Thesis for Bachelor of Science in Computer Science
An Empirical Exploration in the Study of Software-Related Fatal Failures

Mälardalen University
Academy for Innovation, Design and Engineering

Author: Nikolaos Sycofyllos
nss13001@student.mdh.se

Date: 19-05-2016

Examiner: Prof. Daniel Sundmark
Mälardalen University, Västerås, Sweden

Supervisor: Eduard Paul Enoiu
Mälardalen University, Västerås, Sweden
Abstract

This thesis investigates and explores the subject of software-related fatal failures. In our technology-oriented society, deadly disasters due to software failures are not that uncommon as we might think. During recent years there has been a large amount of software-related fatal failures documented, although there have not been as far as we are aware of, any research studies trying to put those failures in the context of a wider evidence. That fact motivated us to answer two research questions: how many lives have been lost through failures of software and what is the nature of the main cause of software-related fatal failures. The aim of this thesis is to explore these questions and provide some empirical answers and also contribute to the knowledge of these failures.

Our goal is to provide an empirical and conceptual basis for investigating fatal software failures that will attempt to place these failure examples in a wider record. A similar study has been conducted by Donald MacKenzie [1] in the area of computer related failures but it is not directly answering our questions of interest and it is somehow outdated. Computer scientist Peter Neumann has done a lot of research in computer safety and is also the author of a wide collection of computer failure cases named “Risks to the public in computer and related systems” also called “RISKS” Reports. Those reports were the main source of our investigation and answers were given out of data collected from those reports. The methodology used in this research was an exploratory systematic review study. Starting off by defining Software-Related Fatal Failures (SRFF) and the inclusion criteria for the cases to be investigated, allowed us to avoid misinterpretations and collect the data in a better way. We searched through the “RISKS” reports and collected cases according to our criteria. The final collected data was reviewed and analyzed. Finally the results was illustrated and presented in terms of tables, plots, charts and descriptive statistics. We found out that in the “RISKS” reports, over 2600 people have lost their lives due to software-related failures and the majority of those failures had been caused by problematic user-software interaction. While answering our research questions we observed based on the information related to fatal software failures that the topic of SRFF is poorly investigated. Our research provides a good basis for future investigation and aims to trigger further research in the subject of software-related fatal failures.
# Table of Contents

Abstract 2

List of Figures 4

List of Tables 5

1. Introduction 6
   1.1. Problem Investigated & Research Goals 7

2. Background 7

3. Research Methodology 8
   3.1. Research Strategy 8
   3.2. Definition of Software-Related Fatal Failures 10
   3.3. Case Selection Criteria 10
   3.4. Case Collection & Data Extraction 14
   3.5. Analysis of Collected Data 16

4. Results 16
   4.1. Overview & Related Results 16
   4.2. Answers to Research Questions 20
      4.2.1. Human lives lost to software failures (RQ1) 20
      4.2.2. Nature of the main cause of SRFF (RQ2) 22

5. Discussion 24
   5.1. Threats to validity & Research Limitations 25
   5.2. Related work 26

6. Conclusion 27
   6.1. Future work 28

References 29

Appendix 30

Abbreviations 31
List of Figures

Figure 1. Research Methodology Diagram 9

Figure 2. Case and Data Collection Process 14

Figure 3. Deaths per Location-Graph 17

Figure 4. Fatal Failures per Location 18

Figure 5. Fatal Failures per Industry/Area 19

Figure 6. Data Accuracy of Cases-Chart 20

Figure 7. Software-Related Deaths through Time 21

Figure 8. Fatalities per Documented Years 22

Figure 9. Nature of the Main Cause of Fatal Failures-Chart 23

Figure 10. Number of Deaths per Nature of the Main Cause 24
List of Tables

Table 1. Demonstration of Collected Cases 15
1. Introduction

Software technology has a major impact in our modern society. In recent years software has found its way into almost every part of our everyday life. From helping in household chores and heavy industrial production to international transportation and advanced military defensive systems. Accidents caused by software problems or the improper use of software are not that uncommon as we might think. In many of these cases lethal injuries and deaths might unfortunately occur.

For example, in 1986, a software-related failure caused the death of several people because of overdoses from radiation. The main fault was a software error in the relationship between the data entry routine and the treatment monitor task of the therapy machine [2]. A more recent example is the case of an artillery cannon which misfired during training in 2002. The artillery crew were blindly relying on the output of the advanced field artillery tactical data system and forgot to set the correct altitude (which was set by default to 0), resulting in the gun aiming to low leading to the death of 2 soldiers and the injury of several others [3].

