Comparing Safety Analysis Based on Sequence Diagrams and Textual Use Cases

Tor Stålhane[1], Guttorm Sindre[1], and Lydie du Bousquet[2]

[1] Dept of Computer and Info. Science, Norwegian Univ. of Sci. and Tech (NTNU)
[2] Laboratoire d'Informatique de Grenoble (LIG)
{stalhane,guttors}@idi.ntnu.no, lydie.du-bousquet@imag.fr

Abstract. Safety is of growing importance for information systems due to increased integration with embedded systems. Discovering potential hazards as early as possible in the development is key to avoid costly redesign later. This implies that hazards should be identified based on the requirements, and it is then useful to compare various specification techniques to find out the strengths and weaknesses of each with respect to finding and documenting hazards. This paper reports on two experiments in hazards identification – one experiment based on textual use cases and one based on systems sequence diagrams. The comparison of the experimental results reveal that use cases are better for identifying hazards related to the operation of the system while system sequence diagrams are better for the identification of hazards related to the system itself. The combination of these two techniques is therefore likely to uncover more hazards than one technique alone.

Keywords: safety analysis, sequence diagrams, misuse cases, experiment.

1 Introduction

Systems safety is concerned with the avoidance or mitigation of harmful accidents, as opposed to security which addresses malicious attacks against the system. Safety has traditionally been more of a concern for embedded systems than for information systems. However, increasing convergence and integration between different kinds of systems, as well as trends towards ubiquitous information systems and "internet of things" [1] means that the border between information systems and embedded systems is getting more blurred. An increasing number of safety-critical systems are to a large degree realized in software. Thus, safety analysis of software gets more and more important. Safety analysis needs to include a wide variety of participants – not just the safety analysis specialists. Two important groups that always need to be involved are the customers with their domain knowledge and the developers who have a firm understanding of all the decisions needed to develop a software system.

As a consequence of this, the safety analysis must apply methods and documents that are easy to understand for everybody involved. In our opinion, both sequence diagrams and textual use cases belong to this category. It might, however, not be without consequences to choose one over the other. It might be the case that one of these methods is uniformly better than the other or they may each have their strong and weak sides. In any case, one or more experiments are needed in order to see how these two methods behave when used as the starting point for a safety analysis. In order to shed some light over this, we have run an experiment partly in Grenoble – system sequence diagrams – and partly at NTNU – textual use cases.

The rest of this paper is structured as follows: Section 2 goes through related work. Section 3 presents the two safety analysis techniques to be compared. Section 4 describes the experimental design and section 5 the analysis of the results. Section 6 discusses threats to validity, whereupon section 7 concludes the paper.

2 Related Work

As far as we have found out, there are few experiments comparing textual (mis)use cases and sequence diagrams. This is no surprise, since there are few controlled experiments about use cases to be found in the literature at all. There are experiments comparing the effectiveness of representation languages, for instance ER or EER diagrams with relational tuples [2]. However, the two forms of representation compared contained the same information but were on different life-cycle levels, ER being an analysis stage language, while relational tuples belong to the design stage. The relationship between use case diagrams and systems sequence diagrams is different: both are requirements stage representations but the textual use case and the systems sequence diagram have different focus. The textual use case focuses on the exchange of messages between system and user, while the systems sequence diagram also includes the exchange of messages between system components.

An experiment more directly targeting the comparison of diagrams and tables and textual representations is [3], confirming claims about advantages of diagrams previously made analytically in [4]. Experiments have also found diagrams superior for presenting procedural instructions [5], which is closely related to conceptual modeling. More generally, the empirical evidence is not so consistently in favor of diagrams. As reported in [6], some experimental studies have shown advantages for diagrams, but others have shown advantages for tables, or given mixed results. One particular study also showed the outcome to be dependent on the subject group, business students solving problems more effectively with tables while engineering students solved problems more effectively with graphs [7].

There are several papers published on the use of UML diagrams as a basis for safety analysis. A good example is [8], where Guiochet and Vilchis use the full range of UML diagrams – use case diagrams, sequence diagrams, class diagrams, state diagrams and deployment diagrams. In [9] Long and Jinglun use event sequence charts for safety analysis. C. Ren also uses several UML diagrams – among them the sequence diagram – for safety inspection of mine equipment [10]. Unfortunately, none of authors have performed experiments. All three papers, and others related to the same area, are case studies to see if the chosen diagrams can be used in a safety analysis. Similarly, Allenby and Kelly propose a technique for safety analysis based on scenarios [11].

