Applying the Situated Cognitive Engineering Method - A Comprehensive Guide

Exemplified with a case study from the domain of ambient assisted living

1. Introduction

At Delft University of Technology and TNO (the Netherlands institute for applied research), researchers have been working on a design methodology for complex, intelligent, and interactive technology. This methodology is the "situated Cognitive Engineering" (sCE) method. The sCE method has been previously applied in multiple projects and a wide variety of application domains. Some examples are: a task support system in naval ships (Neerincx & Lindenberg, 2008); a system that teaches social conventions (Schouten et al., 2013); a support system for astronauts during long-term missions (Neerincx et al., 2006; Neerincx et al., 2008); a support system for human-robot team performance in complex tasks (Mioch et al., 2012); a support system for urban search and rescue missions (De Greef et al., 2009); a support system for technical crews on board of naval vessels (Grootjen et al., 2009); the development of error-predicting cognitive models in the domain of aviation (Van Diggelen et al., 2011); and a support system for elderly living at home (Spiekman et al., 2011). A complete overview can be found in Neerincx (2011).

In the rest of this document, we will explain the sCE method in further detail (Section 3). The method is exemplified with the use of a running example in the domain of ambient assisted living, specifically fall detection. The application domain is briefly explained in Section 2.

2. Application domain

This document explains the use of the sCE method with the use of a running example taken from the SALIG++ project. In the SALIG++ project, researchers and developers collaborate on the design and development of an intelligent ambient living system in the homes of elderly people. SALIG++ consists of various types of sensors that together make an assessment of the elderly's situation in the home and its surroundings. Based on this assessment, SALIG++ determines whether or not assistance is appropriate, and, if so, what form of assistance should be provided: reminders, tips, video calls, alarms, or (emergency) calls requesting for human assistance, for instance, from formal and/or informal caregivers.

SALIG++ is a complex technological system that aims to support elderly people and their informal caregivers in realizing their goals and values with respect to elderly care. More specifically, the SALIG++ project aims to design technology that supports informal caregivers in their care tasks, supports elderly to be as independent as possible, and supports the health care system in maintaining and/or improving the elderly's health. In addition, SALIG++ aims to make sure that the technology is in line with, and supports, the values of everyone involved, such as: health, dignity, privacy, responsibility, security, and peace of mind.

To enable scientific research on the design of complex technology such as SALIG++, a design method is needed that allows for systematic evaluations of the design. Such evaluations should not only aim to (1) verify that the technology is performing in a technically reliable manner, but also (2) evaluate whether the technology successfully achieves its objectives with respect to the common goal in an effective and efficient

manner, (3) make sure that this technology is easy and intuitive to use, and (4) evaluate technology with regard to its achievements in supporting relevant moral values.

3. The situated Cognitive Engineering Method

The situated Cognitive Engineering (sCE) method structures and guides the systematic evaluation and documentation of technological designs that aim to provide social, cognitive and affective support to help users in realizing their goals and values. The method combines different design approaches and techniques in order to establish a sound, theoretically and empirically grounded design solution.

sCE supports the sharing and reusing of design knowledge by a heterogeneous, multi-disciplinary development community. Key is the generation, refinement, validation, maintenance, and reuse of coherent and concise design specifications. Such design specifications describe both what the technology should do and the underlying design rationale (the why and when). Three main segments are distinguished: (A) Foundation, (B) Specification, and (C) Evaluation. As can be seen from Table 1, each of these segments contains a small set of obligatory constituents that must be specified.

Table 1: Application of the sCE method results in design documentation consisting of three parts

Segment	Section	Obligatory constituents
Foundation	Section 2.1	(1) operational demands, (2) human factors, (3) technology
Specification	Section 2.2	(1) scenario, (2) use case, (3) requirement, (4) claim, (5) ontology
Evaluation	Section 2.3	(1) artefact, (2) evaluation method, (3) evaluation results

In contrast to a “waterfall procedure”, sCE allows for quick, incremental, and iterative generate-and-test cycles (e.g. agile development). In this process, researchers are required to continuously specify, refine, and integrate different parts of the system (i.e. components).