These cases are just two out of many, since 1978 until today, a large amount of accidents related with software have been documented, in many cases with lethal consequences. These cases motivated us to study this subject further and also raised some questions. How many lives have been lost because of software-related accidents? What is the nature of the main cause of these fatal failures?

In this thesis we are focusing mainly on those questions but also in how we can build knowledge in the study of software failures, since, as far as we know, there is a lack of research in the specific area. Through this thesis we can observe how well Software-Related Fatal Failures (SRFF) are studied and documented. The thesis itself acts as a basis and startpoint for further research and investigation as it will include, categorize and analyze software-related fatal failures spanning more than 30 years.

The research method we used is an exploratory systematic review study. Through collecting cases of fatal software failures from Peter Neumann’s “Risks to the public in computer and related systems” or “RISKS” reports, we gathered enough data to analyze it and finally answer our research questions. We discovered that the number of deaths caused by software-related failures is estimated to be over 2600 and the main cause of these software failures was the user-software interaction. The results are based on the “RISKS” reports and due to validity threats can not represent a broader conclusion. Based on our results we also observed that the area of software-related failures is poorly investigated, since most of the data supporting enquires and formal reports explaining the accidents are based on insufficient and unreliable data.
1.1. Problem Investigated & Research Goals

In order to understand the patterns and specific causes of failures we need to build some overall record containing the factors that are directly influencing these accidents. In this thesis we are reviewing fatal software-related failures, spanning from the late 1970 to the middle of 2010, by looking at the “Risks to the public in computer and related systems” or simply “RISKS” Reports in the Association for Computing Machinery’s (ACM) Software Engineering Notes (SEN). Even if these failures have been briefly studied and looked upon by scientists, the main problem is that we are lacking the knowledge to answer questions like these:

❖ How many deaths due to software-related failures, have been documented in the “RISKS” Reports? (RQ1)

❖ What is the nature of the main cause of those software-related fatal failures? (RQ2)

The data we used in this research for reaching our goals, is not only good for revealing the answers to our research questions, but will also provide a basis of information associated with fatal failures. We were interested in reporting the following information: which country and which industry is the most affected by software related failures and how well SRFF are studied and investigated.

The aim of this thesis was to explore these questions and provide some empirical answers but also contribute to the knowledge of these failures: indicate what might be involved in an empirical investigation. Our goal is to provide an empirical and conceptual basis for investigating fatal software failures that will attempt to place these failure examples in the context of a wider record of evidence.

Answering these questions is of high importance, especially in our computer-oriented society. This research tries to shed light on the causes of these failures and the questions around them.

2. Background

To answer our research questions, we needed to investigate the research done already in the area of software failures. Although there is a large literature and a lot of research on software and computer system safety, there is a lack of systematic and empirical studies studying a comprehensive set of software failures. Research seems to lack any major attempt to place these failures in a context of a wider record.

Peter Neumann, a computer scientist from the Computer Science Laboratory has contributed with detailed and in-depth studies in the area of software related risks, as
well as computer safety. His research includes a wide collection of computer related accidents documented in the “Risk to the public in computer and related systems” or “RISKS” Reports which are part of the Association of Computer Machinery’s Software Engineering Notes, established in 1976. These reports are helpful in the context of this thesis for answering the questions we are interested in. With the “RISKS” reports as our main source of investigation, we studied the record of fatal software-related failures. The timespan of these reports stretches from 1986 until today 2016. Worth mentioning is that the “RISKS” reports do not specifically address software-related failures, neither fatal accidents, they address all sorts of failures and risks in computers and related systems.

In our research we went through all the volumes and issues of those reports, from 1986 until the middle of 2010. The decision to stop searching in more recent reports was made because going through each case requires a lot of time and effort. Also, investigations into recent fatal failures takes time, even years and so the information became less reliable in terms of confirmed causes and detailed information. Factors which led us to stop the research during the middle of 2010.

MacKenzie’s “Computer-related accidental death: an empirical exploration” [1] is a very helpful study, but unfortunately outdated. We are unaware of any other investigations or research conducted in the area of software-related fatal failures. As far as we know no such contribution has been made during recent years.

3. Research Methodology
A research methodology is a systematic method that achieves a certain research goal. Our research is performed using an exploratory systematic review study.

