed problem(s) (see goal model) | |
| Preconditions | |
| Description | <p><i>This use case describes the functionality to change the support-level (that is, to switch between the first-line and second-line support team).</i></p> <p><i>As in case with HSA6. Login the system provides the user with the respective view and functionality in accordance with the support level that the user selects.</i></p> <p><i>Furthermore, the system provides the functionality to assure that there is at least 1 person in the first-level support by notifying the user that switching to 2nd level is not possible if there is no one else left in the 1st level support team.</i></p> |
| Exceptions | |
| Business Rules | |
| Quality requirements | <i>(Not in scope of experiment)</i> |
| Data (Data model) | <ul style="list-style-type: none"> • Supporter (Support-Level) |
| System Functions | |
| Postcondition | |

| HSA8 | |
|---------------------------------------|--|
| Use Case Name | Make statistic analysis |
| Goal | |
| Actor | |
| Supported goal from goal model | |
| Addressed problem(s) (see goal model) | |
| Preconditions | |
| Description | <i>This use case describes the functionality of assessing statistics from a manager's point of view.</i> |
| Exceptions | |
| Business Rules | |
| Quality requirements | <i>(Not in scope of experiment)</i> |
| Data (Data model) | |
| System Functions | |
| Postcondition | |

| | |
|---------------------------------------|---|
| HSA9 | |
| Use Case Name | Edit request |
| Goal | |
| Actor | |
| Supported goal from goal model | |
| Addressed problem(s) (see goal model) | |
| Preconditions | |
| Description | <i>This use case describes the functionality to edit a request by a first-line or second-line supporter. This use case is used by the HSA3. Process 1st line request and HSA5. Process 2nd line request respectively, comprising functionality such as set reminder, add additional information, change priority or estimated time to solution.</i> |
| Exceptions | |
| Business Rules | |
| Quality requirements | <i>(Not in scope of experiment)</i> |
| Data (Data model) | Request (Reminder Time) |
| System Functions | |
| Postcondition | |

2.6) Interaction Data



2.7) List of System Functions

| ID | Name | Description |
|------|---|---|
| SA1 | Remind of open requests | The system automatically sends a reminder in case that the estimated time to solution is reached or a manually set timer is expired. |
| SA2 | Provide template | System functionality to specify data related to problem request |
| SA3 | Save request | System saves the request including attributes in the system |
| SA4 | Record request and processing information | System automatically records request as well as relevant processing information (such as change of ownership, duration of processing, rejection, etc) for statistical purpose |
| SA5 | Update list of hotline requests | System functionality to add requests, remove requests, or update information related to requests in the list of hotline requests. |
| SA6 | Automatically create request | System automatically creates request (including specification of relevant data, such as user, description). |
| SA7 | Display request details | The system displays detailed information related to a request (problem attributes) |
| SA8 | Notify user | The system automatically notifies the user in case that a process has been solved or in case that an estimated time to solution has been set / changed. |
| SA9 | Set state | The system sets / changes the state of a problem such as taken, parked, open, closed, first line, etc.) |
| SA10 | Assign solution | System functionality to look up and assign solution to request |
| SA11 | Add new solution | System functionality to add new solution. |
| SA12 | Reject request | System functionality to reject the request (including change of status ("rejected") and notifying first line supporter. |
| SA13 | Transmit information to internal expert | System transfers request data (problem, etc.) to internal expert (e.g., via email). |

3) Effort estimate

The effort needed to create these artefacts was estimated with 13 hours.

Appendix A: Short TORE description

TORÉ is a decision framework that encapsulates 18 decisions on four different levels of abstraction that typically have to be made during requirements engineering for 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 high applicability in many different contexts.

