sections, we will briefly discuss each of these FAR process outputs.

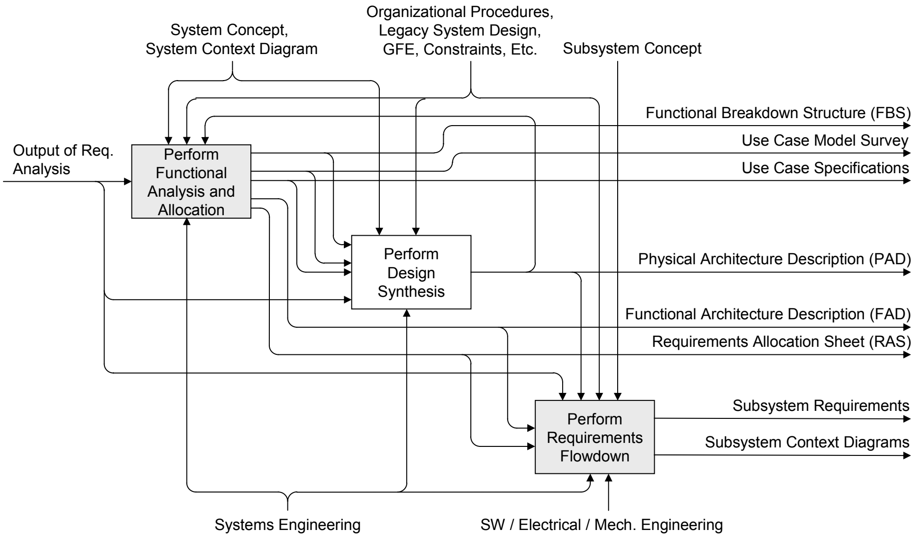


**Figure 1. Context diagram for the FAR approach in IDEFØ.**

**Functional Breakdown Structure (FBS).** The purpose of the FBS is to provide a good compact high level overview of the system functionality, which we feel, is missing in traditional use case models. As illustrated in Figure 2, this FBS should include all major function groups and services provided by the system-of-interest.

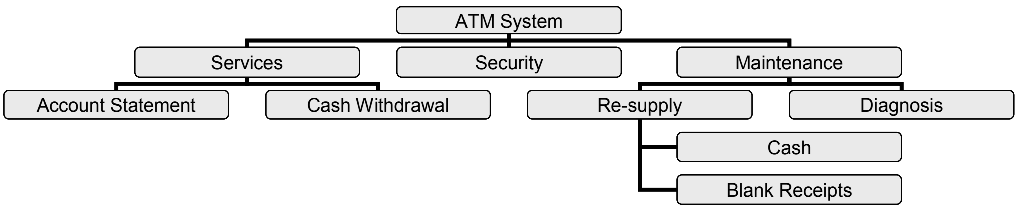


**Figure 2. Example FBS for an ATM system.**

**Use Case Model Survey.** The purpose of a use case model survey is to provide an overview of all use cases in a model. This report contains a system context diagram, descriptions of use case actors and UML use case diagrams, as well as brief descriptions of all use cases in the model (Eriksson, 2006).

**Use Case Specifications.** Use case specifications include both introductory information such as context information, pre- and post-conditions, as well as use case scenarios. These scenarios, in which the system is seen as a black-box, describe interactions between the system and its users. As illustrated in Figure 3, we describe alternative- and exceptional scenarios as so called scenario fragments (Adolph, 2003). These scenario fragments are specified as deltas to the main success scenario. The first step identifier of the fragment determines the position where the fragment should be inserted into the main success scenario. A fragment can either return to the main success scenario or terminate the use case itself.

**Main Success Scenario (MSS)**

| Step | Actor Action | Blackbox System Response | Budgeted Req. |
|---|---|---|---|
| 1 | The use case begins when the ***Customer*** provides a General_ID. | The ***System*** accepts the general ID and requests a Unique_ID from the ***Customer***. | Max response time 1 sec. |
| 2 | The ***Customer*** provides a Unique_ID. | The ***System*** accepts the Unique_ID and requests the ***Bank Mainframe*** to validate the General_ID and the Unique_ID. | Max response time 0.5 sec. |
| 3 | The ***Bank Mainframe*** verifies that the Unique_ID is valid. | The use case ends when the ***System*** present available services to the ***Customer***. | Max response time 0.5 sec. |

**Exceptional Scenarios**

**Incorrect Unique ID**

| | | | |
|---|---|---|---|
| 3 | The ***Bank Mainframe*** notifies the system that the Unique_ID is not valid. | The ***System*** presents an error message to the ***Customer*** and requests a new Unique_ID. | Max response time 0.5 sec. |
| 4 | This use case continue at MSS step 2. | --- | --- |