In the “Cochrane Handbook for Systematic Reviews of Interventions” a systematic review is defined as an attempt “to collate all empirical evidence that fits prespecified eligibility criteria in order to answer a specific research question.” [4]. According to that definition, in this study we went through already existing reports, collected and analyzed data. A systematic review is the most appropriate method for this thesis since we collected the necessary criteria-based data and extracted specific information from already existing studies. The results of this review finally led to answers for our research questions.

3.1. Research Strategy
The research process began with the questions occurring from observing the software-related fatal failures. The first step was to investigate existing studies to understand the current situation in the area of SRFF and get a broader view of the subject. We then defined “Software-Related Fatal Failures” and “Criteria for the Case
Collection” to avoid investigating cases where the definition does not apply, resulting in collecting data we are not interested in. Definitions and criteria also help in avoiding misinterpretations. The most important step of the pre-collection work, was to know exactly what cases should be collected. Since many factors could affect our results we strictly collected cases and data according to our pre-made definitions and criteria.

The data and information was collected from the “RISKS” reports in the Association for Computing Machinery’s Software Engineering Notes according to a certain definition of SRFF. Cases not following the definition or collection criteria was not taken into consideration. The collected data was listed, analyzed and processed. In addition we used other sources to cross-check, verify and compare our findings. The final collected data was reviewed and analyzed. Finally the results was illustrated and presented in terms of tables, plots, charts and descriptive statistics. The tables, statistics and graphs provide information in being able to draw conclusions. Figure 1 illustrates the process and methodology of this research (“SRFF” stands for “Software-Related Fatal Failures”).

![Figure 1. Research Methodology Diagram](image-url)
3.2. Definition of Software-Related Fatal Failures

The first step in understanding this research is to have a clear definition of what a software-related fatal failure is, in order to avoid misinterpretations and to better focus this investigation as well as the collection of the related cases in the “RISKS” Reports.

**Failures:** By failures we mean external or internal incorrect behaviour, with respect to the requirements or other descriptions of the expected behaviour, which means that we only take in consideration accidents. We do not observe for example software-related military systems since “some computer systems are meant to kill people” [1]. Such cases are not investigated in this thesis, even if civilian bystanders have been killed by military intended computer systems. Those cases are very unreliable in gathering data since military operations can include, unfortunately, the killing of civilians [1]. Military-related cases though, are included in the research as long as a software-related failure has affected system, leading to an unexpected, unwanted behaviour. Such cases are difficult to be categorized since during military operations we can’t be sure if the cause of the failure was intended or accidental.

**Software:** As software we consider everything between embedded applications to large networks software, including electronic and control systems. We consider the software used in all systems and devices regardless their use and specification.

**Related:** The presence of software alone is not sufficient, so we investigate the cases where software is an actual part of the failure cause or very closely related. It would be too narrow if software was the only cause taking into account, as in some of these cases human-system interaction would be excluded. Software design which has a huge role in human interaction must also be taken in consideration and investigated. Usually major software and computer related accidents have multiple causes [1].

**Fatal:** Fatal accidents showed a larger amount of information and coverage than non-fatal accidents. Fatal software failures often trigger formal enquiries and press reports, which lead to considerably more reliable sources of information [1]. To investigate all software related accidents would be a too broad subject and a challenge for the scope of this thesis.

3.3. Case Selection Criteria

The definition of Software-related fatal failures clarifies what sort of cases we collected from the “RISKS” Reports. It is also important to define and explain what information we finally collected out of those cases, to help answer our research questions. The data of interest includes:
**Case number:** Needed for sorting cases and knowing the exact number of cases, but also used for more efficient processing of the data during the analysis-phase.

**Date(s):** Contains the year/s of occurrence of a specific fatal failure. The date/s is/are an interesting factor in observing whether the fatalities have decreased or increased over time. It also provides knowledge about when the most fatal failures occurred and when the first fatal software failures was documented. It is possible that failures have been occurring during a specific time period, for example between 1982 and 1990. In cases where the date of the failure is not documented or known, we categorized it as “Unknown”.

**Number of deaths:** Since this research revolves around fatal failures and the number of fatalities is the key for answering RQ1, the number of deaths is of high significance. It is one of the most important criteria in the case collection. The collected cases need to include at least 1 fatality which was caused during or because of the software-related failure. The number of deaths is needed to calculate the total number of software-related deaths and for answering the research questions. Cases which have an unknown number of deaths, are further investigated and either documented or ignored. If we lack accurate information about the death or cannot show evidence that the death is directly related to a software failure, then the case is ignored.