3.1 Foundation

The foundation segment in the sCE methodology describes the design rationale in terms of (a) operational demands, (b) relevant human factors knowledge, and (c) envisioned technologies. Together, these three constituents describe (1) the problem to be solved, (2) the existing knowledge on ways to solve the problem, and (3) the technology needed to implement that solution.

3.1.1 Operational Demands

The *operational demands* describe the current practice as it is, i.e. without the envisioned technology. For the operational demands, the sCE method prescribes as main components the stakeholders and their characteristics, and the problem description and analysis thereof. In the rest of this section, an example from the SALIG++ project is used to illustrate the sCE method, i.e. the scenario described in Use Case 7 “Fall Detection” as can be found in Deliverable D1.1 from the SALIG++ project.

The problem at hand is fall detection (as a result of tripping or loss of balance). The target group are elderly people who still live independently in their own homes and (may) receive informal care from family members and/or close friends. The problem is described in a *problem scenario*. A problem scenario provides a short storyline to

illustrate the way in which stakeholders experience the problem. In the example, the problem scenario provides a short story about Andrew, who is afraid of falling while walking his dog.

Problem description – "Andrew is afraid of falling as a result of tripping or losing balance"

"Andrew is 72 years, lives alone in a two-bedroom apartment in Stockholm. He is divorced. His sons Robert and Robin live in the same city. Andrew has worked as a truck driver. He has a dog that he needs to walk. He likes to go fishing, watch football and meet his friends at the local pub.

Last winter, Andrew was struck by a car while walking his dog. He suffered a severe head injury and was admitted to the emergency hospital. After the acute care he was admitted to inpatient brain injury rehabilitation for a three months rehabilitation program. He has problems with balance and fatigue, but is independent in self-care and household activities. The physiotherapist has prescribed a stick and a walker. Andrew has fallen inside and outdoors and for that reason he is afraid to walk the dog. The risk of falling increases when Andrew gets tired, inside the apartment as well as outside. Andrew has used his mobile phone to get in contact with his sons every time he has fallen. He is afraid that he will lose consciousness if he falls and thus will not be able to make a call."

In the SALIG++ project, several ***stakeholders*** can be identified, such as the elderly caretaker (Andrew) and the family caregivers (Andrew's sons). These are both direct stakeholders: they will be the main users of SALIG++. Indirect stakeholders are for example professional caregivers, insurance companies, and society in general. Indirect stakeholders may not interact with SALIG++ as much, but may still benefit from the availability of SALIG++ for the direct stakeholders.

In the example, we focus on the direct stakeholders, i.e. the elderly caretaker (Andrew) and the family caregivers (his sons). These stakeholders have ***values***. For instance: the elderly caretaker value his health, well-being, privacy, and independence, whereas the family caregivers value their freedom, peace of mind, and family values. The sCE method aims to take such values into account, such that they are supported in the design.

To obtain a clear overview of the problem, it needs to be analysed: what are the causes of the problem? Can the problem be broken down into smaller problems? And what type of problems are related to this problem yet beyond the scope of the current problem description?

Running example - Problem analysis

"Andrew is afraid of falling caused by tripping or loss of balance"

Stakeholders

- "Andrew" represents elderly people who still live independently in their own homes, and are still able to go outside, yet as a result of their age have decreased motoric capabilities (e.g. decreased strength, agility, stability, and accuracy in their movements). *Values*: independence, autonomy, animal care, health, safety
- "Andrew's sons" represent family members and close friends who play an important role in the social life of the elderly as well as in the provision of informal health care and/or safety and security of the elderly. *Values*: freedom, peace of mind, family care

Other possible actors

- Bystanders
- Neighbours
- The dog
- Possible attackers (e.g. robbers)