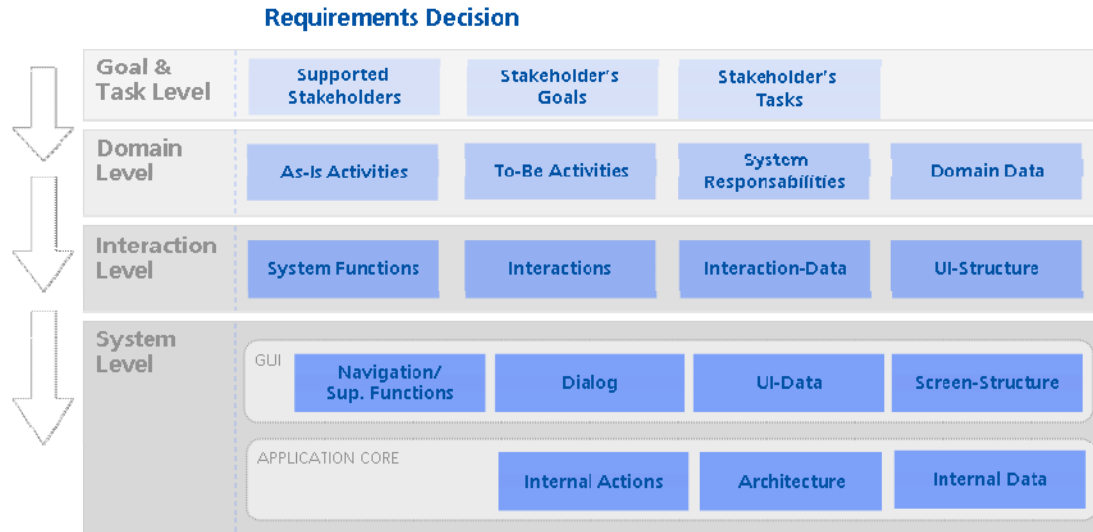


Figure 1. Decision points in the TORE framework

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 such as personas [14], or simple role descriptions. The second decision point is to capture which *Stakeholder's Goals* exist and shall be supported by the system. With TORÉ, 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 such as KAOS [12], i^* [13], or simple AND/OR goal refinement trees. Typically, the functional goals are refined into *Stakeholder's Tasks*. In a simple information system, the *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. At the *Domain Level*, each *Stakeholder's Task* is then refined into its *As-Is Activities*, i.e., the description of how tasks and processes are currently performed 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* are process modeling notations such as EPCs [8], or UML Activity diagrams [7]. With *System Responsibilities*, one then determines which of the *To-Be Activities* are performed automatically, and which are performed only by humans, respectively by humans using system support. Furthermore, *Domain Data* determine which data is handled on the *Domain Level*, respectively within the *To-Be Activities*. Typical notations are ER Diagrams or UML class diagrams. At the *Interaction Level*, the *Interactions* define for all system-supported *To-Be Activities* what the concrete usage of a system by a human should look like. Typical notations used for this decision point include Use Cases [4] or other scenario techniques. For all *System Functions* that are identified during the *To-Be Activities* and *Interactions*, the *System Functions* then describe the corresponding details (visible behavior, input, output, etc.). Furthermore, the *Interaction Data* determine the data used in *Interactions* and *System Functions*. Hence, they are typically a refinement of the *Domain Data*, using similar

notations. With regard to early UI design, the *UI-Structure* is a first logical grouping of functions and data, but with neither a detailed layout nor a modality decision. Typical notations used to document these decisions are workspaces as proposed in [15].

The aforementioned decision points (sometimes also the ones on the subsequent *System Level*, which are left out here) are the typical decision points that we use in order to determine the aspects to be discussed in our requirements engineering activities. More detailed information of TORE in general can be found in [6].

- [4] A. Cockburn, *Writing Effective Use Cases*, Addison-Wesley, 2000
- [6] B. Paech, K. Kohler, "Task-driven Requirements in Object-oriented Development", *Perspectives on Software Engineering*, Kluwer Academic Publishers, 2004
- [7] J. Rumbaugh, et al., *The Unified Modeling Language Reference Manual*, Addison-Wesley, 1998
- [8] G. Keller, M., Nüttgens, A.-W. Scheer, *Semantische Prozeßmodellierung auf der Grundlage „Ereignisgesteuerter Prozeßketten (EPK)“*, Universität des Saarlandes, 1992
- [10] S. Roberston, J., Robertson, *Mastering the Requirements Process*, Addison-Wesley Professional, 2006
- [12] A. Dardenne, A. van Lamsweerde, S. Fickas, „Goal Directed Requirements Acquisition“, *Science of Computer Programming*, Vol. 20 pp: 3-50, Apr. 1993
- [13] E.S.K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering", *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, IEEE, 1997
- [14] A. Cooper, R. Reimann, D. Cronin, *About Face 3.0: The Essentials of Interaction Design*, Wiley, Indianapolis, 2007
- [15] H. Beyer, K. Holtzblatt, *Contextual Design: Defining Customer Centered Systems*, Morgan Kaufmann Publishers, 1998