**Incorrect Unique ID, General ID captured**

**Preconditions**

The customer has provided an invalid Unique_ID twice during the authentication session.

| | | | |
|---|---|---|---|
| 3 | The ***Bank Mainframe*** notifies the ***System*** that the Unique_ID is not valid. | The use case ends when the ***System*** capture the General _ID and presents an error message to the ***Customer***. | Max response time 0.5 sec. |

**Figure 3. Example black-box scenario descriptions.**

**Functional Architecture Description (FAD).** A FAD is a collection of all use case realizations for a system. A use case realization is a description of how different subsystems collaborate to solve a specific use case (Kruchten, 2000), in other words, a white-box description of a use case. As shown in Figure 4, our notation for use case realizations prescribes that each black-box step of a use case scenario is decomposed into a number of so called white-box steps. These white-box steps describe how the different subsystems collaborate to solve each black-box step.

**Incorrect Unique ID, General ID captured**

**Preconditions**

The customer has provided an invalid Unique_ID twice during the authentication session

| Step | Actor Action | Blackbox System Response | Budgeted Req. | Whitebox Action | Whitebox Budgeted Req. |
|---|---|---|---|---|---|
| 3 | The ***Bank Mainframe*** notifies the ***System*** that the Unique_ID is not valid. | The use case ends when the ***System*** capture the General_ID and presents an error message to the ***Customer***. | Max response time 0.5 sec. | The ***MainframeInterface*** accepts the result of the authentication attempt from the ***BankMainframe*** and notifies the ***TransactionControler*** that the Unique_ID is not valid | Max response time 0.1 sec. |
| | | | | The ***TransactionControler*** requests the General_ID to be captuered by the ***CardReader***. | Max response time 0.2 sec. |
| | | | | The ***TransactionControler*** requests an error message to be presented by the ***OutputDevice***. | Max response time 0.1 sec. |
| | | | | The ***OutputDevice*** presents the error message to the ***Customer***. | Max response time 0.1 sec. |

**Figure 4. An example use case realization; white-box steps to solve the last fragment of the black-box description of the use case above.**

**Requirements Allocation Sheet (RAS).** The purpose of the RAS is twofold; (1) it is used to maintain traceability between system requirements and use cases, and (2) it is used for allocating those system requirements, which are not suitable for use case modeling, typically quality attributes, to subsystems. Please note that traditional use cases are very poor at handling nonfunctional requirements.

Use cases
Allocation
UC &Allocation

| | | Use Cases | | | | | Subsystems | | | | | | Allocation Rationale |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Req. ID | Req. Text | Withdraw Cash | Get Account Statement | Authenticate Customer | … | … | Mainframe Interface | Transaction Controler | Card Reader | Output Device | … | … | |
| **Services** | | | | | | | | | | | | | |
| SYSR867 | The system shall provide a cash withdrawal service. | X | | | | | | | | | | | |
| SYSR868 | The system shall provide a account statement service. | | X | | | | | | | | | | |
| … | … | | | | | | | | | | | | |
| **Security** | | | | | | | | | | | | | |
| SYSR870 | The system shall capture the General_ID after three consecutive failed authentication attempts. | | | X | | | | | | | | | |
| … | … | | | | | | | | | | | | |
| **Maintenance** | | | | | | | | | | | | | |
| SYSR900 | It shall be possible to replace any malfunctioning subsystem within 15 minutes of… | | | | | | X | X | X | X | X | X | Some rationale for the allocation… |
| … | … | | | | | | | | | | | | |

**Figure 5. An example Requirements Allocation Sheet.**

**Subsystem Requirements.** In the FAR approach, specifying subsystem requirements is the responsibility of domain engineers. Systems engineering is, however, part of this process to ensure that the resulting subsystem requirements still represent the intent of the original system level requirements and their corresponding allocations. The analysis of the FAD and RAS should result in one or more subsystem requirements being specified for each:

- Subsystem appearing in a FAD white-box step. As shown in Figure 6, these subsystem requirements are typically of input, output or functional type.
- 'X' in the "Subsystems" part of the RAS.

**Incorrect Unique ID, General ID captured**
**Preconditions**
The customer has provided an invalid Unique_ID twice during the authentication session

| Whitebox Action | Whitebox Budgeted Req. | Subsystem Requirement |
|---|---|---|
| The ***TransactionControler*** requests the General_ID to be captuered by the ***CardReader***. | Max response time 0.2 sec. | The ***TransactionControler*** shall produce a request to capture a General_ID within 0.15 sec, if an invalid Unique_ID have been provided three times. |
| | | The ***CardReader*** shall accept requests to capture a General_ID within 0.05 sec. |
| | | The ***CardReader*** shall capture a General_ID on request. |

**Figure 6. Example of subsystem requirements derived from use case realizations.**

## STUDY DESIGN

The method evaluation was designed as a *blocked subject-project study* (Seaman, 1999), where data was collected from two different pilot projects. The study was preformed with the Swedish defense contractor BAE Systems Hägglunds AB, a leading manufacturer of combat vehicles, all terrain vehicles and a supplier of various turret systems. The hypothesis to be tested and its null hypothesis were:

**$H_1$:** The FAR approach performs better than managing the functional view according to the company process baseline in the current industrial setting.