A more comparative work is done by T.A. Alspaugh et al [12]. Their focus is unfortunately not safety problems but defects. What makes their paper interesting is that they compare the ease of identifying defects using use cases, sequence diagram and scenarios formulated in ScenarioML. Their conclusion is clear – scenarios are better than both sequence diagrams and use cases. Their experiment had only four participants and no statistically significant results were thus obtained. On the other hand, they ran a series of interviews with the participants and thus gained a lot of insight into the reasons behind the results. The reason why use cases did not work well was that "the numbering scheme for the steps can be confusing". For the sequenced diagrams, some of the participants had problems with the notation – especially branches and options. On the other hand, the participants had no problems with the ScenarioML documents which they found easy to read and understand.

In addition to the safety and reliability related experiments cited above, there has been a lot of research on other facets of use cases such as authoring – see for instance the well-known work of Achour et al [21] and the follow-up work of Cox et al. [22].

3 The Techniques to Be Compared

3.1 Textual Use Cases – TUC

TUC can be used to identify possible failure modes, using the textual misuse case analysis. The misuse case was originally proposed for collecting security requirements. They can, however, also be used for safety analysis where focus is on accidents causing harm to the system and its environment. A simple example is an automated system used to keep the water level in a tank constant while delivering steam to an industrial process. This is done by filling the tank through one valve and emptying it through another valve when needed. If the pressure in the tank exceeds the critical pressure pre-set by the operator, a relief valve should open automatically. The operator may also manually empty the tank (for instance if the relief valve fails to work when the pressure becomes too high) or manually fill the tank (if the automatic adjustment of water level does not work).

A misuse case such as "Set too high pressure" may have a "threatens" relationship to one or more use cases – in this case "Set critical pressure". It is also possible that one misuse case may "aggravate" the effect of another, or that a use case may have a "mitigates" relationship to that misuse case. An example of a textual representation is shown in Table 1. Here, threats corresponding to misuse cases can be added in a third column called "Threats", and mitigations can be added in a fourth column. An essential idea of misuse case analysis is that the representation format causes only a limited overhead if use cases are already applied in a project, which is often the case in mainstream software projects. In addition, the informality and simplicity of the technique makes it suitable for supporting brainstorming on threats at an early stage of development. For a more complete coverage of misuse case analysis, the reader is referred to [13] and [14].

Table 1. Column format use case with safety issues for the boiler

Use case name	**"Empty tank manually"**		
User actions	**System response**	**Threats**	**Mitigations**
	System alarms operator of high Pressure	System fails to raise alarm; Operator fails to notice alarm	2 independent alarms; Use both sound and blinking lights
Operator issues command to empty tank		Operator fails to react (e.g., incapacitated?) Operator gives wrong command, e.g., filling tank	Alarm backup operator; Auto sanity check, disallow filling at high Pressure
	System opens valve to sewer	System fails to relay command to valve; Valve is stuck	
Operator reads pressure		Operator misreads and stops tank emptying too soon	Maintain alarm blinking until situation normal
	Pressure returns to normal	This is not achieved, see exceptions	
Operator stops tank emptying and logs the event. This ends the use case.			
Exceptional paths			
	Opening valve is insufficient to normalize pressure		
Operator issues command to reduce temperature		Operator gives wrong command, e.g., increase temperature	Automatic sanity check, disallow temp increase at high pressure
	Pressure returns to normal		
Operator logs the event. This ends the use case.			

3.2 System Sequence Diagrams - SSD

The sequence diagram is an UML diagram used to show the dynamic properties of a system. A sequence diagram shows the messages that are passed between two components in a system. In order to be a strict sequence diagram, these two parts have to be objects – instances of classes. In the early phases of system development, however, the components used in the diagram can be e.g. GUI, database servers, internets or subsystems. Sequence diagrams applied in this way are called system sequence diagrams or SSD [15]. Even though they have no formal standing in the UML 2.0 definition they are widely used in the industry. One of the factors that make them both useful in an industrial context and important for our research on safety analysis is that the SSDs, just as the textual use cases, can be constructed early in the development process.