**Location:** As location we consider the geographical part where the failure occurred. A location is considered either a country, an ocean or sea, or unknown, in case no information can be acquired about where the software-failure happened. The location is important for showing the diversity in the failures, observing where the most fatal failures have taken place and adding knowledge to future investigations. Of course linking a software-related case with a location, is again based on the documented cases in the "Risks" Reports. Mostly the location is where the system including the software was stationed and operated or failed (e.g. airliner crash), not the location where the software and hardware was manufactured.

**Nature of the failure:** As nature of a failure we consider the aftermath. What the software-failure caused and resulted in. The nature of the failure is described shortly giving an brief overview to the reader about the failure or accident. The nature of the failure often helps us to define if a death was related to the software or to other factors. It does not directly answer the research questions, but provides a basis for further investigation and necessary information in the collection of failures we present.

**Main Fault(s) causing the failure:** Since software-related failures are complex and often includes multiple causes, we consider as main faults those who had the biggest impact in causing the failure. Of course those causes should at least include one software-related failure, preferably the main cause should be software-related. The
main faults are as well as the nature of the failure important for further investigation and valuable information in the failure collection. The main fault is important in defining the nature of the probable main fault, which will answer one of our main research questions. While knowing the main cause we can define the nature of the main fault. The cases investigated must have a software-related fault causing the failure in order to be collected, since we are interested in those specific cases. Otherwise, they are not taken into consideration. The main fault can in some cases be unclear or undefined, although we must know through the information at hand that the case is somehow software-related in order to be further studied. Defining and understanding the main causes can be really difficult, especially in advanced aviation or military systems where many causes and factors are involved. C.W Johnson’s “Looking Beyond the Cockpit: Human Computer Interaction in the Causal Complexes of Aviation Accidents” study [5], shows how to define probable causes of aviation accidents, information which has been of great assistance in understanding the causes of such accidents in this thesis.

**Nature of the probable main cause:** The nature of the probable main cause, together with the deaths are probably the most important data we are investigating. Discovering the nature of the probable main cause in software-related fatal failures will lead to the answer of RQ2. The nature of the probable main cause involves a problem, error or incorrect behaviour in either the hardware, software or human interaction. One important aspect for the categorization of this data is to define what the nature of the main causes could be. We have divided the accidents into five categories by extending the failure cause categories shown in MacKenzie’s “Computer-related accidental death: an empirical exploration” [1], according to the nature of their probable main software-related cause:

1. **Physical:** A physical failure or disturbance of a computer system using software. Hardware or mechanical problems leading to incorrect behaviour of the system.

2. **Software:** A failure in the software itself, causing the incorrect or unwanted behaviour of a system and causing a fatal failure.

3. **Physical & Software:** A combination of both physical and software-related failures. Either both of the factors were problematic, or the software failure affected the physical part or vice versa.

4. **User-Software Interaction:** For understanding the problems in this category we needed to define User-Software Interaction. It refers to the interfaces between humans and computers and specifically software interfaces. The Association for Computing Machinery (ACM) defines human-computer interaction as "a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major
phenomena surrounding them” [6]. That definition assisted in understanding which cases ought to be included in this category. Human errors are often considered to be the main cause of a failure, not only in designing and creating the software or hardware, but also in interacting with it while in operation [7]. We specifically collected only the cases where the interaction of human and software was problematic and not cases where only human mistakes were the main cause. In many cases a human error was the cause of the failure and the software although present, had nothing to do with it. In this thesis those cases are not of interest.

5. **Insufficient Data:** In this category we added data that had an unclear, unknown or undefined main cause. This category includes cases where we could not extract the necessary information to categorize it in any of the above categories, but knew that it is software-related and with more research the main cause could be revealed. Often those cases have plenty of room for further investigation.

**Area of usage:** The area of usage or industry where the software-related system is used and operating in. This data added diversity to our investigated cases and also is revealing the industry with the most fatal software failure occurrences. The cases are categorized to industries according to the sectors and subsectors (depending on the wideness of the specific area) of the Industry Classification Benchmark (ICB) [8]. Cases that included software that could not be defined or assigned to an Industry according to the ICB have been categorized as “Other”.

**Data accuracy:** The accuracy of the collected data is important to us but also for future investigations. It also helps us to get an overview of how well software-related fatal failures are studied and how the research in this specific area can be improved. All collected data have been discussed and mentioned in the SEN and “RISKS” reports. We have defined the Data accuracy of the cases in the following categories:

- **Good:** Mentioned in SEN, with a known main cause and with official papers and/or scientific reports supporting the data.