**$H_0$:** The FAR approach performs equal to, or worse than, managing the functional view according to the company process baseline in the current industrial setting.

A number of response variables relevant for measuring the performance of the approach were identified as part of the study design. Examples of such response variables were usefulness of the resulting models as means for communication, and support for achieving high levels of reuse.

## Study Context

The two pilot projects studied during the evaluation regarded development of a turret system and a vehicle system. The systems engineering team of the first of these projects utilized the provided tool support (Eriksson, 2006) from the start. The second project team on the other hand, first started using only word processors and moved to the provided tool environment only after approximately 50% of the use case descriptions were already finished. At the time of the study, the first of these projects was in the process of finalizing the systems engineering output, and the second project had already moved on to detailed design and development.

The company process baseline for managing the system functional view prescribes development of two artifacts, the "Functional Description" and the "Functional Specification" The *Functional Description* is a traditional document, structured according to the system physical architecture, where all logical subsystems have a corresponding heading in the document outline. Each of these subsystem's functions is described with logical interfaces, possible states, conditions and exceptions. There are also references to functions allocated to other subsystems in each section. The purpose of these references is to enable tracing of functions from the originating subsystem throughout the rest of the system. The *Functional Specification* allocates the logical interfaces and logical subsystems described in Functional Description to physical interfaces and physical components. The Function Specification is also document based. Traceability between the two documents is maintained by keeping a common heading outline. A problem with this approach was that two entire documents had to be read, front to back, to understand a single function. If only parts of the documents are read it may leave some parts of the function open for interpretation. Another problem was that these descriptions where too tightly coupled to the physical architecture. Reusing parts of these documents in other projects would require considerable restructuring to make them fit the architecture of the target project. In practice, this made reuse almost pointless.

## Data Collection Methods

**Documentation Examination.** The modeling artifacts were inspected to identify possible common modeling errors and misunderstanding with FAR. We also examined inspection records for these artifacts to identify possible common errors that were captured during inspections. Furthermore, a number of meeting minutes and product documents from two historical projects that developed similar products to the pilot projects were reviewed. A better understanding of both the organization process baseline and the systems being developed could thereby be gained.

**Participant Observation.** The research team assumed a mentoring role for the systems engineers applying the FAR approach. This mentoring activity consisted of both answering questions via phone and email, and by participating in a number of modeling sessions. This enabled us to get first hand information regarding possible problems with FAR.

**Questionnaires.** All systems engineers in the study, in total six persons, also filled out questionnaires. These questionnaires collected data regarding their experience applying the approach and their views on the tools and notations used. The main purpose of these questionnaires was to elicit information about possible confounding factors that could influence the evaluation. The information was later taken into account during data analysis. However, questionnaires were designed to have both specific and open ended questions to also elicit unexpected types of information.

**Interviews.** A total number of 11 subjects, representing systems engineering, electrical engineering and software engineering were interviewed during the study. The purpose of these interviews was to gather their views on the usefulness of the resulting models, and on possible pros and cons, problems and risks with FAR.

Interviews began with a short introduction to the research being performed. After the introduction, the FAR process descriptions and artifact examples presented in (Eriksson, 2006) were shown and discussed with each interviewee. Interviews then proceeded in a semi-structured manner, to elicit as much information as possible about opinions and impressions regarding FAR.

## Data Analysis Method

The different types of data were first analyzed individually to find patterns and trends in the responses. The data was then groped in two stages and analyzed again:

- Per development project, to capture views related to possible differences between the projects in regard to how the method was applied.
- Per engineering discipline, to capture differences between the views of software-, electrical- and systems engineering staff.

All data was then combined to evaluate the hypothesis.

## Threats to Validity

**Construct validity.** To minimize threats to the study's construct validity, data was collected using several different methods that also allowed unexpected types of information to be elicited. Furthermore, the case study hypothesis and its null hypothesis were stated as clearly and as early as possible in the case study design to aid in identifying correct and relevant measures (Kitchenham, 1995).