An SSD is close to a textual use case and can be constructed using the same structure and words as a textual use case. The SSD has, however, one advantage over the textual use cases. With the advent of UML 2.0 [16], sequence diagrams in general and thus also the SSDs can contain alternatives – if….then…else… (alt or opt) and loops – for… (loop). These constructs enable us to insert info into an SSD that it might be difficult to include in a textual use case. The diagram below contains the same info as the use case in table 1 except for the last two rows.

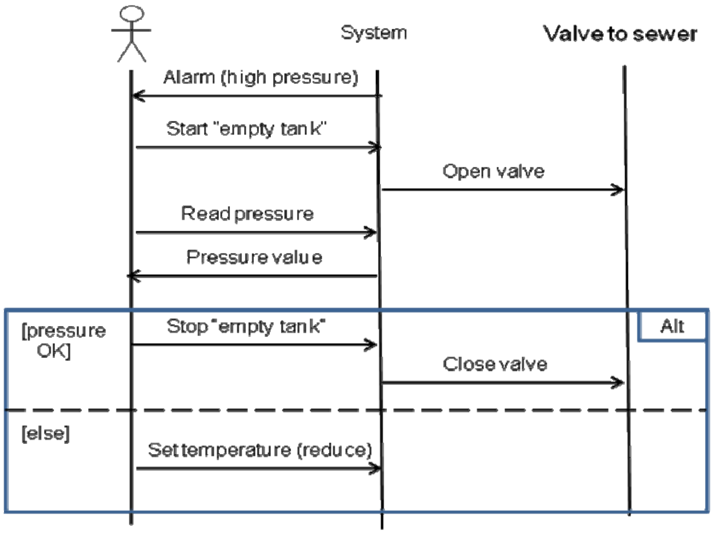

Fig. 1. SSD for the function shown in table 1

The SSD can be used for safety analysis by going through each message sent or received and ask questions like "what will happen if the message is not sent" and "what will happen if the message has the wrong content"?

4 The Experiment

4.1 Research Approach

Both our previous experiments have focused on the question "Is representation A better than representation B for the identification of failure modes?" In both cases it turned out that the answer was not a simple yes or no. Even though we could conclude that in general representation A was better than representation B, there were still areas where representation B was better and areas where they were equal. Thus, we will look at both whether textual use cases are better than sequence diagrams and, if this is the case, are there still areas where sequence diagrams are better? We will try to throw some light over this by answering the following research questions:

- RQ1: Is TUC in general better then SSD for identifying failure modes?
- RQ2: If we conclude in RQ1 that one of the representations in general is better than the other, are there still any problem areas where this does not hold?

- RQ3: If we conclude in RQ2 that in general representation A is better than representation B, then:
  - RQ3-1: Are there still failure modes where representation B is better than representation A?
  - RQ3-2: What can be the reasons for this?

The two first of these research questions can be reformulated as hypotheses. In order to test these hypotheses, we will use the t-test for RQ1 and proportion test for RQ2. The reasons for these choices are that for RQ1 we need to compare the number of failure modes identified while for RQ2 we need to compare the proportion of participants that have identified a single failure mode.

To use the proportion test, we define the following parameters:

- x_i is the number of successes observed in sample i.
- π_i is the portion of success events in sample i.
- n_i is the size of sample i.
- Δ is the difference that we want to test for. In our case Δ is set to zero.
- π is the portion of successes in both samples viewed together – see equation (1).

$$\pi = \frac{x_1 + x_2}{n_1 + n_2} \tag{1}$$

The proportion test is based on the following statistics, where z is assumed to be normally distributed (0, 1):

$$z = \frac{\pi_1 - \pi_2 - \Delta}{\sqrt{\pi(1-\pi)(\frac{1}{n_1} + \frac{1}{n_2})}} \tag{2}$$

RQ3-1 is answered by studying the results from the t-tests used to answer RQ2. However, when trying to answer RQ3-2 we will use the areas defined in our experiment coding as a starting point. RQ3-2 cannot be answered using any statistical approach since the focus is why – not what. Assuming that representation A in general is better than representation B, we will do as follows:

1. For each failure mode where representation B is better than representation A, write down the characteristics of the failure mode that can be related to both representations.
2. Identify what it is in representation B that makes it easy to discovery the failure mode and what it is in representation A that makes it difficult to discover.
3. If we are able to identify a convincing set of factors in the representations that respectively hide and reveal the failure mode under consideration, we will claim that the reason why the failure mode was found by the participants using representation B and not by the participants using representation A stem from these factors.