- Emergency services

Information extraction from problem description

- Afraid of falling, Afraid of being unconscious, Afraid of not being able to call sons
- Risk of falling outside, Risk of falling inside
- Risk of falling increased as result of fatigue and problems with balance
- Possible consequences of falling are, e.g., ending up in the hospital, lying on the street while being unconscious, being vulnerable while lying on the street
- Walks with stick or walker to improve stability (and decrease risk of falling)
- In case of falling, help might be required
- In case that help is required, the elderly may not always be capable of acquiring help him/herself
- The elderly wants to be able to trust that he/she is safe in case of a fall
- The elderly wants to be able to trust that he/she is taken care of after a fall
- The elderly wants to be able to trust that his/her family/friends are informed in case of a fall
- The elderly wants to be able to leave the house
- The elderly wants to be independent and autonomous in e.g. walking the dog
- The solution to the problem must be feasible, reliable, and affordable

Goals

- Prevent elderly from laying on the street for a long time
- Arrange for help as soon as possible (e.g. informal caregivers, bystanders, emergency services)
- Reassure the elderly/ calm the elderly down, keep elderly up to date
- Reassure informal caregivers, keep them up to date

Problem breakdown

- Detect whether elderly has fallen
- Detect whether elderly needs help from SALIG++
- Call for help when help is needed

Possible situations in which the elderly falls

1. Outside:
   a. The elderly falls and does not need SALIG++ to call for help
   b. The elderly falls and does need SALIG++ to call for help
2. Inside:
   a. The elderly falls and does not need SALIG++ to call for help
   b. The elderly falls and does need SALIG++ to call for help

Problems that might also be addressed by SALIG++ yet are beyond the scope of this problem scenario

- The elderly falls as a result of a sudden health problem (e.g. heart failure)
- Preventing the elderly from falling due to tripping or loss of balance
- Providing the elderly with First Aid care in case of an emergency

Solutions that can already be discarded as infeasible based on this description include:

- Setting up a network of people that could offer First Aid in the street and can be reached through a mobile app
- A service that walks the dog for Andrew so he no longer needs to go outside
- A service that keeps Andrew company when walking his dog, so he no longer needs to go outside alone
- An exoskeleton, inflatable airbag suit, a cottonwool suit, etc
- Training the dog to get help

3.1.2. Human Factors

When designing technology, there are two driving questions that need to be well-thought out: (1) What tasks and/or values is the user trying to accomplish and how can the technology support the user in doing so?, and (2) How can the technology be designed such that the user is able to work with the technology?

The *Human Factors* segment of the sCE method describes the available relevant knowledge about, for instance, human cognition, performance, task support, learning, human-machine interaction, ergonomics, etc. Note that we emphasize that this knowledge should be relevant for the problem and its design solution: the knowledge described here should lead to a better understanding of either (1) or (2).

The three constituents important to the human factors analysis are: (a) the human factors knowledge, (b) measures, and (c) interaction design patterns.

Human factors knowledge describes available knowledge coming from previous research about how to solve the problems that have been identified in the problem analysis.

Running example - Relevant human factors knowledge on behaviour in an emergency situation

- how to calm people down when they are in a state of panic
- how to overcome the bystander effect (i.e. the effect that in case of an emergency where there is a large group of bystanders often no one takes charge of the situation due to diffusion of responsibility)
- whether it is more effective to directly call the emergency services or that it's better to call informal caregivers so they can call the emergency services
- whether informal caregivers are able to memorize important information when receiving a phone call about a dear one being in an emergency situation
- whether there is a difference in being reassured by an automated system (such as SALIG++) or by family members

Measures describe how to operationalise the quality of the intended behaviour or performance, i.e. how well is a user working with the design able to reach his/her objectives and what is the quality of the collaboration between the human worker and the technology?