**Internal validity.** To minimize threats to the study's internal validity, pilot projects were staffed using the organizations normal staff-allocation procedures. The study also captured information regarding each subject's level of experience from the company process baseline and the information was taken into account during data analysis (Kitchenham, 1995). Furthermore, subjects were chosen in collaboration with the organization's management to ensure that they properly represented their group of stakeholders. All chosen subjects agreed to participating in the study. To avoid the *Hawthorne effect* (Mayo, 1933), attitudes towards the company process baseline were collected from subjects and taken into account during data analysis. It was also pointed out to subjects that no "correct" answers existed, and that it was important that their answers correctly reflected their views.

One confounding factor that may have affected the internal validity of the study is the close involvement of the research team with the systems engineering teams. We do however judge this risk to be minor since the teams performed most of the actual modeling themselves.

**External validity.** To minimize threats to the study's external validity, the study was conducted in the target domain of long-lived software intensive systems and projects were selected to be of typical size and complexity for the organization (Kitchenham, 1995).

**Conclusion validity.** To minimize threats to the study's conclusion validity, results were triangulated by collecting data with four different methods from several different sources (Seaman, 1999). Furthermore, *member checking* (Seaman, 1999) was performed by allowing subjects to review and comment on draft versions of this paper to assure that their opinions were correctly represented.

One confounding factor that affects the conclusion validity of the study is the limited types of stakeholders that participated in the study. By only including systems engineering, software engineering and electrical engineering, opinions of other important stakeholder groups were not considered. We discuss this issue further in the future work section of this paper.

## RESULTS

### Examining Documentation

Document examination indicated that the teams had problems identifying "good" goal-oriented use cases. It was often possible to sense the structure of the formerly used Function Description in the resulting use case models. This meant that use cases often ended up being subsystem task oriented instead of being end-user goal oriented. This in turn led to a very flat FBS with too many and too small use cases.

Other indicators of structural problems were the existence of use cases named "*Handle…*" and "*Manage…*" In such use cases, alternative scenarios where typically used for describing different task within specific groups of functions, instead of describing different ways of achieving goals of primary use case actors. This led to problems with the goal oriented use case specification template. Examples of such problems were the need of specifying different pre- and post-conditions for each scenario and problems describing exceptional scenarios.

One team also experienced problems maintaining a consistent abstraction level in their use case scenario descriptions. This team hade chosen not to develop a FAR system context diagram (Eriksson, 2006), and we believe this to be a major reason for this problem.

### Participant Observation

Also participant observation indicated that it was hard to identify "good" use cases. Another problem noticed was that it seemed to be hard to keep use case scenarios free from unnecessary MMI design constraints, even though they should be free of such details. Often knowledge from earlier systems and possible prototypes had a tendency to influence the descriptions and thereby more or less force the same interaction methods and sequences.

It was also noticed that teams experienced problems defining stimuli-response in use case scenarios for functions with little information handling, i.e. functions which typically end up with mechanical implementations. Examples of such functions from our domain are applying the turret transport lock, manual laying of the weapon, etc. This problem was however resolved during the mentoring meetings when the issue was discussed and the teams agreed that also mechanical functions have input/output information such as for example requested speed, torque, etc.

Teams also experienced problems modeling functions with extensive information handling. Examples of such functions are reading logs, manuals, user instructions, etc. In our system architectures, these types of functions are typically allocated to a single subsystem, the Vehicle Information System (VIS). Modeling these functions, teams often ended up describing how functions are to be performed, instead of simply stating that they are performed. This can impose unnecessary constraints on the detailed design of VIS and might also result in redundant information in the model, since subsystem-level details are ending up in system-level specifications. Analyzing this issue, it became clear that this problem could affect any function that to a large extent is allocated to a single subsystem. This is a severe problem since it is, without good domain knowledge, basically impossible to catch before allocation is performed. However, once allocation is done it is relatively simple to identify such problems and restructure the model to correct them. The rule-of-thumb to use in these cases is: If several consecutive use case realization (white-box) steps are equivalent to their respective use case scenario (black-box) steps, merge these black-box steps into a single step, by describing the collected intent of all the original steps.

Teams also experienced problems describing functions that relied on using the intercom system. Common practice in use case modeling says that only interaction between the system and its actors are relevant information to capture in scenarios. However, applying this practice may render scenarios that are very hard to read, and alternative scenarios that are hard to understand why they were initiated. For example, consider the scenario in Figure 7. Depending on the information passed via the voice channel between the actors in step 2, the remainder of the scenario may look very different. So, even though this information is hidden from the system and therefore typically would have been excluded from the description, it can have large impact on the remainder of the scenario. Our recommendation is therefore to include such interaction between use case actors if it improves the readability of scenarios. However, such interactions between actors should be merged with the next actor action that actually results in a system response. The reason for this is to not violate one of our good practice guides that all actor action steps of a scenario should result in a system response.

