Automation of Common Cause Analysis - Monitoring Improvement Measure Performance and Predicting Accidents

Dipl.-Inform. Jan Sanders
Computer Networks and Distributed Systems Group
Technical Faculty, Bielefeld University
mail: jsanders@TechFak.Uni-Bielefeld.DE
web: www.rvs.uni-bielefeld.de

Zur Erlangung des akademischen Grades des
Doktors der Ingenieurwissenschaften
der Technischen Fakultät
der Universität Bielefeld
vorgelegte
Dissertation
von Jan Sanders

September 8, 2014

Gedruckt auf alterungsbeständigem Papier nach ISO 9706

Acknowledgements

There are many people who have supported me in one kind or another with my thesis. Of course there is my thesis supervisor, Prof. Peter B. Ladkin PhD. I would not have taken up writing a thesis without his encouragement. I am very grateful that Peter has always shown keen interest in the course of the thesis and is also a frequent user of the software, which has given me plenty of opportunity to fix all the things that did not work. Apart from thesis and software I have learned too many things from working with Peter since I started working with the RVS. I would also like to thank Dr. Bernd Sieker, who also works in the AG RVS, is interested in accident analysis and has probably found the most bugs in the software. I am very much indebted to all the people I have worked with at the Uni Bielefeld. While having a normal computer support job at the AG Rechnerbetriebsgruppe I always had the freedom to work on my thesis when there was some slack. Many thanks go to Prof. Dr, Andre Doering, who offered to be my second thesis advisor. Its been some time since we have done WBAs together. And also to the Prof. Dr. Franz Kummert and Dr. Thorsten Schneider for being on my Promotionsausschuss.

Preface: Thesis

Common cause analysis for accidents is a necessity for effective safety engineering. Modern accident databases, e.g. [7][10][1][9], offer a great number of accident reports and help safety engineers to find reports for known accidents quickly.

The databases mainly serve as report repositories, to search and access specific reports. Some predefined commonalities of accidents can be used in search queries of databases, but there is no provision to analyse databases for common causes in accidents, especially if the common causes are yet unknown to be common causes.

Common cause analysis, obviously, needs causal analysis. There are causal analysis methods which offer the prospect of more structure in describing cause-effect relationships in accidents. State of the art is to formulate accident reports as texts, which is very difficult to parse for cause-effect relationships.

Causal analysis methods[27] offer more explicit descriptions of cause-effect relationships.

If accident report databases must be searched automatically for common causes, the problem of finding common causes has to be solved algorithmically. This entails, that cause-effect denotations must be machine understandable with well defined semantics.

AcciMaps[24], one example of a causal analysis method, has the feature to formally indicate accidents' cause-effect relationships. Unfortunately it lacks well-defined semantics for what a cause-effect relationship is. That means that an algorithm could in principle search for patterns in AcciMaps, but the semantic evaluation of the results must be done in the head of the user/investigator. The algorithm itself cannot be used for causal reasoning.

This thesis will show that with well-defined causal semantics, algorithms can be developed which offer causal reasoning capabilites and can be used for improved funcitonality in accident databases. The accompanying software is a proof of concept. It contains algorithms for

- searching for causally-specified system behaviour,
- comparing two or more accident descriptions for similarity and
- finding previously unknown common causes of accidents automatically.

Contents

Chapter 1

Introduction

Accident Analysis Accidents are analysed to see if there is a way to improve a system, so that future accidents can be prevented. No one can foresee all the hazards that a man-made system encounters during the course of its life. Many man-made systems are simply to complex for even the most advanced methods of safety analysis to make things 100% safe. At the same time the consequences of system failure can be so big that each single prevented accident is worth the effort. Think nuclear power plants, for example.

Every time such a system fails, even if it is not an actual accident, but a near-accident, the system's behaviour is analysed and the conclusions drawn from such analysis may form the basis for significant improvement.

When there are many similar systems, some of which have accidents from time to time, the accident analyses are gathered to do meta-analysis. The analysis of many accidents brings additional insight into the safety properties of a system.

If there are many options in which to improve a system, then meta-analysis can tell where to start. The most severe and most frequent types of accidents have to be addressed first, especially when resources are scarce. System improvements may also be mutually exclusive. A soldier's gun must be operational in action, but safe it is not used. A failure to fire when needed can have as severe consequences as an accidental shot among friendlies. Every measure that is taken to improve the safety of a gun when it is not needed must hinder the gun to fire. Introducing such a safety measure decreases the reliability of the gun and thereby poses a safety threat during combat operations. If enough data is available an informed decision can be taken. Since both cannot be achieved at the same time the kind of accident that poses the greater risk can be eliminated.

Every improvement to a system has to be monitored. It is the hope of any accident analyst that safety recommendations can be drawn from his/her work, but the recommendations have to be proven in use. This can only be achieved by monitoring the safety performance of an improved system. If the rate of a certain kind of accidents reduces or if the amount of harm done in a certain kind of accident reduces, then the improvement is successful. If it does not, other interventions should be explored, so that resources can be spent where they are needed and effective.

Meta-analysis is mostly done by counting how many accident of a specific type have occurred. Many nations regularly publish road accident statistics, for example. How many aircraft have overrun the runway? How many have flown into terrain? How many have lost an engine? etc. These are examples of accident classes according to outcome. Other classes of accident are categorised according to cause. How many road accidents have been caused by drunk driving? How many road accidents have been caused by speeding? How many road accidents have been caused by deer? Trends in the frequencies of such accidents are monitored so that from time to time you will read that this or that is the greatest threat to road safety at the moment.

One drawback of counting how many accidents of a specific type occur is that the type must be known. If there are 10 different accident types for a system, then either one of the 10 types has the most accidents or the "other" category has the most. Some types wane while other emerge when the "others" are inspected closer. Once a new type of accident is deemed interesting the old categorisation may not fit any longer. This is either ignored, so that the history of accidents is not categorised according the new scheme, or all past accidents have to be reclassified. Depending on the number of accidents this may well be too much.

Why-Because Analysis and Common Cause Analysis Why-Because Analysis is a causal accident analysis method. One of the results of Why-Because Analysis is the Why-Because Graph, an acyclic, directed graph depicting the causal relations between accident factors. The causal relations follow a formal notion of causality, so the edges of the Why-Because Graph have a well defined meaning. By looking for recurring patterns in Why-Because Graphs of different accidents commonalities in causation can be found among the Why-Because Graphs.

When designing a new system the safety engineer looks for factors which can be causes for many different types of misbehaviour. Eliminating common causes reduces the probability of many different accident types. The term widely used in a priori safety analysis.

In accident analysis however, in a posteriori safety analysis, the term is not so often used. If different accidents of the same system share a cause, it would have been termed a Common Cause in a priori safety analysis. So it is reasonable to refer to commonalities in accident causation which have been identified by accident analysis as Common Causes.

Finding Accidents Before They Happen With the availability of computer aided accident meta-analysis, as described in this thesis, the number of cases which can be processed increases. If it was only feasible to analyse accidents, it may become feasible to also analyse near-accidents or incidents. Common failure causes may become apparent during the analysis of incidents before a real accident happens. This is even more so when newly emerging trends can be found algorithmically.

Chapter 2

Technical Terms

Throughout I will use terms which have special meaning. In some cases the terms will be obvious, because they are not part of day-to-day language. In the cases where terms used are part of day-to-day language the terms may denote different things from their technical homonyms in the language of accident analysis or system safety.

This section introduces many of the terms and the meaning they will have throughout this thesis. This also has the benefit to lay out core concepts of causal analysis of accidents in particular and system behaviour in general. Even among safety professionals there are sometimes arguments about the meaning of phrases[1].

2.1 Systems

2.1.1 Teleological Systems

The term "system" can practically mean anything, just like "thing" or "object", so there is some need to further specify which types of systems are of interest in this thesis. For accident analysis systems are of interest which are systems that serve a purpose and systems over which we have influence.

Systems with a Purpose I could call everything a system, nothing prevents me from saying that the arrangement of pencils on my desk is a system. I can even give it a name: "P". As should be obvious, this is not the kind of system that accident analysis concerns itself with. I have just arbitrarily "made" the pencils a system. The system P lacks a purpose[2], it is not a teleological system.

[1]For an example please see the discussion of the different definitions for "hazard" in Chapter 5, pp 71 of [27]

[2]Yes I know, now it has a purpose, namely demonstrating what a purposeless system is.

Take the Fukushima Dai-Ichi nuclear power plant for example. The plant was struck by an earthquake and a tidal wave, which resulted in one of the worlds most severe nuclear crises. The tectonic plates whose interaction caused the earthquake and the tidal wave are a system, but not a teleological one. The plant itself is man-made for the express purpose of power generation, it is a teleological system.

Influence on Systems Accident analysis aims to learn from past mistakes in order not to repeat them in the future. If systems behave in an unwanted way and the causes are known, then the systems can be changed in the light of the knowledge gained from the analysis of unwanted behaviour.