Running example - Relevant measures

In the running example, the following measures may help to determine the quality of the design solution:

- the extent to which SALIG++ is correctly used by the elderly (e.g. brought along, worn, switched on, recharged, etc.)
- the extent to which SALIG++ accurately assesses the "falling down" situation (e.g. false positives, false negatives)
- the extent to which SALIG++ effectively arranges for help in case of need (e.g. was it the right help, did help arrive soon)
- the extent to which SALIG++ effectively reassures the elderly while waiting for help (e.g. does the elderly feel worried, anxious, at ease, etc)
- the extent to which SALIG++ effectively reassures the informal and informs them of the situation (e.g. do the informal caregivers feel worried, informed, etc)
- the extent to which the elderly trusts SALIG++ to arrange for help in case of need (e.g. assess through a questionnaire)
- the extent to which the elderly is afraid of falling down (e.g. assess through a questionnaire)
- the extent to which the elderly is satisfied with the system (e.g. assess through a questionnaire)

As can be seen from these examples, it may not always be obvious how to operationalise the intended outcomes of a technology-based solution. For instance, when looking at improved quality of life or improved health, it can be difficult to find measures that quantify such concepts.

Interaction design patterns (IDPs) focus on the human-computer interaction, such as usable interface design and control options. IDPs offer generic solutions to recurring HCI design problems that have been proven to be effective. IDPs are often made available in repositories.

Running example - Useful IDPs

In the running example, the following IDPs could be of interest to document for reuse in future designs.

- solutions to the general interfacing of the technology such as haptic, visual, or auditory alarms
- more specific interactions such as the automatic repetition of an alarm signal until subsequent action has been registered

3.1.3. Envisioned Technology

The *envisioned technology* describes the available options of using existing technology and/or the need to develop novel technology in order to come to a system solution. The sCE method asks designers to specify what devices (hardware) and software could be used in the system design. In addition, for each type of technology, an argument should be provided as to why this technology might be of use and what the possible downsides might be of that particular type of technology.

Running example - Envisioned technology

Possibilities for envisioned technology in the running example could be:

- wearable technology for sensors (i.e. accelerometer, heart rate), sim card, and mini-computer
  Expected benefits:
  - not so easily forgotten, can be used inside and outside
  - able to perform physiological measures
  - able to recognize fall through accelerometer
  - able to perform simple pattern recognition
    able to contact mobile phone network and send automated message

  Possible downsides:

  - might be perceived as stigmatizing
  - need to acquire new technology
- speaker and microphone (in walking stick or walker for outside emergency, in each room of the house for inside emergency)
  Expected benefits:
  - not so easily forgotten when placed on walking stick/walker
  - intuitive communication (spoken language)

  Possible downsides:

  - need to build into stick/walker
  - may not be useable for people with auditory handicaps
- a server-based solution for data logging
  Expected benefits:
  - only small wearable micro-computer required to do local pattern recognition

  Possible downsides:

  - in case of network failure data may be lost

- the regular mobile phone network to support communication

  Expected benefits:

  - current quality and coverage of the mobile phone network is pretty good

  Possible downsides:

  - not 100% reliable
- family caregivers' smart phones to reach them in case of an emergency
  Expected benefits:
  - no need to purchase new technology

  Possible downsides:

  - alarm messages may not receive the right amount of attention because of multi-functionality of the device

3.2. System design specification

The *system design specification* describes the solution to the problem in the form of a system design that makes use of the identified relevant human factors knowledge and the envisioned technology. The system specification consists of (a) design scenarios, (b) actors and use cases, (c) requirements, (d) claims, and (e) ontology.

3.2.1. Design scenarios

The sCE method prescribes the specification of *design scenarios*. Design scenarios are short stories that provide a clear description of how the user will work with the technology thereby enjoying the solution offered to one of the problem scenarios. Together, the problem and design scenarios provide a contextualized view on:

1. the problem the design aims to solve
2. the people that are currently affected by this problem
3. the way in which the current system design aims to solve this problem and
4. how people will use the system.

Below, we take another look at the scenario previously described in Deliverable D1.1 from the SALIG++ project. This is a scenario describing how SALIG++ may help the elderly caretaker in case he or she falls down.