4.2 Experiment Design

4.2.1 The Preparation

The experiment using SSDs were done at the University J. Fourier - Grenoble I (UJF), France by the Laboratoire d'Informatique of Grenoble (LIG) and the experiments on

TUC were done in Trondheim at the department of Information and Computer Science at NTNU. Since the participants in the two sites had different background, the preparation phase for the two experiments differed while the rest of the experiment design was the same at both sites.

At UJF, the participants were in the first year of a Master in Computer Sciences, where courseware is offered in English[1]. Students were gathered in an auditorium, where they received a short introduction on safety issues and safety analysis. An example of safety analysis was illustrated. This part lasts 1 hour and 30 minutes. Then participants were asked to fill in the pre-experiment questionnaire, read the document and fill in the post-experiment questionnaire. They then had 1 hour and 30 minutes to do it but most of the participants left the room before the end of the period.…

At NTNU, the participants were gathered in an auditorium where they receive a short introduction to safety analysis and why it was important to be able to do this at an early stage. They then had 15 minutes to refresh their knowledge of textual use cases and misuse cases before filling in the pre-experiment questionnaire.

When the pre-experiment questionnaire was finished the participants worked on their task at identifying failure modes based on the textual use cases and a system diagram, showing the main components of the train control system and how the components – radio, operators, trains and track side equipment – cooperated in order to run the railway in a safe and efficient manner.

For both experiments, each participant documented the identified failure modes in free text on the sheet of paper where the TUC or SSD was shown.

4.2.2 The Coding Process

The researchers read the failure mode descriptions and related it to one of the failure modes used in the coding schema. If several failure modes could apply, the researchers reached a common understanding of the category that was more close to the participant's intentions. In order to reduce variation in the application of the coding schema, all the forms turned in by the experiment participants were coded by the same researcher and controlled by another. The coding schema was designed in advanced, based on an analysis of the failure mode categories used by the respondents in pervious experiments. The schema has three levels and the top level contains the following categories:

- OP – Operator problems
- CS – Computer system problems
- EP – Engineer problems
- MP – Maintenance personnel problems
- TC – Technical communication problems
- TE – Track-side equipment problems

Each of these categories has two or more subcategories. The subcategories also have a set of sub-subcategories, which is not shown here. This coding schema has 104 failure modes and gave us a flexible way to categorise the experiment results. The subcategories for operator problems – OP – are shown below:

[1] http://mosig.imag.fr/MainEn/HomePage

- OP100 Incoming messages
- OP200 Operator action
- OP300 Operator scheduling
- OP400 Operator equipment problems
- OP500 Operator knowledge
- OP600 Operator overload, e.g. due to panic
- OP700 Communication problems

The responses were registered in an Excel worksheet – each failure mode category at the lowest level has a line in the worksheet and each participant has a column. The entries for each participant are binary – found a certain failure mode or not. I.e. there is no "bonus" for identifying the same failure mode several times.

5 Experiment Results and Analysis

The best participant in the TUC experiment found 25 failure modes while the one that did worst found only 8. Thus, the identification probability ranges from 24% to 8%. For those who used the SSD, the best participant found 19 failure modes while the one who did worst found only 4. In percentages this gives us a range of 18% to less than 4%.

In order to test whether a difference between two representations is important, we need to look at the significance level α and the effect size ES. Even though the significance level – the p-values – are below 0.05, the difference may still be uninteresting if the effect size is low. Thus, we will base our discussions both on the significance level and the effect size. Let t be the value of the t statistics and df the number of degrees of freedom. We will use the following expression to calculate the effect size ES:

$$ES = \frac{2t}{\sqrt{df}} \tag{3}$$

RQ1 is answered using the t-test. The result is as follows:

Table 2. t-test for total number of failure modes identified

t-Test: Two-Sample Assuming Unequal Variances		
	TUC	*SSD*
Mean	15.07	11.70
Variance	10.92	24.46
Observations	29.00	10.00
Hypothesized Mean Difference	0.00	
Df	12.00	
t Stat	2.01	
P(T<=t) one-tail	0.03	
t Critical one-tail	1.78	
P(T<=t) two-tail	0.07	
t Critical two-tail	2.18	

Based on this, the answer to RQ1 is that the experiment participants using TUCs found significantly more failure modes that those that used SSDs. The effect size is 1.16 which indicates a large effect.