To prevent future accidents, the way in which a known accident occurred must be found out and then the system must be changed. Change can only be accomplished in so far as there is a possibility to influence a system.

In the above example the tectonic plates clearly behaved in an unwanted way, but there is no way we could influence the tectonics to prevent future earthquakes. The power plant is man-mand and there is considerable influence over its design, construction, maintenance, operation, decommissioning, etc. At least in principle.

2.1.2 Systems, Environment and the World

When we analyse unwanted behaviour of teleological systems the surroundings of the systems' whose behaviour we want to understand play an important role. The environment consists of other systems, teleological, influenceable and otherwise. Everything belongs to a system's environment with which the system interacts, all else is the world.

In the Fukushima Dai-Ichi example the teleological system "power plant" is of central interest, because it is the system we need to change in order to reduce the likelihood or impact of a nuclear crisis. The tectonic plates and the ocean are the environment of the power plant. They are only interesting in so far that they influence the behaviour of the power plant.

2.1.3 System State and Behaviour

A system state is a (sufficiently) complete description or a system at a fixed point in time. If the description changes, then the state changes.

If we observe a system and note all the state changes we have a history of system states. This is the systems observed behaviour (Behaviour-1).

From our knowledge of the system and its environment we can infer future Behaviour-1 under different circumstances. What will be the observed (or what is the expected) system state history in environment X, what will it be in environment Y or in environment Z. The way in which the system changes its state depending on the state of the environment and its own current state is the systems potential behaviour (Behaviour-2).

If we have observed unwanted states in Behaviour-1 then we can change the system and thereby its Behaviour-2, so that the unwanted states are more unlikely or less harmful in a systems new Behaviour-2.

For example, after observing a car crash, we may want to change the car. The introduction of anti-skid brakes reduces the likelihood of an accident and the introduction of airbags reduces the harm inflicted in a crash.

2.2 Different Kinds of Accidents

Accident analysis is the analysis of unwanted Behaviour-1 of teleological systems with respect to their environment.

When is behaviour unwanted? All kinds of behaviour could be unwanted. If unwanted behaviour warrants an analysis depends on the amount of harm done. On a side note: In before-the-fact safety analysis the likelihood of harm being done will also be a factor, but in accident analysis, after-the-fact analysis, we know that harm was already done.

Severity Threshold Let's assume that all system behaviour is results in a measurable amount of harm. If the measurable amount of harm lies over a certain threshold then it is unwanted behaviour, if it is below or on the threshold then it is normal behaviour.

In practice there are even more severity thresholds. For example the International Civil Aviation Organisation has two severity thresholds [21] explicitly stated for the purpose of accident investigation. The ICAO distinguishes between "Accident" and "Incident" as defined in [21]. All above the Accident-Threshold is an Accident, all between the Accident-Threshold and the Incident-Threshold is an Incident and all below the Incident-Threshold is not to be reported for investigation.

An "Accident" is an "occurrence [...] in which [...] a person is fatally or seriously injured" and/or "the aircraft sustains damage or structural failure" or "is missing or is completely inaccessible". An Incident is an "occurrence other than an accident, [...] which affects or could affect the safety of operations."

Both definitions relate to the concept of "Occurrence" which we may infer from [21] to be a behaviour which deviates from normal behaviour, so all "Accidents" and "Incidents" are "Occurrences", but not vice versa.

Occurrences by Severity In principle there is no limit on the number of thresholds, as long as the words to denote them do not run out. Common accident types found in safety literature are terms like "mishaps", "near misses", "catastrophies" or "minor issues". Thresholds can be set arbitrarily

Which number of thresholds to use and which level of damage or harm to set a threshold is up to an anlyst, as long as it fits the analysts need.

Most of the time in this thesis I will use the term "accident" to denote all unwanted behaviour of teleological systems. In some context a distinction between different types of accidents will be necessary. I will make it apparent to the reader when a specific type of accident is meant by using *italics*.

- The term *occurrence* will denote all kinds of unwanted behaviour of teleological systems.
- The term *accident* will denote the most severe kind of unwanted behaviour, above the *accident threshold*.
- The term *incident* will denote all unwanted behaviour below the *accident threshold*, but above the *incident threshold*.
- The term *mishap* will denote all *occurrences* below the *incident threshold*.

In principle thresholds are arbitrary, but in practice they will be set so that they best serve their purpose. Depending on the amount of *occurrences* over time the thresholds may be set to allow proper handling of the most severe *accidents* while completely ignoring *mishaps*. Handling *occurrences* may be costly, so cost-benefit trade-offs have to be found and thresholds have to be set accordingly. In Germany for example, road accidents go on record without damage if the damage in money terms is below EUR 25.

Chapter 3

The *Occurrence* Investigation Process

3.1 *Occurrence* Investigation Process, OIP

Accident investigation is not a means to itself, it does not happen in a void. Accident investigations aim to improve the knowledge of system behavior (2.1.3, p. 16), in order to improve systems, assign blame[1] or gain awareness on the state of things.

Other things than accdidents can also be analysed using the same methods. Some domains have specific meaning attached to the term "accident" and distinguish between different forms of occurrences.

But this does not change the means nor the aims.

This chapter outlines a general *occurrence* investigation process. Although implementations differ, the general process is the same in aircraft accident investigation or in company quality assurance processes. At the different stages of the process I will indicate how the presented approach and the WBA software can support the stage.

The purpose is to give an overview where the presented approach and the software will fit into the process. This is not intended as an in depth discussion of the analysis process and its subprocesses.

3.1.1 Single *Occurrence Analysis*

Reporting and Detection Anomalies are either reported, when they are observed by someone, or detected automatically. Without knowledge of an anomaly there can be no investigation. For example, in industrial quality assurance programs, some product flaws are detected during production by dedicated sensors, while others are reported by the customer to quality assurance staff.

[1]Although this is frowned upon in the System Safety community

When an anomaly is made known to an investigator the investigator must decide if

- the anomaly fits the requirements for any kind of further investigation,
- is of sufficient interest,
- and/or promises sufficient new insight.

For each type of detected anomaly there may be rules which state what is to be investigated and if so, how much resources are to be spent[2].

Fact Gathering The fact gathering phase establishes the state of the environment and the systems involved in an*occurrence* at the time of the *occurrence* and the relevant time frame before the *occurrence*.

E.g. in the aviation domain fact gathering means to retrieve data recorders of air and ground systems and download their data, gather debris in case of structural damage and interview operators and witnesses. Cockpit voice recorders of commercial aircraft will only contain the last 30 minutes before the device stopped recording, which is, for cockpit voice recorders, considered the appropriate relevant time frame before an *occurrence*. CVRs and other recorders also reveal information from which the environment of the system at the time of *occurrence* can be inferred.

In other domains, with simpler systems, fact gathering has already been done simultaneously with reporting.

The WBA software allows management of the gathered facts. The list of facts can be annotated to keep a record where facts have come from, provide justification and other annotations. The list can be used directly in the analysis stage.

Reconstruction From the gathered data the history of an *occurrence* should ideally be reconstructable. This is the first phase where insufficiencies may become apparent and further fact gathering is warranted.

E.g. flight recorder data is analysed to reconstruct the history of states 2.1.3 an aircraft went through before an accident.

Analysis There are many ways in which an *occurrence* can be analysed. There exist a large number of frameworks, models and methods for analysis and simple systems may be satisfactorily analysed completely without them.

This thesis focuses on causal analysis, so at this point a causal analysis method would be used to develop an understanding of the facts and history of an *occurrence*. Other non-causal methods exist. Some rely on checklists and/or questionnaires (also used for fact finding) to identify points of

[2]This is done mostly in terms of a description of the investigation process appropriate.

interest. Some rely on deviation of the reconstructed *occurrence* from a predefined reference model.

For a comparison of different analysis methods see [4], [30], [11] and [27].

The software supports causal analysis methods like WBA and, to a limited extent, AcciMaps. Other methods may be possible, but should be supported by a clear statement which elements of the software denote which relations of a method.

Improving Systems An anaysis can lead to system improvements, e.g. by implementing countermeasures that prevent unwanted behaviour or its causes.

The software can document implemented countermeasures. The documentation of countermeasures is integrated into the WBG clearly showing which factors in an *occurrence* are affected by it.

This means that countermeasures can also be integrated into the Automation of Common Cause Analysis, see "Monitoring System Improvements" below.

Preventing *Accidents* This is the prime motivator for *accident* analysis. To prevent *accidents* which are known to happen, either because they have been observed or because it can be inferred from observes *incidents* or *mishaps*, trumps other system improvements. Improvement of unwanted behaviour (3.1.1) in general could mean, e.g. to improve economic performance. Preventing *accidents* may even lead to system changes which would otherwise be considered unwanted behaviour. The improvement of system safety can negatively influence other performance factors of a system.