Running example - Design scenario

"SALIG++ helps Andrew overcome his fear of falling"

"Andrew is no longer afraid of falling down and not being able to contact his sons. Since a couple of months, he is using SALIG++. It took a little bit of getting used to wearing the bracelets and ankle band all the time. But now that he's experienced how well SALIG++ responds to his falls, he cherishes the system and takes great care of recharging the battery and wearing the sensors at all times. Together with his sons and doctor he was able to tweak the system's response to his falls to his personal preferences. In case he has fallen down and is merely shaken from the scare, the system contacts his sons who will arrange for someone to come and see him shortly after. In case he is injured, SALIG++ calls the emergency services and notifies his sons, so they are aware of what is happening."

3.2.2. Use cases

Scenarios are used to create more specific descriptions of step-by-step interactions between the technology and its users (i.e. *use cases*). As can be seen from the example use case, use cases include actors as to specify which stakeholders/agents are

interacting with each other in a given action sequence (use case). In the example below, the actors involved in the interaction are the elderly caretaker and SALIG++.

NB: Use cases do not specify the way in which the technology enables the described interactions. For example, in the use case presented below, the interactions may take place through voice commands and audio, but could also be text-based, be instantiated with the use a drop-down menu, or even by a human operator sitting on the other side of the application, listening to the elderly and responding to the events taking place. No assumptions are made about the level of automation or the current capabilities of the technology in mind. A use case simply describes the behaviour of the system, regardless of the technology required to produce that behaviour. Because the sCE method describes an iterative process of specifying the system's design, at later stages some behaviours may prove to be infeasible or not viable. This may result in alternative use cases that may be better aligned with the available technology. But it may also be the case that one ends up with a slightly different version of the technology that in fact is able to produce the ideal behaviour described in the original use case. The main goal of iteratively refining the system specification is to gather all the alternative design solutions, compare them in systematic evaluations, and converge to one design solution that is effective, reliable, affordable, etc.

Running example - Use case

"Elderly caretaker has fallen down while outside"

Actors: SALIG++, elderly caretaker

Circumstance: Elderly is away from home and has fallen down

Precondition: Elderly has brought the walker and is wearing a charged wrist band and ankle band

Post condition: The elderly is safe and receives proper help and care

Method: Checks through user interaction, communication, alarm, vocal reassurance, inform

Steps:

1. SALIG++ asks the elderly: "I think I registered a fall. Did you fall down?"
2. The elderly says: "Yes, I have!"
3. SALIG++ asks the elderly: "I am concerned. Did you hurt yourself?"
4. The elderly responds: "No, I am merely a bit shaken from the scare."
5. SALIG++ asks: "Do you want me to call your sons to come and see you?"
6. The elderly responds: "Yes, please. How kind of you."
7. SALIG++ says: "Your sons have been informed. They have arranged for Sonia (Robin's wife) to come and see you right away. Please stay put. I have sent them your location."

Alternative steps:

1. SALIG++ asks the elderly: "I think I registered a fall. Did you fall down?"
2. The elderly says: "Yes, I have!"
3. SALIG++ asks the elderly: "I am concerned. Did you hurt yourself?"
4. The elderly responds: "No, I am merely a bit shaken from the scare."
5. SALIG++ asks: "Do you want me to call your sons to come and see you?"
6. The elderly responds: "Yes, please. How kind of you."
7. SALIG++ says: "Your sons have been informed. They are not able to come see you straight away, but they will come to your house as soon as they can. Do you need someone else to come and get you?"
8. The elderly responds: "Could you contact my neighbour, Clara?"
9. SALIG++ says: "Clara has been informed. She will come to you right away. I have sent her your location."

Alternative steps:

1. SALIG++ asks the elderly: "I think I registered a fall. Did you fall down?"

2. The elderly says: “No, I haven’t!”
3. SALIG++ asks the elderly: “Ah, that’s a relief. Well, if you need me, I’m here.”