Of the 104 failure modes, 70 were related to human behaviour, 28 were related to the technical systems and six were related to operational conditions. If we look at each category separately we find that the difference is significant only for the human related failure modes with a p-value less than 0.01 and an effect size of 1.1. For the two other categories, we find no significant differences.

To answer RQ2, we first need to consider the differences in proportion of experiment participants that identify the failure modes for each main area.

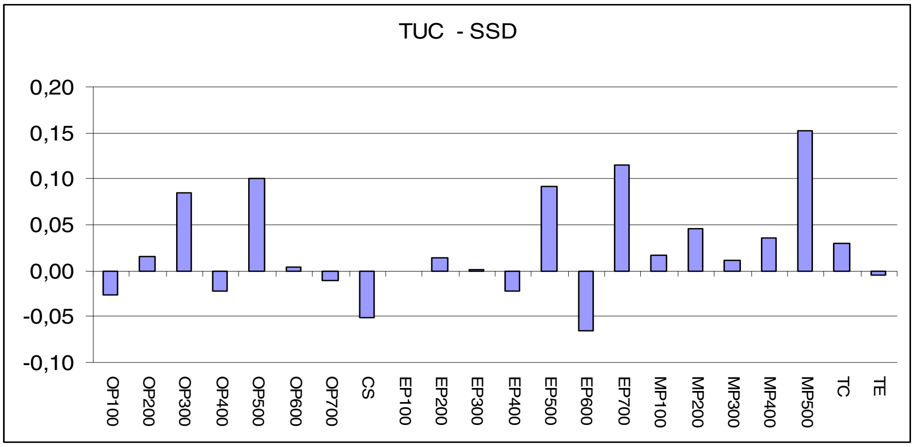

Fig. 2. Frequency differences between TUC and SSD

Our answer to RQ2 is that everywhere where there is a statistically significant difference between the areas, failure mode identification based on textual use cases is better or just as good as the one based on SSD with $\alpha = 0.10$. TUC is significantly better than SSD for the areas listed below:

Table 3. t-test for number of failure modes identified for each main area

Failure mode	p-value	ES
OP 300 Operator scheduling	0.02	0.88
OP 500 Operator knowledge	0.02	0.90
EP 500 Engineer knowledge	0.04	0.92
EP 700 Communication problems	0.02	1.15
MP 200 Maintenance action	0.01	1.01
MP 500 Maintenance personnel knowledge	0.00	1.22

We see that all the effect sizes are moderate to large. Note that a large part of the areas where TUC is significantly better than SSD are related to knowledge and actions – factors that relate to how personnel use the system.

In order to get a clearer picture of the experimental results related to RQ2, we need to look at the results for each single failure mode. These results are shown in table 4 below, while the plot of individual failure modes shows the more detailed picture shown in figure 3. We see that half of the frequency effect sizes are small – 0.10 and that roughly a third are moderate to large – 0.30 to 0.50.

Table 4. Proportion tests for each failure mode

Failure mode	p value	Δ_{freq}
OP 204 Wrong ack	0.067	0.10
OP 212 Do not save changes to schedule etc.	0.002	0.24
OP 302 Wrong train scheduling *	0.007	0.42
OP 501 Wrong situation analysis *	0.013	0.39
OP 503 Wrong interpretation of system's functionality	0.067	0.10
CS 100 Does not save or deletes info	0.006	-0.46
EP 209 Enters wrong info to operator	0.094	0.29
EP 504 Lack of training *	0.006	0.21
EP 700 Communication problems *	0.067	0.10
MP 209 Enters wrong info *	0.003	0.46
MP 212 Do not save changes to schedule etc. *	0.067	0.10
MP 401 Lack of person-to-person trust or personnel conflicts	0.067	0.10
MP 402 Problems with telephone or radio	0.067	0.10
MP 501 Wrong situation analysis *	0.000	0.49
MP 504 Lack of training *	0.031	0.14
TC 400 Other technical communication problems	0.067	0.10

In order to answer RQ3, we start with the table above. As we see, the only failure mode where SSD is significantly better than TUC is for CS100 – Does not save or delete info. Thus, the answer to RQ3-1 is that for CS100, SSD is doing significantly better than TUC. In addition we see that the frequency difference is 0.46, indicating a large effect.