3.1.2 Meta-analysis

Monitoring System Improvements Some analyses will lead to system changes to improve their safety, reliability or other desired properties. The effectiveness has to be monitored. The change of the rate of occurrence of specific unwanted behaviour, which the system improvement should address, will indicat the success of the improvement. With the integration of countermeasures into the WBG an algorithm can search all known WBGs in a database for an *occurrence* and also distinguish between those where countermeasurs have been implemented and those that don't. Distiction between different kinds of countermeasures is also possible, allowing comparative analysis.

Marking Failed Attempts to System Improvement Failed improvements, which have been identified, e.g. by the above mentioned monitoring, need to be documented. Otherwise attempts may be tried twice. In the same way that countermeasures are integrated into the WBGs failed countermeasures are. Before a system change is implemented a system designer can research whether the proposed countermeasure has been tried and with which success.

Early Warning Process Another stage can be included if Automatic Common Cause Anaysis can be implemented. *Incidents* can be analysed for similarities with other *incidents*. Risk assessment data available for a system can be reassessed with the new found knowledge of frequently occurring common causes of *incidents*. The reassessment may result in a system change if the risk assessment results no longer hold in light of the *incident* data. For example, frequently occurring common causes could easily lead to *accidents* instead of *incidents* in slightly different circumstances. In that case there is early warning before an *accident* happened.

Analysing *incidents* and applying Automatic Common Cause Analysis can prevent *accidents*.

Chapter 4

Accident Investigation Report Databases

The level of improvement of the proposed approached can only be compared with current practice. As noted in the introduction, the most sophisticated accident investigation bodies have accident databases which are open to the public. Four of them will be presented here.

There is also a project by the U.S. Department of Transportation, which researches improvements in accidnet databases, among other things. Some of the requirements of the project are addressed by this thesis and implemented in the WBA Toolkit.

4.1 Report Corpora

4.1.1 U.S. National Transportation Safety Board (NTSB)

The NTSB is the U.S. body for accident investigation in th aviation, railway, highway, marine, pipeline and hazardous materials domain. On the NTSB website you can find the complete NTSB accident investigation report corpus[7]. The corpus is searchable by a number of criteria, such as mode (e.g. aviation, railway) and date of report. The site search of the NTSB can also be used for full text search in the complete accident investigation report corpus.

This at least is the interface that is open to the public to analyse the investigation report corpus. In principle the availability of all accident investigation reports can be downloaded and used for further processing. If the neccessary resources are available the accident investigation reports could be manually processed and converted into whatever format is desired. The text structure of the PDF files containing the accident investigation reports is known, so that text mining can be used to extract information from accident investigation reports. That would be the approach for automatic processing. A hybrid approach can also be imagined where manual processing is supported by automation for a better cost-benefit ratio.

4.1.2 Australian Transportation Safety Bureau (ATSB)

The Australian ATSB[10] offers nearly identical services as the NTSB, but is restricted to aviation, marine and railway domains.

4.1.3 Canadian Transportation Accident Investigation and Safety Board

The CTAISB[28] investigates accidents in the marine, aviation, railway and road domain. It also closely resembles the U.S. NTSB.

4.1.4 French Bureau d' Enquêtes et d'Analyses pour la securite de l' aviation civile (BEA)

The BEA[14] offers a keyword, type of aircraft and date search for its accident investigation report corpus.

4.1.5 German Bundesstelle für Flugunfalluntersucung

The BFU[20] offers all accident investigation reports on its publication website. The search functions are limited to date and type of aircraft based search and a keyword search[1].

4.1.6 U.K. Air Accidents Investigation Branch

The AAIB[9] is the U.K. body for aviation accident investigation. Unlike the NTSB it is specialized on aviation accidents and does not investigate other domains. Those are covered by other investigation bodies. The AAIB report site[2] features search mask with keyword search. The keyword search is not a full text search in the investigation reports, so for easy access the AAIB site offer less in terms of accident investigation report analysis.

4.1.7 NASA Aviation Safety Reporting System

The ASRS[1] is a self reporting system. NASA provides the ASRS web service for all who wish to report on an aviation *occurrence*. The report database is open to the public, both for searching the database contents an to providing reports to the database. The database query website is a very good example of a searchable database with predefined classes assigned to reports(see 8.4).

[1]For some reason the keyword does not find phrases that contain a line break.
[2]http://www.aaib.gov.uk/publications/index.cfm

4.1.8 Boeing Statistical Summary of Statistical Jet Airplane Accidents

The StatSum[8] is a statistical summary report of commercial heavy jet aircraft[3]. StatSum contains lists of brief descriptions of aircraft accidents for the year under consideration. The yearly safety record is compared with past years, according to a number of different measures, e.g. flight hours per accident or cycles per accident. The statistical summary goes bach until the year 1959. The StatSum report is a very good illustration of the aviation sector's safety track record and the progress in safety that has been achieved.

4.2 Improving Report Corpuses

All institutions regularly publish statistical summaries of accident rates, fatality and accident numbers. The U.S. Department of Tranportation[46] and the NTSB[5] have an ongoing program to improve the utility of safety data. As will explained (see 8.1) there are limits to how safety data can be computationally processed. [46] explicitly states that research should improve

- ”common data on accident circumstances” and
- ”better data on accident precursors”.

Both of which is addressed in this thesis. Automation of Common Cause Analysis will make discovering leading accident indicators, that is data indicating likely accidents before they happen, feasible and reduce the reliance on lagging, after the fact, indicators.

[3]More than 60.000 pounds maximum gross weight

Chapter 5

Introduction to Why-Because Analysis

5.1 Accidents and Causality

5.1.1 Fukushima: Three Disasters in One Day

At the time of writing the last Big Disaster happened on March 11th 2011 in Fukushima Prefecture, Japan. The first disaster was the so called Tohoku Earthquake, which occurred approximately at a quarter to three, local time. The epicenter of the quake was approximately 70 km off the Oshika Peninsula the coast with a magnitude of 9.0. A magnitude 9.0 quake is an immensely powerful earthquake, indeed it was one of the most powerful earthquakes ever recorded. The close proximity to the north eastern part of the Japanese main island of Honshu and the huge amount of power meant serious destruction.

Then there was a second disaster following immediately. Shortly after the earthquake a tsunami struck the east coast of northern Honshu. The tsunami was a direct result of the moving sea bed. The tsunami traveled as far as 10 km inland and wiped out entire villages.

After the tsunami retreated a third disaster was already becoming apparent. The Fukushima Dai-Ichi nuclear power plant was first struck by the Tohoku quake, then by the tsunami and this caused problems with decay heat removal of reactor cores and spent fuel pools and containment of radioactive matter. A nuclear disaster followed.

It is unfortunate enough to be the victim of an earthquake *or* a tsunami *or* exposure to radioactive contamination, much more so to be a victim of an earchquake *and* a tsunami *and* exposure to radioactive contamination. In tha case of Fukushima it was *not* bad luck to be exposed to all three disasters, since the three disasters did not occur independent of each other. The three disasters were linked causally. One causing another. The earthquake caused the tsunami and damaged the Fukushima Dai-Ichi nuclear plant. The tsunami further damaged the nuclear plant and left it in a state that was not longer controllable.

Two of the disasters are naturally occurring disasters. The nuclear crisis could only happen because there was a man-made system with the potential of an uncontrollable radioactive reaction. This potential was realized, both by the existence of the plant and the preceding natural disasters.

From the point of view of accident analysis only the man-made system is within the ability of humans to control and influence. An accident analysis will focus on the nuclear accident and try to find out what went wrong. That does not mean that analyzing the quake and tsunami does not have benefits. But since humans can only change the design of the nuclear plant, and not of tectonic plates or the sea, the two natural disasters are only of interest insofar that they concern the safe operation of the plant.

5.1.2 Concepts of Accident Analysis

The Fukushima story illustrates some basic and important concepts of accident analysis. Some occurrences are accidents, others are not. How do we distinguish them? Another important concept is causality. Everybody has an intuitive grasp on the concept of causality, but for accident analysis it is very helpful to have a formal understanding of the concept. As with accidents, causality in this scope will be used as a technical term, to be distinguished from its day to day use.

For the scope of this book the term Accident, and some other related terms, will have special meaning. They are technical terms, which will receive special attention in this book.

Technical Terms

Technical terms will be introduced throughout this book. Some are part of jargon and are widely used in safety literature. Not all literature uses the exact same definitions as I do, but they will be sufficiently similar to allow the understanding of other texts on the topic. Jargon is used to express very specific things that day-to-day language cannot easily. It is helpful to learn the jargon presented here.

Technical Terms: Accident and Incident

There are many definitions out there that define the term Accident. One good example is the definition used by the International Civil Aviation Organization ICAO.