- **Poor:** Mentioned in SEN with a known main cause, but only newspaper articles and magazines supports that data.

- **Controversial:** Mentioned in SEN with unknown main cause and mentioned in journals or magazines.

**References:** References are an important part in documenting on which evidence our data was based upon, as well potentially helping future research. The references also
affect the accuracy of the data. In most cases the SEN or “RISKS” reports are the basic referencing source. These reports that were used in this thesis can be accessed through the ACM Digital Library. Sources like journals, magazines, scientific papers and official reports are also added in the case they were used to extract data.

3.4. Case Collection & Data Extraction

The most time-consuming but highly important phase of this research was the collection of cases and their data. As mentioned earlier we strictly collected software-related fatal failure cases, with their respective criteria-based information and data. The categorization and collection of the data was the basis for reaching the results leading to answering our research questions. Peter Neumann’s, “RISKS” in the ACM SEN was the main source of investigation.

These RISKS reports start in 1986 and have an average of 4 volumes per year (some years more, some years less), which contained a huge amount of cases that we needed to search through. The “RISKS” reports include a huge variety of risks to the public from computer-related systems (not necessarily fatal accidents). Stretching from economical miscalculations to hardware malfunctions and more, which resulted in a large amount of non-interesting cases that we had to somehow exclude. This thesis was time-limited to eight weeks, which means a time-efficient but still accurate strategy for reading the reports and collecting the cases of interest, needed to be followed.

![Figure 2. Case and Data Collection Process](image-url)
The diagram in Figure 2 illustrates the process of collecting cases and data from the “RISKS” reports (SRFF stands for “Software-Related Fatal Failures”). Instead of reading through all of the cases in all of the reports, we chose to use this method/strategy to discover SRFF-cases. The titles of each case in the report were often representative of the case being analyzed in that section. For example, if a case title was about economical failures, voting disorders or delayed traffic, that case was not of interest and was not investigated further.

Titles which included accidents, disasters or other various hints of cases involving fatalities, were read through carefully. If those cases were following our definition then they were categorized together with their respective information and data in our collection of cases, in the form of data listed in a spreadsheet. If crucial data (e.g. the number of deaths) was missing from a case then further investigation was made through research using other databases and search engines.

After reaching the end of the report, we went through the document one more time using the built-in “Find” operation available for each report. While scanning the report with keywords such as “kill”, “death”, “die” and “disaster “, we discovered cases that may not have been noticed through the first read-through. If the cases were related to our definition they were added to the collection while the others were ignored. The process ended when no more results were shown by the “Find”-operation. After this process ended a new report was investigated. The collection phase ended when we reached the last report of interest.

For demonstration purposes, we picked out some cases and displayed them together with some of the collected data in Table 1. In this way we can give a clearer image of how cases and data were sorted after being collected. Some of the data might be slightly altered and information reduced from the original collection just for exemplification purposes. The complete collection of cases can be found in the appendix.

<table>
<thead>
<tr>
<th>Date(s)</th>
<th>Deaths</th>
<th>Location</th>
<th>Failure’s Nature</th>
<th>Main Fault Causing Failure</th>
<th>Nature of main fault</th>
<th>Industry</th>
<th>Data Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1985-1987</td>
<td>3</td>
<td>USA</td>
<td>Overdoses from radiation therapy machine</td>
<td>Error in relationship of monitor-data entry routine</td>
<td>Software</td>
<td>Health Care Equipment &amp; Services</td>
<td>Good</td>
</tr>
<tr>
<td>1997</td>
<td>228</td>
<td>Guam</td>
<td>Airliner crash into mountain</td>
<td>Bug triggered incorrect altimetry in the GPWS</td>
<td>Software</td>
<td>Aerospace</td>
<td>Poor</td>
</tr>
</tbody>
</table>
3.5. Analysis of Collected Data
When all the data and cases of interest were inserted in the collection then we could analyze, sort, cross-check and build statistics. Since the cases were roughly added in a list we needed to sort the data. The data was sorted according to the “Nature of the main probable fault” and the “Date(s)” in ascending order for making the analysis easier.

After sorting the data we wanted to be certain that the data was accurate, so we compared and verified the cases through other available sources, either official reports, scientific papers or journals and magazines. It is worth mentioning that some cases could not be verified by more reliable sources than the SEN, leading in some cases to poor accuracy data. If the collected information and data agrees with our definitions the case is kept, otherwise it is removed from the list.