| Step | Actor Action | Blackbox System Response |
|---|---|---|
| 1 | The use case begins when the ***Commander*** requests a voice channel to the ***Loader***. | The ***System*** opens a voice channel to the ***Loader*** and indicates that the channel is active to the ***Commander***. |
| 2 | The ***Commander*** orders some action to be performed by the ***Loader*** and then requests the voice channel to be closed by the ***System***. | The ***System*** closes the voice channel and indicates that the channel is closed to the ***Commander***. |
| 3 | The ***Loader*** requests some action from the ***System***. | The use case ends when the ***System*** has processed the request. |

**Figure 7. An example scenario with communication between actors.**

Yet another problem noticed during the modeling sessions was a tendency to model end-user business processes instead of system usage. This phenomenon was known as the “should/could-problem” within the organization. In defense systems there are often requirements stating that the same functions, for example fire, should be available to several of the system users. Modeling such functions, teams tried to model how they believed the system *should* be used, instead of modeling how the system *could* be used which is more relevant from the perspective of finding subsystem requirements. Modeling *should usage* as normal and *could usage* as exceptions, can also lead to new requirements being invented, for example regarding different usage modes. However, knowledge about intended usage is important information to capture, for example as input to the human factors team. Our recommendation is therefore that use case scenarios shall describe could usage. However, intended usage should be captured as background information in the introductory part of a use case specification.

A question raised during the mentoring sessions was how to handle Government Furnished Equipment (GFE). Applying common practice in use case modeling would result in viewing GFE as use case actors. The reason for this is that GFE is not developed by the organization and therefore external to the system-of-interest. This would however in some cases result in scenarios with much interaction between actors, and little interaction between actors and the system. Such scenarios are according to our experience not very intuitive and provide little support for design synthesis. Consider for example if the intercom system used in the scenario in Figure 7 would be GFE and seen as a use case actor instead of part of the system. This would result in the scenario shown in Figure 8. Our recommendation is therefore to view GFE as subsystems instead of actors. One exception from recommendation is however GFE that provides external interfaces to the system-of-interest on the next higher system level. In our domain this could for example be a Battlefield Management System (BMS). Such GFE should instead be viewed as use case actors. The reason for this exception is that such a view enables systems engineers to abstract and only analyze the internal GFE interface, and not what is on the external side of the interface.

| Step | Actor Action | Blackbox System Response |
|---|---|---|
| 1 | The use case begins when the ***Commander*** requests a voice channel to the ***Loader*** from the ***Intercom***.<br>The ***Intercom*** opens a voice channel to the ***Loader*** and indicates that the channel is active to the ***Commander***.<br>The ***Commander*** order some action to be performed by the ***Loader*** and then request the voice channel to be closed by the ***Intercom***.<br>The ***Intercom*** closes the voice channel and indicates that the channel is closed to the ***Commander***.<br>The ***Loader*** request some action from the ***System***. | The use case ends when the ***System*** have processed the request. |

**Figure 8. Seeing the intercom system as a use case actor.**

Another problem noticed was that experienced software engineers in particular could get confused by FAR's "usage" of use case modeling. They assume, based on their experience from software development, that developing use case realizations is detailed design and should result in identified classes. However, since FAR utilizes use case realizations for the purpose of functional allocation as part of the systems engineering process, the goal is not to identify classes but rather to identify subsystem requirements. Similar confusion was also noticed in other parts of the organization. People that had not been directly involved in the pilot projects had a tendency to believe that FAR specifications were pure software specification and therefore a software engineering rather than a systems engineering responsibility. We believe that these problems are related to the fact that BAE Systems Hägglunds AB develops software according to a tailored version of the IBM-Rational Unified Process (Kruchten, 2000), which utilizes similar modeling concepts as the FAR approach.

## Questionnaires

Questionnaires showed that a majority of the systems engineers considered the major artifacts of the FAR approach either useful or very useful (see Figure 9). Questionnaires also showed that subjects considered the FAR notations to be relatively simple to understand and use.

Furthermore, questionnaires also showed that subjects considered the FAR tool support (Eriksson, 2006) to have a number of positive characteristics. Examples of these characteristics are strong support for traceability and the possibility to maintain one, well organized and version controlled database with all the systems engineering process outputs. From the answers, it was however obvious that the graphical user interface of DOORS was causing a lot of annoyance among the subjects. Examples of problems pointed out were the lack of shortcut keys, user interfaces bugs and that the user interface does not follow MS Windows standards.