Accident - ICAO *"An occurrence associated with the operation of an aircraft which takes place between the time any person boards the aircraft with the intention of flight until such time as all such persons have disembarked, in which (a) a person is fatally or seriously injured [...] or (b) the aircraft sustains damage or structural failure [...] or (c) the aircraft is missing or is completely inaccessible."*

The definition is about aircraft accidents and does not aim to give a general definition of what an Accident is. The scope is even further narrowed e.g. by the "between the time" phrase. The ICAO is interested in Accidents of a specific type, so it is reasonable to exclude from the definition all oc- currences e.g. not "associated with the operation of an aircraft". On a side note, "associated" is not well defined, in a formal way, but for our purposes it should be fairly obvious what the meaning of the statement is. ICAO is not interested in occurrences that involve injury to persons who dismantle a decommissioned aircraft. The purpose of ICAO is to set international rules for civil aviation. This includes aviation accident investigation. Since decommissioned aircraft are no longer operated in a civil aviation environment they can be excluded from the scope of interest. Note also that there has to be a certain level of loss. Light injury to a person associated with the operation of an aircraft does not constitute an Accident. There is a practical reason to require a minimum of harm for an ocurrence to be called an Accident. For ICAO member states an Accident means work and there are only finite resources to do that work. The efforts of analysts are best spent on worthwhile Accidents, those that have caused significant harm. Another reason is, that a low threshold means that large numbers of Accidents show up in statistics. The reduction of Accident numbers should mirror the increase in flight safety.

Man-Made Systems In the introduction I mentioned different types of occurrences. I simply wrote that neither the Tohoku quake, nor the tsunami were Accidents. The failure to remove excess heat however is an Accident. The reason to exclude the natural disasters from the list of occurrences that may be Accidents is that natural disasters differ in one very important way from nuclear power plants. Power plants are man-made, but natural disastersarenot. If a nuclear Accident is analyzed and understood, changes can be made to other similar plants to prevent future Accidents. But we cannot change the design of tectonic plates of fluid body physics, which does not mean that we should not try to understand them. Both natural disasters were causes for the nuclear Accident and understanding their behavior can also lead to insights to improve the design of nuclear plants.

Our own working definition of Accident will now be:

Accident *An Accident is an occurrence that results in very significant loss.*

For pragmatic reason the ICAO defined minimum level of loss for an occurrence to be an Accident. We do not want to go into specifics, that are best dealt with by domain experts, who will ultimately responsible for analysing Accidents. The phrase *very significant loss* will stand in to remind us that there should be threshold that divides the Accidents from the non-Accidents. Another pragmatic consideration is, to allow for multiple levels of loss, so that responses can be tailored to occurrences.

Incident An Incident is an unwanted occurrence that results in significant loss.

Within this definition all Accidents are also Incidents, but not the other way round. Intuitively we could say that Incidents are near-Accidents.

Something went wrong so that an analysis is warranted, but no *very significant* amount of loss resulted from the Incident.

Let's have a look at ICAO's definition of Incident.

Incident - ICAO *"An occurrence, other than an accident, associated with the operation of an aircraft which affects or could affect the safety of operation."*

It is defined relative to the definition of Accident, but here Accidents are not Incidents. They have been explicitly excluded. Both ways are OK if they fit their purpose.

I chose to define Accidents to be a subset of Incidents so that I would not have to write "Accidents and Incidents" throughout this book. I prefer to just write "Incidents" where I mean both, which will be most of the time.

Causality

The Tohoku earthquake caused the tsunami and both contributed to the nuclear crisis. Did both cause the crisis? Intuitively we would say, Yes they did. But when I rephrase the above statement to "The Tohoku earthquake caused the tsunami and both contributed to the nuclear crisis" then the phrasing implies that there have been other causes to the nuclear crisis.

One occurrence causing exactly one other occurrence is a special case. Usually a number of causes are necesary to cause one other occurrence. Consider the number of safety systems in your car: anti lock brakes, anti skid, airbags or the crush zone to name a few. For a frontal crash to be fatal a number of these must either be ineffective or inoperative. Each safety system is supposed to increase the number of causes necessary for a fatal frontal crash by one. Think of each safety system a barrier, with barriers lined up to prevent you from reaching the accident. The barriers are not perfect, there are wholes in them, like there are wholes in swiss cheese. Every time you move in the direction of the accident one of the barriers is supposed to stop you. If the first barrier does not stop you, because you went right through one of its holes, then the next one should, and so on. In the unfortunate case that you found only holes where the barriers shold have been, the accident happens. This illustration of barrier failure is sometimes referred to under the name *The Swiss Cheese Model*[49].

The common view on causation is sometimes called a "causal chain", which suggests one-to-one causal relations between causes and effects. For simple Incidents that may be sufficiently explanatory, but if it is accepted as the norm it will stop an analyst looking for other causes. Complex systems tend to have complex cause and effect relations that require a more rigorous approach to causality.

WBA is built upon one such rigorous notion of causality, the basis of which is *counterfactual reasoning.* A counterfactual statement is a statement, that is contrary to fact. It is not true. To determine if one thing is the cause of another, we ask ourselves what would be if a counterfactual statement would in fact be true. Could the bomb have exploded if *I had not pushed the trigger*? The part in *italics* is the counterfactual statement. We imagine it to be true and we also imagine what the consequences would have been for the bomb. If the answer to the question is "NO", the bomb would *not* have been exploded I had not pushed the trigger, then the trigger pushing caused the explosion. This kind of *Would-if-not?* question is called a *counterfactual test.*

Counterfactual Test *If C had not happened, could E have happened, all else being equal?*

In other words, contrary to the fact that C and E both happened, we assume that C would not have happened. In that case can E happen? If the answer is *no*, then C is necessary for E to happen. If the answer is *yes*, then C is not necessary for E to happen, E would have happened anyway.

In WBA speak we say:

Necessary Causal Factor (NFC) *Cause C is a Necessary Causal Factor for effect E if C passes the Counterfactual Test for E. C is an NFC for E.*

Why another word for *cause*? If C is a *cause* of E according to the Counterfactual Test, then why do we need another name for their relationship? First, another concept is to come, that of *sufficiency* (see below). Second, we are interested in factors, a concept that will be introduced later (**??**).

An example: You want to withdraw money from your bank account. This can most easily be done at an ATM[1]. You insert a mag-stripe card into the card slot of the ATM and then enter your PIN[2] and the ATM will dispense the amount you requested. Would you receive the money if you had inserted the card, but would not have entered the PIN? No, the PIN is necessary. Would you receive the money if you had entered the PIN, but would not have inserted the card? No, the card is also necessary. Card and PIN are necessary, but none is sufficient on its own.

I have already presented the Counterfactual Test, and there is a complementary test for sufficiency:

Causal Sufficiency Test *Causes C1, C2, ... and Cn are causally sufficient for effect E to happen if, and only if, E inevitably happens when C1, C2, ... and C3 happen, all else being equal.*

[1]Automated Teller Machine

[2]Personal Identification Number

With both tests, the Counterfactual Test and the Causal Sufficiency Test, it is possible to determine if a suspected cause-effect relation really is one. And if the causation of an effect can be completely determined.

All Else Being Equal The phrase *all else being equal* has a special meaning for the Counterfactual Test and the Causal Sufficiency Test. It means that the statements which are assumed to be true in both tests, are to be minimally invasive. This means for one thing, that only simple counterfactual statements must be used for the Counterfactual Test. It also means, that only the immediate cause-relations must be the subject of the tests.

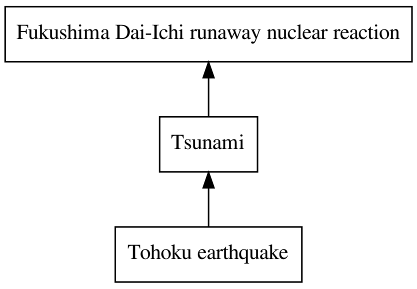


Figure 5.1: Simple causation from Tohoku earthquake to the Fukushima Dai-Ichi nuclear disaster

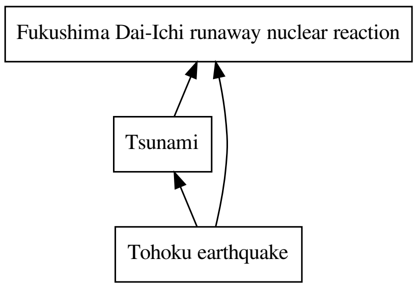


Figure 5.2: The earthquake and the tsunami both contributed to the nuclear disaster.

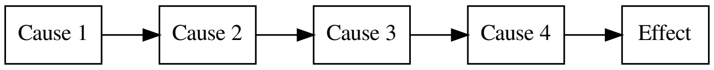


Figure 5.3: Causal chains are too simplistic to explain complex causal relationships.