Finally we had the final list of sorted cases. From this collected data we created graphs and looked on the statistics to try to answer our research questions. In this scenario a new list was created by just including the years and the related deaths. For answering RQ1 we needed to calculate the total number of deaths caused by software-related failures. For answering RQ2 we firstly needed to calculate the amount of failures and deaths caused per each category in the “Nature of the probable main cause” column. After this step we compared the results in order to observe which category has the most deaths and which the most failures.

4. Results
After finishing the collection and analysis of the data we finally had some results that led to answering our research questions and contributing to the knowledge of software-related fatal failures. Those results have been divided in illustrated graphs together with statistics.

4.1. Overview & Related Results
Our final collection of SRFF-cases contains a total of 73 cases, stretching from 1978 to 2014. During the collection-phase we gathered an initial list of 85 cases. During the
analysis phases 12 of the cases had to be removed since they were not following our definition and criteria.
The rest of this section is dedicated to showing some overall data and information revolving around the cases we collected. Through our research we had the chance to have a look on data related to SRFF, which we considered interesting for the reader before presenting the answers to the research questions in Section 4.2.

![Deaths per Location Graph](Figure 3. Deaths per Location-Graph)
Figure 3 and 4 are illustrations of where most of the failures and their deaths occurred. The location with most fatal failures was the United States of America, with an amount of 24 out of 73 cases and a total number of 331 deaths. The locations with a big number of deaths are mostly airplane crash sites. Worth mentioning is that the United Kingdom had 9 fatal failures, whilst 8 of the 73 cases had an unknown location. Figure 4 shows where software fatalities have occurred. It is interesting to note that the location of the failures shows that there is a prevalence of cases reported in english-speaking countries which might affect the results of our thesis.
The 73 cases were categorized in 12 Areas/Industries, one of them being “Other”, which included cases where the sector could not be categorized to an industry according to the ICB [8]. As illustrated in Figure 5, the industry with the most fatal failures is the Health Care Equipment & Services, followed by the Aerospace with second most fatal failures and Defense with the third most fatal failures. The collected data also showed that the industry with most deaths due to software-related fatal failures is the Aerospace area with a total of 1907 deaths, while all the other industries combined have just 851. This is a fact which shows that just one software error perhaps can cause the death of hundreds of humans, especially travelers and airline crew.
Figure 6. Data Accuracy of Cases-Chart

Through our research we also discovered that 30 of the 73 cases had poor data accuracy, 22 had controversial data accuracy and only 21 had good data accuracy. Those cases with good data accuracy were mostly aircraft crashes and other incidents including big number of deaths and so triggering formal enquiries, official reports and scientific papers. Again we realize that the subject of software-related fatal failures is just briefly investigated and most cases do not benefit from any public investigation and in-depth studies.

4.2. Answers to Research Questions

From our collection of cases we extracted the data necessary to build the plots and statistics. Through our research we acquired the following results and answers to our research questions.

4.2.1. Human lives lost due to software failures, documented in the “RISKS” (RQ1)

The total amount of software-related deaths from our first documented case in 1978 until our last documented case in 2014 reaches the number of 2636. This number is calculated out of all the cases where we are certain that the data is reliable and accurate. If we also add the number of deaths of cases where the data accuracy is unreliable obtain a total of approximately 2758 deaths.

It is very difficult and controversial to give an exact number as an answer to this question as we cannot completely rely on the accuracy of some of the data we collected. In addition, other fatal software failures may not have been documented in the investigated reports, leading us to report less fatalities than the actual real number of deaths. Based on the data we collected, the number of such deaths, worldwide, up until
2014, is estimated to be over 2600. Since in reality the number of deaths could be larger than our calculated number, we are not showing a precise and broader answer. We are indicating an answer based on the “RISKS” reports and the cases included in them and not a general answer for all SRFF.

**Answer to RQ1:** Over 2600 lives have been lost, as reported in the “RISKS” reports, due to software failures.

We collected some more data related to the deaths caused by software failures, which are presented in the rest of Section 4.2.1.