Alternative steps:

1. SALIG++ asks the elderly: “I think I registered a fall. Did you fall down?”
2. The elderly says: “Yes, I have!”
3. SALIG++ asks the elderly: “I am concerned. Did you hurt yourself?”
4. The elderly responds: “Yes, I think I have.”
5. SALIG++ asks: “Do you want me to call emergency services?”
6. The elderly responds: “Yes, please. And please notify my sons!”
7. SALIG++ says: “Your sons have been informed. The emergency services are on their way. Please stay put. I have sent them your location.”

Use cases can branch into new use cases. For instance, the use case described above could branch into a different use case where SALIG++ is also able to reach out to bystanders in an attempt to activate them and stay with the elderly until help has arrived. Together, the use cases provide a detailed description of the interactions between the technology and its user. Use cases make the design scenario more concrete by describing exactly how the technology makes sure that the elderly is safe and taken care of. Use cases are informed by human factors theories (described in the system’s foundation).

Running example - A note on personalisation

Sometimes, what happens in the use cases is different from person to person, because interactions are customized through personalisation. For instance, in the case of the “falling down” scenario, there may be different options on how to respond to the fall, such as:

- Contact emergency services with an automated message
- Contact a specific informal caregiver with an automated message
- Contact a specific informal caregiver and allow the elderly to speak to them him/herself
- Contact multiple informal caregivers one by one in case they don’t respond by using a predefined priority ranking
- Communicate with the elderly

A different use case should describe this personalisation process, i.e. how the elderly will specify what should happen in case of an emergency. This could for instance be done together with his/her (in)formal caregivers when the system is prepared for the first time.

3.2.3. Functional requirements

In the sCE method, use cases are used to derive ***functional requirements***, i.e. specific functionalities the technology should provide to its user.

Running example - Functional requirement

In the running example, SALIG++ should be able to:

- Help the elderly in case of a fall
- Comfort the elderly in case of a fall
- Inform the elderly’s informal caregivers in case of a fall

3.2.4. Claims

The sCE method prescribes a strong link between the system’s functional requirements, the system’s objectives, and the hypotheses to be tested during system evaluations. This

is accomplished by annotating all functional requirements with their underlying objectives (called *claims*).

Running example - Claim

An example of a functional requirement linked to the running example might be:

- `The system shall *keep the elderly up to date about the situation*'.

A possible underlying claim is that

- 'This functionality will *increase the elderly's feeling of reassurance'*.

This explicit linking of requirements to claims enables designers to formulate hypotheses that need to be tested in system evaluations to justify the adoption of the functional requirement: `By explaining to the elderly what to expect, the elderly feels significantly more reassured compared to when such information is not provided.' (NB: hence the importance of specifying proper measures, see Section 2.1.2). If the claim cannot be proven to be valid through system evaluations, the designers need to refine their system design, for instance, by trying to improve the functionality, replacing the functionality with a different one, or dropping the functionality and the claim altogether (i.e. by deciding that the objective is not reachable at this point). Either way, there is no use of including a functionality that does not achieve its underlying claims.

3.2.5. Ontology

Lastly, the sCE method prescribes the construction of an ontology, i.e. a vocabulary describing a common language to be used throughout the system specification to avoid miscommunication, misunderstanding, and inconsistencies. Furthermore, the ontology can serve as the basis for the technology's data structure. By specifying important concepts in the ontology and also choosing to use only one word instead of various ambiguous synonyms, communication becomes clearer and misunderstandings can be reduced to a minimum. The terms specified in the ontology are consistently used throughout the entire project.

Running example - Ontology

In the running example, the following concepts would likely be specified in the ontology:

- **SALIG++** – a system consisting of sensors, communication devices, and people with the aim to support elderly people to live independently in their own homes for an extended period of time in a safe, comfortable and dignified way.
- **Elderly** – people at an age above 75, who still live independently in their own homes, and are still able to go outside, yet as a result of their age have decreased motoric capabilities.
- **Reassurance** – removing someone's doubts or fears
- **Wrist band** – an arm bracelet containing a sensor suite, micro-processor, and sim-card
- **Ankle band** – an ankle bracelet containing a sensor suite
- **Walker** – a frame that is designed to support someone who needs help walking
- **Bystander** – a person in the proximity of the elderly in an emergency situation outdoors
- **Informal caregiver** – family members and close friends who play an important role in the social life of the elderly as well as in the provision of informal health care and/or safety and security of the elderly

3.3. Design evaluation

The last part of the sCE method is the *design evaluation*. The design evaluation aims to test and validate the system's design, or to discriminate between multiple design options,

such that the current design can be improved upon in incremental development cycles. The sCE method describes three parts that are relevant with respect to the system evaluation: (1) the artefact, (2) the evaluation method, and (3) the evaluation results.