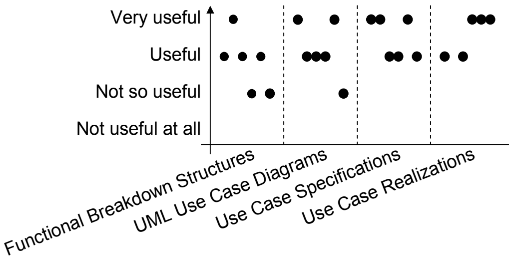


**Figure 9. Usefulness of artifacts according to questionnaires**

## Interviews

**Systems engineering.** Interviews with systems engineers showed that they considered FAR to have a number of positive characteristics. Examples of theses positive aspects pointed out during the interviews were:

- The FAR approach is better at giving the whole picture. It provides more context information by answering the question: “Why is this function important?”
- Being end-user oriented, the resulting artifacts provide better input to the ILS team responsible for developing user instructions and manuals.
- It provides a good and powerful framework for analyzing functionality using end-user goals. This goal-orientation also means that the approach provides better input to the human factors team.
- Appling the FAR approach provides better means of understanding the product in contrast to only understanding the system.
- It has stronger support for finding unclear requirements and other issues so that they can be clarified early in the development life cycle.
- Being scenario based, it provides stronger means for early architecture evaluations.
- Focus on traceability makes it obvious where subsystem requirements come form.
- Good that domain engineers are specifying subsystem requirements to force a deeper understanding of the requirements.
- Generally easy to work with notations used.

Systems engineering also identified a number of negative characteristics and problems applying the FAR approach:

- When developing FAR models, more access to specialists is required. It can sometimes also be hard to get these specialists to document their knowledge.
- The FAR approach is more formalized than the previous approach which makes everything slower.
- The FAR approach is a more advanced way of managing the functional view. This means

that it takes more time to get everyone to understand the methodology.

- Some functions are hard to describe as use cases. Examples of such are functions triggered by timers or sensors.
- It is hard to identify good use cases and scenarios. Mining information from historic projects often results in too detailed scenarios.
- Maintaining a “single voice” and consistent abstraction levels when several people where involved in writing use cases is problematic.
- Some team members have earlier experiences of use case modeling which differs from the FAR approach to use case modeling. This leads to problems agreeing on the use case model structure.
- The FAR approach is harder to perform if telecommuting, as it seems to work best as teamwork.

Systems engineering also identified a number of risks using the FAR approach compared to the previous document based approach:

- Using more advanced tools and methodology might cause very skilled people with a lot field experience in for example the verification team or the production team to be left outside the loop.
- Teams new to the methodology may develop use case models that have an unsuitable structure due to previous experiences from use case modeling. They should therefore have access to a methodology mentor and have scheduled mentoring meeting to force them to ask questions.
- Using a more formalized approach may cause domain engineers to get too trusting when reading specifications. This could cause them not to notice if domain specialists forget to put something in a specifications.

Even considering these drawbacks, problems and risks, all systems engineers in the study agreed that the FAR approach is a better method than the one used before. They considered the FAR approach to be an efficient way of managing the functional view, and that it is the right approach to keep following also in future projects. All systems engineers in the study also agreed that most of the problems experienced could be related to the fact that the methodology had been applied in pilot projects with too few good examples and instructions to follow.

**Software engineering.** Also software engineering expressed a number of positive characteristics of the FAR approach during interviews:

- FAR models provide a good overview of the functional scope of the system and it puts functions in their correct context.
- The FAR approach is better at bringing unclear requirements up on the agenda early in the project and could thereby make detailed design much more efficient. It furthermore provides strong support for finding software requirements, as long as the quality attributes are not forgotten.
- The FAR approach provides better support for reuse of specifications between projects, since the resulting artifacts are structured around end-user goals rather than the physical architecture. This provides a possibility for the organization to significantly reduce its development costs in the future.

Software engineering mainly pointed at two negative characteristic of the FAR approach compared to the previous method used. The first of these was that it is harder to verify that everything fits together using several small artifacts instead of one large artifact. The second was that it is much harder to write a good use case than to fill out sections in a traditional document structure. This fact significantly reduces the resources available that are able to do the job.

Software engineering also pointed at a risk with the approach. Since these artifacts are better means of communication with customers, it may provide customers with better opportunities to sneak in new requirements during common reviews. This may also lead to a lot more time being spent on negotiating with customers.

Also all software engineers in the study agreed that the FAR approach is generally better than the earlier method used and that the FAR approach is the right path to follow also in future projects.

**Electrical engineering.** Also electrical engineering expressed a number of positive characteristics of the FAR approach during interviews:

- The FAR approach is better at forcing people to really understand the intent of the requirements.
- Goal and end-user focus provide stronger means to find "good" system solutions.
- The resulting artifacts are also good input to the ILS team writing user instructions.
- FAR models works very well for finding subsystem requirements.
- It provides good support for early architecture evaluations from the perspectives of necessary interfaces, system safety considerations, etc.

Drawbacks with the FAR approach compared to the previous method raised by electrical engineering was that the FAR approach requires access to a lot of specialists and that it might require more time to finish specifications.