![Software-Related Deaths through Time](image)

**Figure 7. Software-Related Deaths through Time**

Figure 7 illustrates the number of total deaths through different time periods. From 1978 until the year 2000 there had been a constant increase in the amount of deaths caused by software-related failures. In the time period 1991-2000 we collected a total of 1032 deaths, which is the largest number of deaths from any other time period. From 2001 until 2014 though, we have seen a decrease in software-related deaths. Again the data used in the results is collected from the “RISKS” reports and does not represent the total range and amount of software-related fatal failures.
Figure 8 presents a more detailed illustration of the number of deaths through shorter periods of time. It is interesting to mention that instead of a constant number of deaths every year there are big gaps in the number of deaths from year to year. That is mostly because of the huge amount of deaths involved in aircraft crashes. In almost every decade such an software-related accident occurred costing the lives of over 200 people in each such case.

4.2.2. Nature of the main cause of SRFF (RQ2)
Our results show that the nature of the probable main cause of the failures in the collected cases was user-software interaction. Figure 9 illustrates the percentage of fatal failures according to the probable main cause of the failure.
We observe that only 6.8% of the cases had a physical main cause and 16.4% only software as the main cause. Both physical & software was the main cause for 21.9% of the cases. Finally 26% of the cases had problematic user-software interaction as the main cause of the failure. Adding to the statement that this topic is just briefly and poorly investigated is the fact that 28.8% of the cases, a bigger percentage than all the other probable main causes, consists of insufficient data.

Out of our 73 cases, 21 of them had insufficient data for us to be able to categorize the nature of their main cause. User-Software Interaction was the main cause in 19 of the cases, while software alone was the cause in 16 cases. Both physical & software causes were present in 12 of the cases, while only physical causes were present in 5 of the cases.
Figure 10 shows the amount of deaths caused per nature of the main cause of the failure. In both deaths and cases user-software interaction is the main cause involved in these failures. A total of 862 deaths were categorized into the insufficient data category because the main cause was unknown. Even though most of the collected cases had an unknown main cause and so assigned to the Insufficient Data category, the majority of known cases were assigned to User-Software Interaction. Since most deaths were also assigned to the same category we consider the following answer to RQ2:

**Answer to RQ2:** Problematic user-software interaction is the main cause of SRFF.

### 5. Discussion

The results of this thesis are not as unexpected as some might think. The results are based on an exploratory investigation and a good basis for further investigation. Through the results and collection of cases we provided an empirical and conceptual basis for investigating software-related fatal failures, which attempts to place them in a wider evidence.

In this thesis our aim was to explore the research questions and provide some empirical answers. We are also able to contribute to the knowledge of these failures. We also indicated what might be involved in an empirical investigation of this kind.
We discovered that more than 2600 lives have been lost due to software related failures, and the main cause in most of these cases was the user-software interaction. The number of deaths might be a lot higher in reality, since we believe not all cases involving software failures has been included, as there are cases not mentioned in the “RISKS” report. Our data suggests that SRFF are not as well investigated as they should be.

Our answers to the research questions are of high significance since no earlier contribution have been trying to both answer them and put them in a wider perspective. We argue that the collection of these failures contributes to a solid basis for further investigation in this topic.

The used method was the most appropriate for this sort of investigation. We consider it difficult to have done it with any other research method since the goal was to collect and extract data from already existing reported cases.

5.1. Threats to Validity & Research Limitations
There are research limitations to our thesis that can affect not only the used research method itself but also the final results.

- Reports which were reviewed were only written in English, which means that other failures that are reported in other non-English languages, were not taken into consideration and were not used in the research as it would be difficult to discover them and draw accurate information.

- The “RISKS” Reports which are our main source of investigation may not have discovered and documented specific cases. Which means that it was not possible for us to add them to our collection of cases, since in this thesis we collected our information directly from the “RISKS” Reports.

- In the collection phase we went through all the reports that were available from the ACM Digital Library database. Some volumes and some issues of the “RISKS” were not available.

- In recent years due to the large amount of cases, the “RISKS” authors decided to include some cases in the “RISKS”-forums for further analyzing and discussion. Those cases where not taken into account in our research since most of them were inaccessible.
Some software-related failures have confidential information, for example military or space-related accidents. Which made it difficult to recover data for the thesis.

Due to the short-term period for research during this thesis, recent “RISKS” Reports and other sources could not be investigated. The time-limitation also contributed in the lack of further investigating cases with unreliable data in our collection.

The partial use of unreliable sources, which could include non-scientific information about software-related failures [1], are also a limitation to the thesis and a threat to the results.

Old cases were investigated using possibly outdated reports which could result in outdated information used in our results.

During both the research and the report we have carefully defined both our area of investigation but also the criteria for data we collected. Unfortunately we could not avoid this completely. Our results are mostly threatened by the probable causes:

Undiscovered or inaccurate data affects both the number of deaths and the nature of the main cause of SRFF, as well as our collection of cases.