3.3.1. Artefact

The *artefact* is an implementation or prototype that incorporates a given set of requirements, interaction design patterns, and technological means. In the example this is the SALIG++ system based on the medication adherence problem, meaning that it currently only supports this problem scenario. If in a later stage more problem scenarios are added to the project, the technology will include more and more functionalities, etc.

Running example - Artefact

A simple version of the SALIG system for proof-of-concept purposes, consisting of:

- A walker with a walkie talkie device strapped on top of it
- A human operator using the other walkie talkie to respond to emergency situations
- A prototyped bracelet with accelerometer, microprocessor, and sim card that sends a text message to a human operator with the location of the elderly every time a fall is detected

3.3.2. Evaluation method

The *evaluation method* can take many forms, such as a human-in-the-loop study, a use-case-based simulation, or an expert review.

Running example - Evaluation method

In the running example, the following method might be used to evaluate the artefact:

- A group of 30 elderly people are participating in an exploratory field study
- Before the start of the study, the elderly have been assessed with regard to their motoric capabilities, their fear of falling, their fear of falling without anyone around to help them, and their expectations of the system
- The elderly people all use the system for 3 months when going outside for a walk.
- After three months, the elderly people are asked about their experiences with the system. How did they feel during these 3 months? Did the system live up to their expectations? Are they less fearful? Do they feel more independent? Do they trust the system?

A possible addition to this method is to add a second group of people who rely on a mobile phone and panic button for three months. This allows for a comparison between the SALIG++ system and current common practice among elderly.

3.3.3. Evaluation results

The *evaluation results* describe the outcomes of the test. Because of the iterative and rapid research cycles, the evaluation does not necessarily include all requirements/claims/use cases available in the system specification. Oftentimes the evaluation investigates a subset of the system specification. Therefore, it is often useful to also specify what claims were tested, with the use of what evaluation method, and what artefact was used during the evaluation (i.e. which requirements, technology, interaction design patterns were included in the artefact).

Running example - Evaluation results

In the running example, we have only specified one functionality to support one use case. Therefore, the evaluation results address all of the above. Looking at this running example, the following

(fictitious) evaluation results could be obtained:

- After using the SALIG++ system for three months, the elderly feel more assured, less dependent, and more autonomous than their counterparts in the 'old solution' condition (phone plus panic button).
- The users trusted the SALIG++ system more than their old solution
- The users would have liked to continue using the system.
- Compared to the old solution, users were less inclined to forget to bring the SALIG++ system when leaving the house.
- 1 out of 10 emergency text messages were false alarms.

4. Literature

Brinkman W.P. and Neerincx, M.A. (2012). Applying situated cognitive engineering to mental health computing. CHI2012 workshop: Interaction Design and Emotional Wellbeing.

Brinkman, W.-P., Vermetten, E., van den Steen, M., and Neerincx, M.A. (2011). Cognitive engineering of a military multi-modal memory restructuring system. Journal of CyberTherapy & Rehabilitation. Vol 4(1), pp. 83-99.

De Greef, T., Oomes, A. H. J. & Neerincx, M. A., "Distilling support opportunities to improve urban search and rescue missions", in: Human-Computer Interaction - Interacting in Various Application Domains, Springer, 2009, pp. 703–712.

De Greef, T.E., Mohabir, A., Van der Poel, I. and Neerincx, M.A. (2013). sCEthics: Embedding Ethical Values in Cognitive Engineering. Proceedings of the 31st. European Conference on Cognitive Ergonomics, pp. 1-8. New York, USA: ACM.