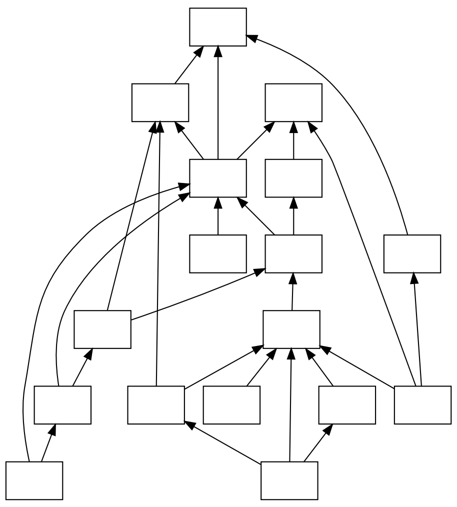


Figure 5.4: Causal relationships can be quite complex. This does not bear any resemblance to a chain.

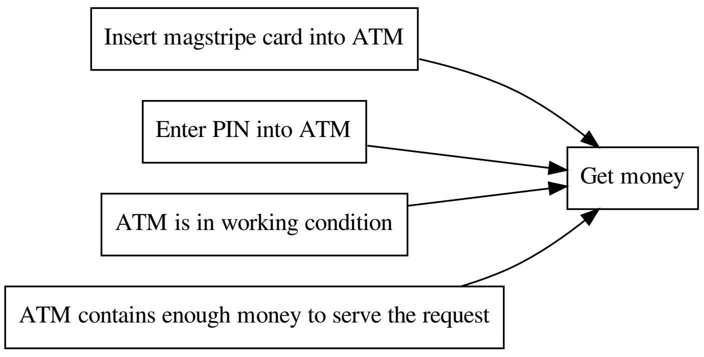

Insert magstripe card into ATM
Enter PIN into ATM
ATM is in working condition
ATM contains enough money to serve the request
Get money

Figure 5.5: Necessary and sufficient causes to get money from an ATM

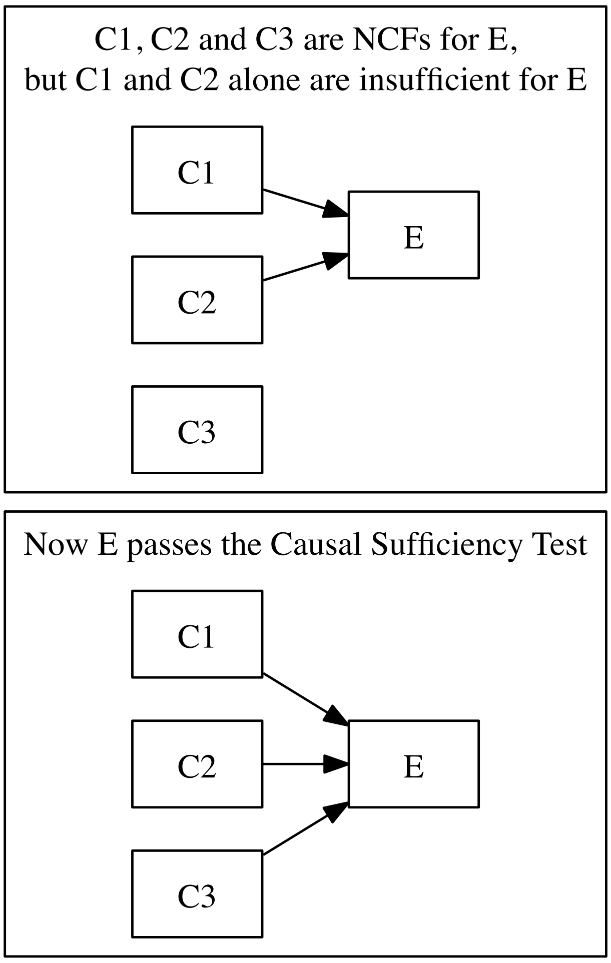

C1, C2 and C3 are NCFs for E,
but C1 and C2 alone are insufficient for E
C1
C2
C3
E
Now E passes the Causal Sufficiency Test
C1
C2
C3
E

Figure 5.6: Sufficient and Insufficient Set of Causes for E

5.2 Counterfactual Reasoning

5.2.1 Seeing it Coming

On July 1st, 2002 two aircraft collided mid-air over Lake Constance in southern Germany. The Accident happened despite the fact that both aircraft were equipped with collision avoidance systems. But before a collision avoidance system alerts the crew a ground based system alerts the air traffic controller. The ground based system, called Short Term Collision Avoidance STCA, sounds an alarm if it detects a possible mid-air collision within approximately two minutes. The air traffic controller then has to instruct the aircraft to keep them sufficiently separated.

The airborne system, the Traffic Collision Avoidance System TCAS only warns the crew a little under 50 seconds ahead. In case of a potential mid-air collision the air traffic controller usually has advance warning and the airborne TCAS is only needed if the air traffic controller cannot keep the aircraft separated.

The air traffic controller at Zurich, Switzerland, did not have the STCA system at his disposal due to maintenance. The radar system, the air traffic controller was using, was running in a degraded mode, but was deemed sufficient for the low density night time traffic, that was expected. Another system was down due to maintenance on the same night: One of the telephone systems used to contact other air traffic controllers in other air control centers.

Air control center Zurich was not the only air control centers whose radar covered the flight path of the two aircraft that were about to collide. The so called upper area control center at Karlsruhe, Germany, was monitoring the situation, but was not responsible for air traffic control in that area. But the STCA in Karlsruhe was fully operational and a controller in Karsruhe tried to phone Zurich three times immediately before the collision. Unfortunately the air traffic controller at Zurich was busy directing an aircraft to the airport Friedrichshafen, Germany. The Zurich controller was not continuously monitoring the situation over Lake Constance.

Please note that the situation was a little bit more complex that outlined here, but this simplified version is better suited for learning the concepts of Counterfactual Reasoning. For the same reason, lets focus on the inability of Karlsruhe to reach Zurich by phone. Please excuse the rather cumbersome statements below, but I will try and keep the form of the Counterfactual Test intact.

- Could the mid-air collision have happened, had the air traffic controller at Zurich been monitoring the situation?

- Could the the air traffic controller at Zurich not have been monitoring the situation, had the air traffic controller at Karlsruhe been able to reach him by phone?

- Could the air traffic controller at Karlsruhe have reached Zurich by phone, had the phone system not been offline for maintenance?

If you read the three Counterfactual Tests another question suggests itself: *Could the mid-air collision have happened, had the phone system not been offline for maintenance?*

5.2.2 Cause-Relation

Intuitively we would answer the last question in the mid-air collision story with No, if we would answer the three preceding Counterfactual Tests with No. The factors we used in the three questions above, in our simplified account of the Lake Constance mid-air collision, are

- **A** - mid-air collision
- **B** - the air traffic controller at Zurich is not monitoring the situation over Lake Constance
- **C** - the air traffic controller at Karlsruhe is not able to reach Zurich by phone
- **D** - part of the phone system at Zurich is offline for maintenance

As the three Counterfactual Tests showed D is an NFC[3] for C. C is an NFC for B. B is an NFC for A.

Is D a NFC for A? No. If that would be that case there would be no need for B and C. Remember the phrase *all else being equal.* If the Counterfactual Test is done for D and A, then B and C must remain what they are! In other cases, which have the same form, D could well be a NFC for A. In a later section I will show the differences between analyzes that differ in their level of detail. It may well be possible to omit detail and suddenly two factors are NCFs, because intermittent factors fall away. For the moment the important thing is to understand, that the phone system's maintenance is a causal factor for the mid-air collision, but not a NFC for the mid-air itself. There is another word that we will use for this type of causal relationship.

Technical Term: Cause-Relation *If two factors F1 and F2 are causally linked by NCFs N1, .., Nn, so that F1 is an NFC for Nn, ... N2 is an NFC for N1 and F2 is a NFC for N1, the Cause-Relation holds for F1 and F2.*

To make things a little simpler,

[3]Remember: Necessary Causal Factor

Technical Term: Cause *If F1 and F2 are either related by the Counterfactual Test or are subject to the Cause-Relation, the we simply say F1 is a Cause for (or caused) F2.*

Not only was our story a simplified version of the real Lake Constance mid-air collision, but our selection of factors for the Counterfactual Tests was also limited. There are other Causes for the collision within the simplified story.

- Would the mid-air collision have happened, had the STCA not been inoperative?
- Would the STCA have been inoperative, had it not been taken offline for maintenance?
- Would the air traffic controller at Zurich not have monitored the situation over Lake Constance continuously, had he not been busy directing incoming traffic for Friedrichshafen airport?

5.2.3 Countermeasures

The main reason for the analysis of Accidents is to improve systems. If a system fails, an Accident analysis is conducted, the results are evaluated and this may lead to the change in the design of the system. This is the way in which Accident analysis can make the world a safer place. Counterfactual reasoning is one way of bridging the gap between an Accident analysis and the decision to change a system's design.