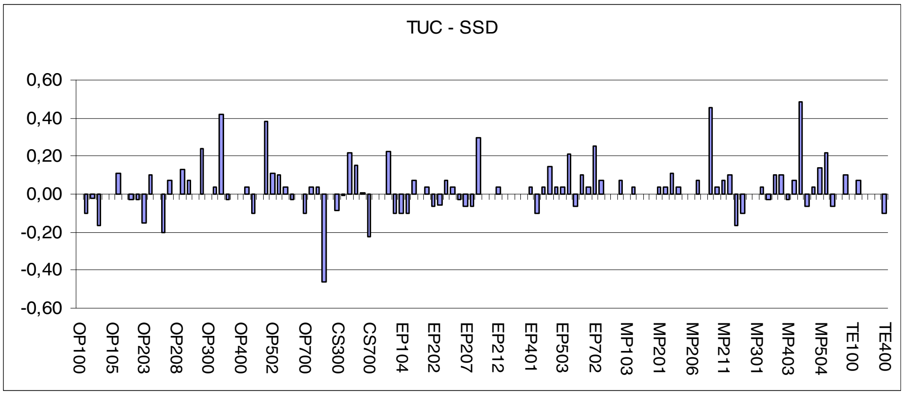

Fig. 3. Frequency differences for all failure modes

An alternative way to look at the results of the experiment is to consider all failure modes that have been identified by more than 50% of the participants in each group. This gives us the results shown in table 5.

Table 5. Top ten failure modes identified for each representation

	Percentage of participants		
	TUC	SSD	
OP209	82.76	80.00	CS400
CS400	79.31	70.00	OP209
EP209	79.31	60.00	CS100
EP102	72.41	60.00	CS700
EP702	65.52	50.00	OP203
MP209	65.52	50.00	CS300
OP302	62.07	50.00	EP102
OP501	58.62	50.00	EP209
MP501	58.62		

The table shows that for TUC there is only one computer system (CS) failure mode among the top 10 failure modes, while we have four CS failure modes for the SSD. These failure modes are:

CS100 Does not save or deletes info
CS300 Reacts wrongly to command or do not react at all
CS400 Shows wrong info, including false alarms – both TUC and SSD
CS700 Other software errors

For all of these failure modes SSD is better than TUC although the significance level and frequency differences are small, except for CS700 where we have a difference of -0.22. The p-value, however, is 0.22. We should also note that while there is 11% computer system related failure modes among the top ten found for TUC, the corresponding value for SSD is 50%. The same effect has been observed in previous experiments. We have found that FMEA outperform misuse case diagrams [17] when it comes to network problems and that misuse cases based on both diagrams and text score low on the failure modes “software unavailable”, “network down” and “delete files” [18]. Thus, as a general conclusion, we will claim that TUC is better than SSD for all failure modes except those pertaining to the inner working of the computer system.

We will use the process suggested for RQ3-2 in chapter 4.1 in order to understand why we get this result.

1. Characteristics of CS failure modes: The main characteristic of a CS failure mode is that it describes computer system failure modes. The user and his behaviour are not involved.

2. What in SSD helps us to discover CS failure modes and what in TUC hinders this discovery? The answer to this question is that SSD makes explicit the sending and receiving of messages – e.g. commands – between actors and one or more subsystems. The TUC, on the other hand has all its focus on the user and how the system reacts to his commands. Sending and receiving messages is only included indirectly.
3. Based on this, we will conclude that TUC helps the analyst to focus on user / system interactions, while SSD helps the user to focus on how the user / system interactions are performed – e.g. via sending and receiving messages.

6 Threats to Validity

We will use the categories defined in [19] as a starting point for our discussion on threats to validity. We will look at each threat in a short section before giving a sum-up of our validity claims.

6.1 Conclusion Validity

Conclusion validity is concerned with our ability to draw the right conclusions about the relationship between the treatment and the outcome. An important question here is sample size. For the main failure mode categories, we have seen a medium to large effect – ES values from 0.88 to 1.22.