Grootjen, M., Neerincx, M.A., de Vries, N. Th., and Badon Ghijben, N.A. (2009). Applying situated cognitive engineering for platform automation in the 21st century. 14th Ship Control Systems Symposium, September 20-23, 2009 Ottawa, Canada.

Laverman M., Neerincx, M.A., Alpay L.L., Rovekamp T.A.J.M. & Schonk B.J.H.M. (2014). How to Develop Personalized eHealth for Behavioural Change: Method & Example. TNO report R10758. Delft: TNO.

Mioch, T., Smets, N. J. J. M. & Neerincx, M. A., "Predicting performance and situation awareness of robot operators in complex situations by unit task tests", in: Conference on Advances in Computer-Human Interaction, 2012, pp. 241–246.

Mioch, T., Ledegang, W. Paulissen, R. Neerincx, M.A., and Van Diggelen, J. (2014). Interaction Design Patterns for Coherent and Re-usable Shape Specifications of Human-Robot Collaboration. EICS 2014 (Rome, Italy, June 17-20), pp 75-83, ACM.

Neerincx, M.A. & Lindenberg, J. (2008). Situated cognitive engineering for complex task environments. In: Schraagen, J.M.C., Militello, L., Ormerod, T., & Lipshitz, R. (Eds). Naturalistic Decision Making and Macrocognition (pp. 373-390). Aldershot, UK: Ashgate Publishing Limited.

Neerincx, M.A., Te Brake, G.M., Van de Ven, J.G.M., Arciszewski, H.F.R., De Greef, T.E., and Lindenberg, J. (2008). Situated cognitive engineering: Developing adaptive track handling support for naval command and control centers. In: B. Patrick, C. Gilles and L. Philippe (Eds.). HCP-2008 - Third Internation Conference on Human-Centered Processes, pp. 3-20. Bretagne, France: TELECOM Bretagne.

Neerincx, M. A., Lindenberg, J, Smets, N, Grant, T, Bos, A, Olmedo-Soler, A, Brauer, U & Wolff, M, "Cognitive engineering for long duration missions: human-machine

collaboration on the Moon and Mars", in: Conference on Space Mission Challenges for Information Technology, IEEE, 2006, pp. 7–14.

Neerincx, M.A. (2011). Situated Cognitive Engineering for Crew Support in Space. Personal and Ubiquitous Computing. Volume 15, Issue 5, pp. 445-456.

Peeters, M., Van Den Bosch, K., Meyer, J-J.Ch. and Neerincx, M.A. (2012). Situated cognitive engineering: the requirements and design of directed scenario-based training. The Fifth International Conference on Advances in Computer-Human Interactions (ACHI 2012, January 30 - February 4), pp. 266-272. Xpert Publishing Services (XPS).

Schouten, D., Smets, N., Driessen, M., Hanekamp, M., Cremers, A. H. & Neerincx, M. A., "User requirement analysis of social conventions learning applications for non-natives and low-literates", in: Engineering Psychology and Cognitive Ergonomics. Understanding Human Cognition, Springer, 2013, pp. 354–363.

Spiekman, M. E., Haazebroek, P. & Neerincx, M. A., "Requirements and platforms for social agents that alarm and support elderly living alone", in: Social Robotics, Springer, 2011, pp. 226–235.

Van Diggelen, J., Janssen, J., Mioch, T. & Neerincx, M., "Flexible design and implementation of cognitive models for predicting pilot errors in cockpit design", in: Human Modelling in Assisted Transportation, Springer, 2011, pp. 147–154.

Westera, M., Boschloo, J., van Diggelen, J., Koelewijn, L.S., Neerincx, M.A. and Smets, N.J.J.M. (2010). Employing use-cases for piecewise evaluation of requirements and claims. In: W.P. Brinkman and M.A. Neerincx (Eds.). Proceedings of ECCE2010. pp. 279-286. Delft University of Technology, The Netherlands.