A problem mentioned by electrical engineering was that it can sometimes be hard to understand complex scenarios in the notation used, and that a supplementary diagram would be very useful in those cases.

Electrical engineering identified mainly two risks with applying FAR. The first risk is that if development of the FAD is not performed by multi-disciplinary teams, it might result in unrealistic specifications. The second risk is that if the FAD is not mature enough at the start of the detailed hardware design it will not be used. Because of the long delivery times on hardware, it is not possible to delay the detailed design to wait for these specifications.

Also all electrical engineers in the study agreed that the FAR approach is generally better than the earlier method used and that FAR is the right path to follow in future projects.

**Notations used.** Interviews also elicited information regarding possible shortcoming of the notations used for describing use case scenarios and use case realizations. The subjects identified the following problems:

- It is hard to describe conditions that have to be fulfilled only during part of a scenario. The use case template provides the possibility to specify pre-conditions for a scenario, but no intuitive way to specify for example that the dead-mans-grip has to be held down during step four to step seven.
- The semantics of the notation is unclear regarding whether a scenario is just one way of performing the function or a required sequence of actions. For example a drive sequence: [accelerate → use turning signal → turn → use horn → break], versus a fire sequence: [aim → arm → fire]. How to specify if the order of sequence is important or not?
- It is hard to describe state dependent behavior.
- Using scenario fragments may result in a single alternative scenario, which should contain several unconnected fragments, describing several deltas to the main success

scenario. There is no intuitive way to describe this situation in the notation.

**Usefulness of artifacts.** All subjects in the study considered the *System Context Diagram* to be an important artifact for the following reasons:

- The context diagram makes everything easier; it is inefficient to work without it.
- It is important to agree on naming conventions and scope early; the context diagram is helpful in this area.
- The context diagram improves the quality of use cases since it specifies all entities that are allowed to appear in use case scenarios.

A majority of the systems engineers, all software engineers and a majority of the electrical engineers in the study considered the *Functional Breakdown Structure* to be an important artifact for the following reasons:

- It is important to agree on the model structure early.
- It provides a good overview of the system's capabilities.
- It provides help to find use cases, and also to say - Stop, we found all use cases.
- It provides context to the use cases in the model.
- It is fun to develop.
- It helps to keep track of the progress of the systems engineering effort.

A majority of the systems engineers, half of the software engineers and a majority of the electrical engineers in the study considered the *Use Case Model Survey* to be an important artifact. Reasons given for this were that it provides a good overview of the use cases in the model and that it helps make work more controlled. The general comment form subjects that did not consider it to be an important artifact, was that it is not worth the effort to develop, since all information is basically also present in the use case specifications.

All subjects considered *Use Case Specifications* to be important artifacts for the following reasons:

- They force people to really think about how the system is intended to be used in the field.
- They are good for reuse and change impact analysis, when developing a new system within the same product line.
- They are an effective way to capture the functional view of a system.

A problem with use case specifications pointed out during interviews was, however, that they sometimes were considered almost ridicules by specialists because of their lack of implementation details.

All subjects considered *Use Case Realizations* to be important artifacts for the following reasons:

- They are an effective way to perform functional allocation.
- They lead to higher reusability of the resulting solutions.
- They are a good tool for architecture evaluation.
- They are a good source for clear subsystem requirements.
- They provide good input to the verification team when specifying test cases.
- They provide a good basis for detailed hardware design.

Problems pointed out during interviews regarding development of use case realizations was that they require access to lots of system knowledge and that they are more expensive to develop than use case specifications.

**Resulting artifacts as means of communication.** All subjects considered the resulting models to be a good means of communication both with non-technical stakeholders such as customers, and between different engineering disciplines. FAR models were considered to help the different engineering disciplines to communicate on a common abstraction level. One issue pointed out was, however, that some engineers seem to want more details in the specifications.

Several engineers also had practical experience discussing FAR specification with customers. These experiences had been very positive except for one issue; customers sometimes have problems understanding the specifications because they contain too much organization specific terminology. It was also pointed out that the specifications should be supplemented with a MMI prototype to work really well when communicating with customers.

**Avoiding multiple implementations of common sub-functions.** A majority of the systems engineers, all software engineers and a majority of the electrical engineers considered the FAR approach to provide stronger support for finding common sub-functions and thereby avoiding multiple implementations, than the previous document based approach. *Systems engineering* also considered the approach to be better than structured analysis in this area, because of its end-user goal focus. *Software engineering* considered the FAR approach to provide the architect with exactly the tools needed to catch such sub-functions. *Electrical engineering* considered the FAR approach to be better in this area since it provides a better overview of such functions.