If we denote the type I error probability by α and the type II error probability by β, and let N be the sample size, the following relationship holds:

$$N = \frac{4(u_{\alpha/2} + u_{\beta})^2}{ES^2} \tag{4}$$

If we use $\alpha = 0.05$ and $\beta = 0.20$, we get for smallest effects $N = 26/0.9^2$ which gives use an N-value of 32. Since we have a total of 39 participants, we have a sufficient number of observations for our conclusion.

For proportions, a large effect size is 0.40 or more. The only single failure mode which departs from the general picture – TUC better than SSD – is failure mode CS100. In this case the ES is large – 0.46. If we use the same notation as in equation (4) except that N now is the sample size of each part of the experiment, we have for proportions that:

$$N = (u_{\alpha/2} + u_{\beta})^2 \frac{p_1(1-p_1) + p_2(1-p_2)}{d^2} + \frac{2}{d} + 2 \tag{5}$$

For CS100 we have $p_1 = 0.60$ and $p_2 = 0.14$ and thus $d = 0.46$. If we, as before use $\alpha = 0.05$ and $\beta = 0.20$, we get $N = 19.7$ Thus, we need 40 observations in order to observe the difference for CS100. We have 39 observations, which is so close to 40 that we will accept the result as valid. We only have to increase the risks of a type II error from 0.20 to 0.21 in order to reduce the number of observations necessary to 39.

6.2 Internal validity

Internal validity is concerned with the relationship between treatment and outcome – was it the treatment that caused the outcome?

The main discussion point is whether the two groups of experiment participants are equal. In order to answer this question we will look at results from the pre-experiment questionnaire. Out of the eight questions in pre-experiment questionnaire, there are significant differences between the two groups in the answers to only three of them. The two groups score the same for knowledge of sequence diagrams, analysis of safety, reliability and misuse cases plus the number of months of practical experience with IT work.

The three questions where the two groups differ are questions PQ1, where the SSD group has less experience with class diagrams, PQ5, where the SSD group has less experience with writing use cases and PQ7, where the SSD group on the average has completed six semesters of their studies while the TUC has completed only 4.8. The important difference, however, is the difference between the two groups' knowledge of their respective methods – the SSD group's knowledge of sequence diagrams and the TUC group's knowledge of use cases. On a Likert scale from 1 (no knowledge) to 5 (good knowledge) the SSD group has an average score of 2.1 for sequence diagram experience while the TUC group has an average score of 1.6 for use cases. The difference is significant at the 0.05 level. The effect size, however, is just 0.5 – a small to moderate effect. As an aside, we know there are purists who do not like to use arithmetic on Likert scale values. We will not take up this discussion here but point the reader to John W. Tukey's article [20] on this topic.

Besides the factors mentioned above, cultural differences may have influenced the results. There is, however, no way that we can assess the size of this influence.

6.3 Construct Validity

Construct validity is concerned with the relationship between theory and observations. The theory is in our case about the relationship between the number of failure modes identified and the documents used in this process – TUC or SSD. Counting the number of failure modes enables us to observe any differences in the effect of the two ways to represent the situation – TUC or SSD. Since we have already shown that there are no or just small differences between the participants' capabilities, we will claim that the observed effect stems from the different representations.

6.4 External Validity

External validity is concerned with generalization – where and when are the conclusions applicable and can we generalize from our experiments to industrial practice? External validity is threatened in two ways: the experiment participants received little training in the new method and the quality of the result has no influence, neither on a real product nor on the participant's working situation. However, these threats will influence both methods in the same way. Since we are only looking for differences and not for any absolute measure this will not influence our conclusions on the relative merits of the two methods.

6.5 Our Claims to Validity

Based on the discussions above, and besides a possible caveat for cultural differences, we claim that there are no serious threats to validity for our conclusions on the number of identified failure modes.

7 Conclusion

Based on our data analysis we will conclude that TUC is better than SSD when it comes to identifying failure modes related to required functionality and operator behaviour. The systems sequence diagrams outperform textual use cases when it comes to failure modes pertaining to the system's internal working. Based on this, we recommend that, for hazard analysis in the early phases, we should provide a set of documents that enables the analysts to focus on each area under consideration.

A natural direction for further work would be to do more experiments, for instance using industry practitioners in addition to students. It would also be interesting to compare various techniques in bigger case studies, since controlled experiments necessarily limit the size of the tasks that can be performed, thus lacking the realism and complexity of information systems development projects.