What can we learn from our simplistic story of the Lake Constance mid- air collision? From the Cause-Relation we know that the Acci- dent has many Causes whose disappearance would effect the disappearance of the Accident, or at least alter the course of the Accident significantly. If we want to eliminate this Accident, or to be a bit more ambitious, many similar Accidents, then we have to implement Countermeasures that eliminate one of the Causes of the Accident. There are Causes in our simplified version of the story that we do not have influence on. For example non of the equipment used for air traffic control is maintenance free. We cannot assume that simply not doing maintenance work will solve the problem, simply because the equipment will deteriorate and then the air traffic controllers are again in a similar, if not worse, situation with degraded radar or telephone functionality.

What about a rule that requires maintenance to be conducted on only on one system at a time? In our case the Karlsruhe air traffic controller would have been able to phone the Zurich air traffic controller. This would surely be an improvement, but only for those Accidents where there is a second air traffic controller monitoring neighboring controlled air space. An additional rule requiring the notification of other air traffic controllers on the reduced functionality of an air control center may further improve the odds of preventing another Accident of this type.

If we develop Countermeasures we can use our Accident analysis for a first effectiveness check. From every Accident analysis we can derive a number of Countermeasures. Even from our simplified example we can infer Countermeasures that would also hold in the real thing, given that all the information in the simple story also hold in the real one. Some Countermeasures may be of technical nature, others may be of legal nature. Thus different people can each work on Countermeasures from their field of expertise.

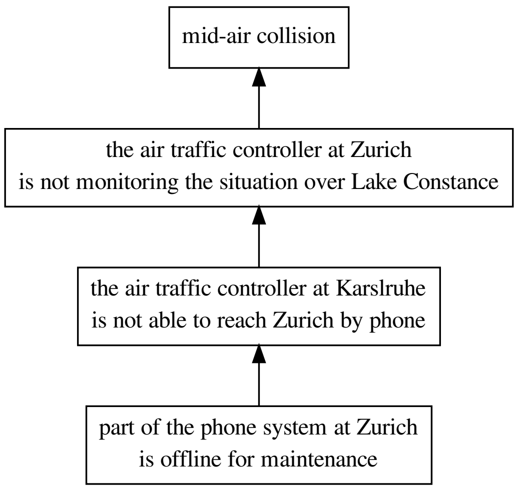

mid-air collision
the air traffic controller at Zurich is not monitoring the situation over Lake Constance
the air traffic controller at Karslruhe is not able to reach Zurich by phone
part of the phone system at Zurich is offline for maintenance

Figure 5.7: Simplistic causation of the Lake Constance Mid Air. Note, that this graph does not fulfill the Causal Sufficiency Test.

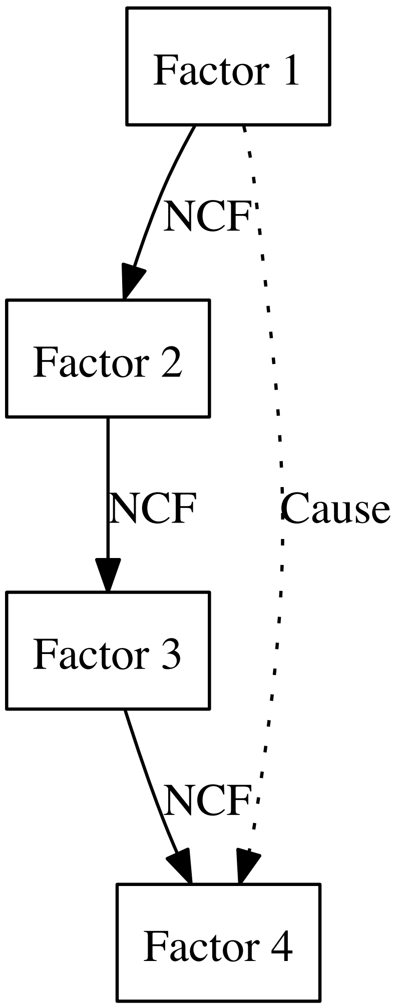


Figure 5.8: Cause-Relation: F1 and F2 are not direcly related as Necesary Causal Factors.

5.3 The First Accident Analysis

If you are using the WBA Software Tool you can find instructions to the topics covered in the following sections at the end of each section. Instuctions are not very detailled. It is assumed that at least the Quick Start Guide was read.

5.3.1 Why-Because Graph (WBG)

In previous sections we have already learned concepts of causal relations. Causal relations are directed, which means that it is important which occurrence is at which end of a causal relation. One is a Cause and the other is an Effect. If we revert the relation it would denote something different.

There have already been some diagrams depicting causal relations using boxes and arrows. The arrows are always pointing from the Cause to the Effect, which is the same direction in which a time-arrow would point. For a complex Accident analysis many such relations exist and it is helpful to visualize complex data and complex relations. We draw boxes for all ocurrences and arrows for all NCF relations in one diagramm then we have a constructed a Why-Because Graph.

The WBG is a directed graph, which is mathmatician-speak for a graph where the arrows[4] have arrow heads on one end. Following the arrows must not lead to circles. This would mean that causality is circular which violates the laws of causality. A WBG is a *non-circular directed graph.*

Other terms that are used:

- A **Node** is a box in the WBG.
- An **Edge** is an arrow in the WBG.
- An **Effect** is the box at the pointy end of an arrow or **Edge**.
- A **Cause** is a box at the stub end of an arrow or **Edge**.
- A **Factor** is nearly the same a **Node**, but is not neccessarily a part of the WBG.

5.3.2 Friendly Fire isn't

For the next sections the following story is used to illustrate how a WBA is done. The story is a retelling of an article that was published in the Washington Post, by Vernon Loeb, a Staff Writer, on March 24th, 2002. It describes a Friendly Fire accident which occurred in Decmber 2001 during Operation Enduring Freedom in Afghanistan.

[4]The mathmatician uses the term Edge, or directed Edge in this case, instead of arrow.

The version given here is a redacted one which does not accurately resemble the events described by Vernon Loeb. The Friendly Fire Accident is used as a first WBA case during System Safety lectures at Bielefeld University and during industrial courses given by Causalis Limited. I do both and so I changed the story to be more study-friendly and less resembling the real story. In an interactive environment many of the unknown concepts can be explained on demand and the lecturer can substitute for a lack of domain knowledge on the side of the students. The main aim is that this introduction to WBA should allow self study, so some trade-offs have been made.

The Setting

The Accident happens during the first months of Operation Enduring Freedom in Afghanistan. U.S. Forces have invaded Afghanistan and combat operations are still on an all-out-war level.

The Accident

A team of U.S. Special Forces soldiers, operating in Afghanistan, were engaging a Taliban position. The U.S. Special Forces soldiers called in an air strike on the Taliban position. The target's GPS coordinates came from a device called a "plugger". The official acronym of the device is PLGR, which is short for Precision Lightweiht GPS Receiver. The plugger can be used to calculate positions for air strike targets. The U.S. Special Forces air controller[5], the person responsible for calling in an air strike, successfully used the plugger to strike the Taliban position. The controller called in a U.S. Navy F/A-18 ground attack aircraft which attacked the Taliban position using GPS guided munitions[6]. A couple of minutes later the U.S. Special Forces intended to strike the same Taliban position a second time. The first calculation of GPS targeting coordinates was done using the GPS minutes-seconds format. An example would be 57 deg 38' 56.83" N, 10 deg 24' 26.79" E, which would mean 57 degrees, 38 minutes, 56 seconds and 83 hundreds of a second northern latitude and 10 degrees, 24 minutes, 26 seconds and 79 hundreds of a second eastern latitude. A different format is the degrees-decimal format. 57.64911 10.40744 would be the same position, but with minutes and seconds denoted as fractions of degrees. The F/A-18 required the minutes-seconds format, but for the second strike a B-52 was tasked with the attack. But the B-52 needed the degrees-decimal format, so some additional calculation was needed. The U.S. Special Forces air controller was doing the calculation, but before transmitting the coordinates to the B-52 crew the plugger's battery died. A replacement battery was

[5]Please do not confuse the air controller with an air traffic controller.

[6]So called JDAM, Joint Direct Attack Munitions

put in the plugger which came back to life. Unknown to the air controller the plugger initializes itself with its own position, rather that the last one displayed or used, which was what the controller was assuming. Not recognizing the difference in coordinates the controller transmitted the displayed coordinates to the B-52. The air strike hit the U.S. position killing three soldiers and injuring 20.

Figure 5.9: Precision Lightweight GPS Receiver, source: Wikipedia

Some Unknowns

It is not clear from the article how exactly the calculation of targeting coordinates is done. The plugger knows its own position. Ways of aquiring the target position could be by using precise maps, by laser designation and range finding or simply by estimation of distance and direction of the target realtive to ones own position.