**Reuse of existing solutions.** All subjects in the study agreed that FAR provides stronger means for achieving high levels of reuse, than the previous approach. *Systems engineering* pointed out that using this methodology improves the possibilities to sell capabilities that can be implemented by existing solutions rather than selling new solutions. *Software engineering* stated that this area is probably the strongest feature of the FAR approach. The information in the specifications is captured in a form that is easily reusable in the next project, and it thereby encourages reuse also of the solutions which resulted from those specifications in the first place. *Electrical engineering* considered the resulting use case specifications to be almost platform independent information that can easily be transferred to new projects. This fact combined with the strong support for traceability makes the FAR approach very good for reuse.

## CONCLUSIONS

Triangulating the collected study data leads us to reject the study null hypothesis. The FAR approach performs better than managing the functional view according to the company process baseline in the specified industrial context. All subjects participating in the study consider the FAR approach superior to the previously used approach. We believe that many of the problems experienced during the case study were related to the fact that it was applied in pilot projects. This opinion was also supported by several subjects during interviews. We also conclude that all artifacts prescribed by the FAR approach provide value and are worth the effort developing. However, we do believe that clearer guidelines are needed on how to tailor the FAR process to fit different stages of the system life cycle.

## FUTURE WORK

Due to the fact that the study only included subjects from systems engineering, software engineering and electrical engineering, a wider follow-up study capturing also the views of other important groups of stakeholders, such as mechanical engineering, the ILS team and the verification team would be desirable. From this perspective it would also be interesting to perform formal experiments to compare FAR models to traditional structured analysis artifacts.

From a more practical point of view, the next important step of our work is to mine best practice from these pilot projects. This will help us developing guidelines and patterns for describing the different types of functions considered problematic by the subjects. Based on the study results, it is likely that these guidelines in some cases will recommend that the textual notations used in FAR should be supplemented with (graphical) state machines to resolve ambiguities and improve accuracy.

Another issue noticed was the problem of identifying good use cases. The study indicates that more methodological support is needed in this area. One interesting approach to address this issue would be to investigate if Cognitive Task Analysis (CTA) (Schraagen, 2000) could provide such support. CTA, which is a goal-oriented approach used in human factors engineering, might provide a good basis for goal-oriented use case modeling. Utilizing CTA as a basis for FAR use case modeling would also have the positive side-effect of integrating the human factors team into the early systems engineering effort.

## ACKNOWLEDGEMENTS

The authors would like to thank all people at BAE Systems Hägglunds AB that contributed to this work.

## REFERENCES

Adolph S., Bramble P., Cockburn A., Pols A., *Patterns for Effective Use Cases*, Addison-Wesley, 2003

Eriksson M., Borg K., Börstler J., *The FAR Approach – Functional Analysis/Allocation and Requirements Flowdown Using Use Case Realizations*, Proceedings of the 16'th International INCOSE Symposium, 2006

Kitchenham B., Pickard L., Pfleeger S., *Case Studies for Method and Tool Evaluation*, IEEE Software, Vol. 12 Nr. 45, 1995

Kruchten P., *The Rational Unified Process - An Introduction*, Second Edition, Addison-Wesley, 2000

Mayo E., *The human problems of an industrial civilization*, New York: MacMillan, 1933

Seaman C., *Qualitative Methods in Empirical Studies of Software Engineering*, IEEE Transactions on Software Engineering, July/August 1999

Schraagen J., Chipman S., Shalin V., *Cognitive Task Analysis*, Lawrence Earlbaum, London 2000

## BIOGRAPHY

**Magnus Eriksson** is a systems engineer with BAE Systems Hägglunds AB in Örnsköldsvik, Sweden. Magnus has, since he joined Hägglunds, been a key contributor in areas such as requirements engineering, system architecture and process improvement. He is currently also active as PhD student at Umeå University in Sweden, from where he also earned his Master of Science degree in engineering. His main research interests are in the areas software product line development and systems engineering.

**Jürgen Börstler** is an associate professor and director of studies in computing science at Umeå University, Umeå, Sweden. He received a Masters degree in Computer Science, with Economics as a subsidiary subject, from Saarland University, Saarbrücken, Germany and a PhD in Computer Science from Aachen University of Technology, Aachen, Germany. His main research interests are in the areas object-oriented software development, requirements management, process improvement, and educational issues in computer science.

**Kjell Borg** is currently software quality manager at BAE Systems Hägglunds in Örnsköldsvik, Sweden, and has over 20 years of experience in software engineering and research. He holds a BSc in Mathematics and a PhLic in Computer Science from Umeå University in Sweden. Mr Borg has experience in the fields of Usability Design, Requirements Engineering, Software Engineering, Quality Management, Embedded Systems, Project Management and Systems Engineering. He has business experience, both as an employee and consultant, in areas of Telecom, Defence, Transport, Process Industry, etc.