References

1. Gershenfeld, N., Krikorian, R., Cohen, D.: The Internet of Things. Scientific American 291(44), 76–81 (2004)
2. Batra, D., Hoffer, J.A., Bostrom, R.P.: Comparing Representations with Relational and EER Models. Communications of the ACM 33, 126–139 (1990)
3. Cheng, P.C.-H.: Why Diagrams Are (Sometimes) Six Times Easier than Words: Benefits beyond Locational Indexing. In: Blackwell, A.F., Marriott, K., Shimojima, A. (eds.) Diagrams 2004. LNCS (LNAI), vol. 2980, pp. 242–260. Springer, Heidelberg (2004)
4. Larkin, J.H., Simon, H.A.: Why a Diagram is (Sometimes) Worth Ten Thousand Words. Cognitive Science 11 (1987)
5. Boekelder, A., Steehouder, M.: Selecting and Switching: Some Advantages of Diagrams over Tables and Lists for Presenting Instructions. IEEE Transactions on Professional Communication 41, 229–241 (1998)
6. Allmendinger, L.: Diagrams and Design Tools in Context. ACM SIGDOC Asterisk Journal of Computer Documentation 18, 25–41 (1994)
7. Coll, R.A., Coll, J.H., Thakur, G.: Graphs and tables: a four factor experiment. Communications of the ACM 37, 77–84 (1994)
8. Guiochet, J., Vilchis, A.: Safety Analysis of a Medical Robot for Tele-echography
9. Long, Z., Jinglun, Z.: Analysis and Study of System Safety Based on Event Sequence Diagram. International Journal of Computer Science and Network Security 8(2) (February 2008)
10. Ren, C.: A Safety Inspection Management System for Mine equipment Based on UML. In: 2009 International Conference on Signal Processing Systems (2009)
11. Allenby, K., Kelly, T.: Deriving Safety Requirements Using Scenarios. In: Proc. RE 2001, Toronto, Canada, August 27-31. IEEE, Los Alamitos (2001)

12. Alspaugh, T.A., et al.: Clarity for Stakeholders: Empirical Evaluation of ScenarioML, Use cases and Sequence Diagrams. In: Fifth International Workshop on Comparative Evaluation in Requirements Engineering (2007)
13. Sindre, G., Opdahl, A.L.: Eliciting Security Requirements with Misuse Cases. Requirements Engineering 10, 34–44 (2005)
14. Alexander, I.F.: Misuse Cases, Use Cases with Hostile Intent. IEEE Software 20, 58–66 (2003)
15. Larman, C.: Applying UML and Patterns – An Introduction to Object-Oriented Analysis and Design and Iterative Development, 3rd edn. Pearson Education Inc., London, ISBN 0-13-148906-2
16. Pender, T.: UML Bible, Wiley Publishing Inc., Indianapolis, Indiana, US (2003), ISBN 0-7645-2604-9
17. Stålhane, T., Sindre, G.: A comparison of two approaches to safety analysis based on use cases. In: Parent, C., Schewe, K.-D., Storey, V.C., Thalheim, B. (eds.) ER 2007. LNCS, vol. 4801, pp. 423–437. Springer, Heidelberg (2007)
18. Stålhane, T., Sindre, G.: Safety Hazard Identification by Misuse Cases: Experimental Comparison of Text and Diagrams. In: Czarnecki, K., Ober, I., Bruel, J.-M., Uhl, A., Völter, M. (eds.) MODELS 2008. LNCS, vol. 5301, pp. 721–735. Springer, Heidelberg (2008)
19. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic, Norwell (2000)
20. Tukey, J.W.: Data analysis and behavioral science or learning to bear the quantitative man burden by shunning badmandments. In: Jones, L.W. (ed.) The Collected Works of John W. Tukey, Wadsworth, Monterey, CA, vol. III, pp. 187–389 (1986)
21. Achour, C.B., et al.: Guiding Use Case Authoring: Results of an Empirical Study. In: Proceedings of the 4th IEEE International Symposium on Requirements Engineering – RE 1999, Limerick, Ireland (1999)
22. Cox, K., Phalp, K.: Replicating the CREWS use case Authoring Guidelines Experiment. Empirical Software Engineering 5, 245–267 (2000)