Possible failures in the interpretation and investigation of cases by the author of this thesis but also the “RISKS” authors.

5.2. Related work
Although there is some similar research conducted in this area, software-related failures are not that well studied as they should be. There is a lot research and studies about computer and software safety but almost none of them gives a broader view of fatal failures and deadly hazards that occur through the use of software.

MacKenzie investigated these failures [1] in the context of computers and looked at cases from 1978 until 1992. His work is an important resource and starting point for this thesis. Some of MacKenzie’s cases have been included in the data collection even if they might not be present in the “RISKS” Reports. Since it is as far as we now the only collection of computer-related fatal failures, his software-related cases seemed suitable to be included in our research as well. Although, this study is providing solid information and answers to some of the questions we are answering, it does not cover accidents after 1992. Also some of the information collected in this cases could be
outdated. The study in [1] uses among other the “RISKS” reports as a source of information gathering.

MacKenzie’s results suggests that computer related deaths can be estimated to 1100, although the study suggested that the real number of deaths could probably be higher than that, since many cases especially in non-English talking countries had not been investigated [1]. The study discovered most of the deaths were because of Human-Computer Interaction failures, although the study did not consider it the main cause of failures. [1]

Computer scientist Peter Neumann who is the author of the “RISKS” reports which are included in the ACM Software Engineering Notes has also written the book “Computer-Related Risks” where he analyzes the risks and proposes solutions but does not mention fatal cases and the main causes of those accidents [9]. Neumann has also written several articles and reports about computer and software accidents, but those cases tend to be scattered, not analyzed and cannot contribute to our research since there is a few reviewed fatal accidents and not every case is strictly software-related [10]. His work is a great commitment to this subject and was of great assistance in this specific research.

In specific cases, computer-related accidents like the Therac-25 [2] and several spacecraft accidents like the Ariane 501 [11] investigation has been conducted by Nancy G. Leveson among others. Leveson provides rich sources of information with focus on the causes but still fail to put the software accidents in a broader perspective.

Our collection and results of the Software-Related fatal failures strives to put those failures in a broader context and filling the gaps left by other studies in this specific topic. The results of this thesis are also contributing to the topic and trying to shed some light in the undiscovered questions of the area.

6. Conclusion

Through the research and results of the thesis we understand the importance of software-related fatal failures and their investigation. Society is tending in becoming more and more oriented towards technology and software-related systems. We have discovered that software can cause horrible disasters, taking hundreds of lives. We have discovered that many industries and countries are affected by such disasters and still no investigation or research, as far as we know, have been able to contribute to this serious subject. Our contribution is just a small and limited investigation, but we want to trigger further interest for research on the topic.
Engineers, programmers and designers focus nowadays in making the most user-friendly software and efficient software, but through our research we have also discovered the danger of using software. Which is something that should be studied and finally improved. No computer system is ever going to be 100% guaranteed to behave properly and people will always be a source of problems [12] but we can always strive to make software (and our lives) safer and better.

6.1. Future Work
As earlier mentioned in the report, this thesis provides a good basis for further investigation on the topic of software-related fatal failures. While providing some empirical answers to software-related questions and statistical results it aims to trigger further investigation and in-depth research. As far as the case collection, more sources can be studied for a more complete collection of software-related fatal failures. Not only more sources can be studied but the already documented cases can be further investigated for more accurate data. Of course those research questions in this thesis are not covering completely the topic of software-related fatal failures. More questions exist that needs to be answered.

It is important in future studies to lead the research towards a way of understanding the problematic behaviour in software and striving to discover solutions and methods of hindering such failures taking human lives and disturbing our daily life.
References


Appendix

Link to the Excel Sheet including the complete collection of Software-Related Fatal Failures, gathered for this thesis:

https://docs.google.com/spreadsheets/d/1H0ouawPlaLZXueKUZ54fn2T3etxM-zghlQr9etZljrQ/edit?usp=sharing
## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RISKS</td>
<td>Risks to the public in computer and related systems</td>
</tr>
<tr>
<td>SRFF</td>
<td>Software-Related Fatal Failures</td>
</tr>
<tr>
<td>ACM</td>
<td>Association for Computer Machinery</td>
</tr>
<tr>
<td>SEN</td>
<td>Software Engineering Notes</td>
</tr>
<tr>
<td>ICB</td>
<td>Industry Classification Benchmark</td>
</tr>
</tbody>
</table>