Also it is unknown which other aids were used by the air controller to help in calculating the target coordinates.

The distance between the U.S. position and the Taliban position is not known, so the number of digits that were different in their respective coordinates is not known.

Why the F/A-18 and the B-52 required different formats for targetting is also not known.

First Things First

Damage What is the damage in the story? Clearly the loss of life and the injuries inflicted to friendlies.

Accident What is the Accident? Which event caused the damage done? Clearly the bomb, dropped by the B-52, caused the casualties.

A quick check using the Counterfactual Test: Had the bomb not been dropped on the U.S. Special Forces Soldiers postion, had the three Soldiers died and 20 others been wounded?

The dropping of the bomb also was sufficient. The bomb did not need the presence of other Factors to do the damage.

Proximate Causes We term all those Causes Proximate Causes that immediately precede the Accident. This is dependent on the level of detail of an analysis. An analysis of a greater level of detail may have other Proximate Causes, that an analysis of a lower level of detail. The simple reason is, that higher detail may describe the same set of events in multiple Factors, that a lower level of detail may subsume in one Factor.

In this case there is only one Accident Factor. This may not always be the case, so do not try to force the structure of your WBG on the presence of only one Accident.

Lets start with the Causal Sufficiency Test, to gather all Factors that enable the bomb to be dropped on the U.S. Special Forces Soldier's position.

- There had to be an aircraft that was capable of GPS guided ground attack.
- The air traffic controller and the air craft had to communicate in order to coordinate strikes.
- The aircraft needed clearance and target coordinates to attack.

When all these three Factors come together then the Accident can happen. We check with the Counterfactual Test:

- Had there been no aircraft equipped with GPS guided munitions, had the bomb been dropped? No.
- Had there been no way of communication between the aircraft and the air controller, had the bomb been dropped? No.
- Had the aircraft not been cleared to target the GPS coordinates received by the air controller, had the bomb benn dropped there? No.

On a side note, observe that the Counterfactual Test reveals two different kinds of questions. The first two questions are whether the bomb had been dropped or not. The last question is about the location the bomb had been dropped.

The intention of the U.S. Special Forces soldiers clearly was the release of a bomb, but on a different location, the latter question is clearly the one leading us to where things went wrong.

What is it called again? Students who work on this case often no not use precise descriptions or names. One example is confusion between the terms *position* and *coordinates.* In the above story the term *Taliban position* denotes the physical location of the Taliban, while the term *coordinates* is used to denote the GPS data. When the original story is read the distinction is not obvious and many students use the terms interchangeably.

There is also inappropriate use of value judgements to describe things. For example some students describe the bombing of the U.S. position as attacking the *wrong coordinates.* But it is not clear in whose frame of mind the coordinates are wrong. The coordinates are valid in that there is a physical location on the earth's surface that corresponds to them. The coordinates are within the B-52s area of operation and the B-52 crew successfully attacks the position belonging to the coordinates given to them.

We will use the following terms from here on:

- *Soldiers* will denote the U.S. Special Forces soldiers as a group.
- *Air Controller* will denote the soldier, member of the Soldiers, who was responsible for calculating targeting coordinates and calling in air strikes.
- *F18* will denote the F/A-18, the aircraft that conduted the first airstrike on the Taliban Position.
- *B52* will denote the B-52 bomer aircraft, that conducted the second airstrike, which hit the Soldier's Position.
- *Taliban Position* will denote the physical location of the Taliban which the Soldiers intended to strike.
- *Soldier's Position* will denote the physical location of the Soldiers.
- *Taliban Coordinates* will denote the GPS coordinates that correspond to the Taliban Position.
- *Soldier's Coordinates* will denote the GPS coordinates that correspond to the Soldier's Position.
- *Displayed Coordinates* will denote the GPS coordinates that appear on the display of the plugger.
- *JDAM* will denote a GPS guided bomb.

Factorizing the Narrative

We already introduced the technical term Factor. We want to take the Accident narrative, our story above, and extract all relevant information so that we can determine the Causes of the Accident. In other words the analysis aims to produce a WBG from the data given in the text. But in almost no case would we be able to take the narrative verbatim. There are many sentences that do not lend themselves to the Counterfactual Test or the Causal Sufficiency Test. Instead we formulate new sentences, new Factors, that preserve the meaning of the original narrative, but are better suited to be tested. In the above paragraphs that describe the Damage for example I have already done that. If you reread the last sentence of the narrative you will find, that there are two factors in the same sentence. In the same sentence the Accident, JDAM dropped on Soldier's Position, and the Damage, 3 dead and 20 wounded, are mentioned.

In general we want aim for each Factor to

- accurately depict data from the narrative (or other sources of Accident data) and
- not be further divisible into meaningful statements.

For example the sentence "I hit the brakes to slow (the car) down." contains three pieces of information. The sentence can be expressed by the three following Factors:

- I intend to slow down.
- I hit the brakes.
- The car slows down.

The "not further divisible" criterion strongly depends on the level of detail. If the level of detail is increased then "The car slows down" could be further explained or substituted by the mechanical interactions from pedal to tires. Another question is if other implied Factors of a statement add insight into the Accident analysed. It would not make much sense to include a Factor like "I am in the car."

Even though we have two rigorous tests, the Counterfactual Test and the Causal Sufficiency Test, there is still plenty of room. As long as the analyst is satisfied with the choice the Counterfactual Test and Causal Sufficiency Test should suffice in assuring the correctness and completeness[7]. If the correctness and/or completeness is challenged consider if the challenge is appropriate by

- checking if the challenge considers one Factor or more,
- estimating the impact on the descriptive value of a new Factor or a changed Factor,

[7]Completeness is very hard to argue. Here I use the term loosely. Someone questioning your choice of Factors will always find something which will violate completeness in a more rigorous sense. Just imagine listing each law of physics that is applicable. It just would not add to the descriptive value of the WBG.

- performing the Counterfactual Test to find out if a Factor is a Cause or not and
- performing the Causal Sufficiency Test to find out if a Factor is indeed missing.

This means to write up an argument for the choice made and refuting claims of incorrectness. This not part of the WBG to be constructed, but it is part of the WBA. This is another reason to be careful with the formulation of Factors and the naming of actors, objects and concepts.

The First Graph

After these first steps the WBG should look like this:

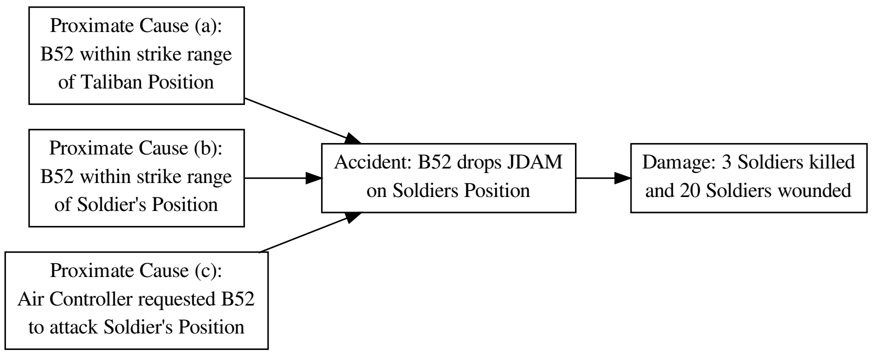

Proximate Cause (a): B52 within strike range of Taliban Position
Proximate Cause (b): B52 within strike range of Soldier's Position
Proximate Cause (c): Air Controller requested B52 to attack Soldier's Position
Accident: B52 drops JDAM on Soldiers Position
Damage: 3 Soldiers killed and 20 Soldiers wounded

Figure 5.10: The first Nodes of the Firendly Fire WBG.

The Node texts are compliant with the non-divisible criterion and also use the terms we defined earlier. But that does not mean that this is the only solution. Feel free to change the statements.

5.4 First Accident, contd.

In the previous section we have seen how to start our work of Accident analysis using WBA. This is not the only way to start, there are probably as many as there are different analysts. But the scope of this introduction is limited, so we cannot explore in depth the pros and cons of different approaches. Be assured, that with each WBA your approach will differ more and more from the one initially presented here. The approach presented here needs to take into account the readers limited knowledge of the concepts of

WBA. As those concepts become clear there are less restrictions on how to conduct an analysis.

In this section we will continue with our analysis of the Friendly Fire Accident from the previous section.

5.4.1 Factors from the Narrative

In the last section we already constructed an initial WBA with a Damage Factor, an Accident Factor and Proximate Cause Factors.

Lets go through the Proximate Causes and see if we can find out their NCFs and discover some other aspects of WBA along the way.

5.4.2 The Stopping Rule

Proximate Causes (a, b): B52 was within strike range of Taliban Position / Soldier's Position There are plenty of Causes why the B52 was where it was, there is little value in further analysis. Operation Enduring Freedom was designed to include combat air support (CAS). Even though we would prevent this accident had there been no CAS available, from a military point of view this would be extremely undesirable.

Stopping because of infeasability of countermeasures The Accident could have been prevented had there been not CAS. No bomb could have been dropped on anybody. But this is also the prohibiting Factor. Ceasing CAS operations would (most probably) endanger more friendlies than it would harm them. The availability of CAS is a Factor of both positive and negative consequences, but the positive consequences largely outweigh the negative ones. The aim of investigating this Accident should not be to eliminate Causes at any cost, but rather to reduce negative consequences.

Stopping because of insufficient explanatory value Every NFC has to be checked by the Counterfactual Test. Some Factors cannot be explained from the Accident narrative alone, but require further research. At some point the cost of further explaining the Causation of some Factors is prohibitive. It is better to stop[8] and concentrate on Factors that offer greater insight if they are further analyzed.

Stopping because of insufficient influence If a Factor is beyond influence, there is little need to further investigate its Causes. Consider the Tohoku earthquake mentioned in the first section. There is no way that earthquakes could be prevented. It is beyond human capability. Within the

[8]At least for the moment. During the course of analysis the cost-benefit-ratio for further research may change.

scope of an Accident analysis there is little value in determining the Causes of such Factors as earthquakes.

5.4.3 Further Analysis

Proximate Cause (c): Air Controller requested B52 to attack Soldier's Position We do not know if the Air Controller was directly communicating with the crew of the B52 or if there were intermediaries. But even if there were intermediaries, for example air combat controllers on board of an AWACS12[9], they would only have relayed the information given by the air controller to the B52. For all practical purposes we can assume that the Air Controller was communicating directly with the B52 crew.

It is also unclear who authorized the strike. If a different party from the Air Controller authorized the strike, there was no way of checking whether the Air Controller had done his job properly. Again we assume that the Air Controller authorized the strike, since another party's authorization could not have relied on better data than the air controller's.

What are the necessary and sufficient Factors for this Proximate Cause?

- The Air Controller's intention was to call in an a second air strike on the Taliban Position.

- The Air Controller (mistakenly) thought that he was giving the Taliban Coordinates to the B52.

Are those NCFs? A quick Counterfactual Test:

- Had the Air Controller requested an air strike on the Soldier's Coordinates if he had not intended to strike the Taliban Position a second time? No.

- Had the Air Controller requested an air strike on the Soldier's Coordinates if he had not (mistakenly) assumed that they were the Taliban Coordinates? No.

And the Causal Sufficiency Test? All the B52 needs to strike a target are GPS coordinates. They were provided with the Soldier's Coordinates. There is no reason why the B52 would not target the coordinates given if requested, as long as they are within the B52's operating area. There is also no reason why the Air Controller would not request an air strike on an assumingly enemy position. The two Factors are sufficient.

Rephrased to fit the phrasing requirements mentioned in the previous section:

[9] Airborne Warning And Control System

- *Cause (a): Air Controller intends to call in air strike on Taliban Position.*
- *Cause (b): Air Controller believes Soldier's Coordinates to be Taliban Coordinates.*

The WBA now looks as like Figure 8:

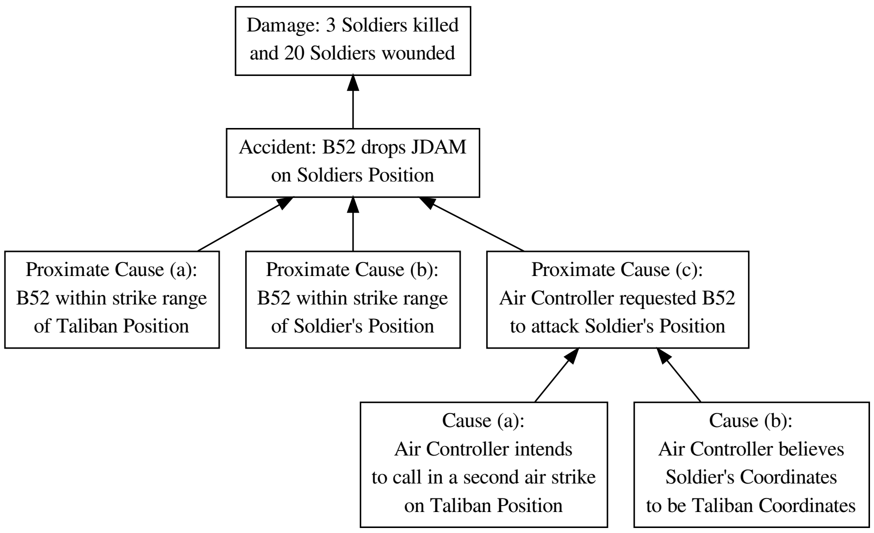

Damage: 3 Soldiers killed and 20 Soldiers wounded
Accident: B52 drops JDAM on Soldiers Position
Proximate Cause (a): B52 within strike range of Taliban Position
Proximate Cause (b): B52 within strike range of Soldier's Position
Proximate Cause (c): Air Controller requested B52 to attack Soldier's Position
Cause (a): Air Controller intends to call in a second air strike on Taliban Position
Cause (b): Air Controller believes Soldier's Coordinates to be Taliban Coordinates

Figure 5.11: Friendly Fire WBG, so far.

Now we again have two Factors that we wish to further investigate. Cause (a) and Cause (b).

Cause (a): Air Controller intends to call in air strike on Taliban Position. The narrative is not explicitly stating the reasons why the Air Controller intended to strike the Taliban Position a second time. But we can reasonably assume that the Soldiers were engaging the Taliban and that the first air strike was insufficient to decide the engagement.

As Factors:

- *Cause (c): Soldiers engaged in combat with Taliban*
- *Cause (d): F18 air strike insufficient to neutralize Taliban*

The Counterfactual Test:

- Had the Soldiers not been engaged with the Taliban, had they intended a second air strike on the Taliban Position? No.

- Had the F18 air strike been sufficiently effective to neutralize the Taliban, had the Soldiers intended a second air strike on the Taliban Position? No.

We include both into the WBG, see Figure 9:

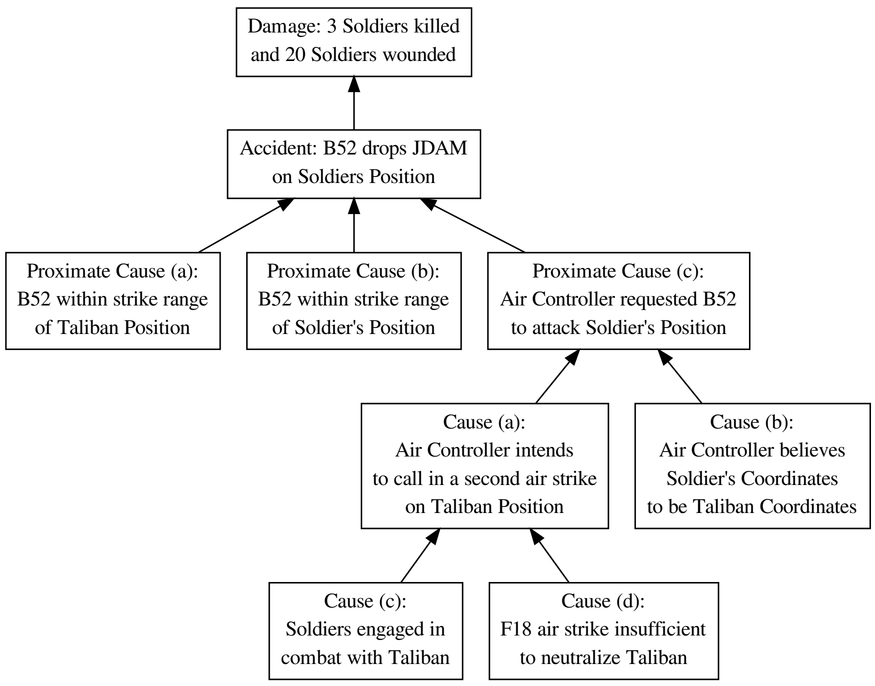

Damage: 3 Soldiers killed and 20 Soldiers wounded
Accident: B52 drops JDAM on Soldiers Position
Proximate Cause (a): B52 within strike range of Taliban Position
Proximate Cause (b): B52 within strike range of Soldier's Position
Proximate Cause (c): Air Controller requested B52 to attack Soldier's Position
Cause (a): Air Controller intends to call in a second air strike on Taliban Position
Cause (b): Air Controller believes Soldier's Coordinates to be Taliban Coordinates
Cause (c): Soldiers engaged in combat with Taliban
Cause (d): F18 air strike insufficient to neutralize Taliban

Figure 5.12: Friendly Fire WBG; Version 3.

Cause (b): : Air Controller believes Displayed Coordinates to be Taliban Coordinates. We continue, breadth first. We could also continue width first, there is no reason why one should be better than the other.

What are the Causes of Cause (b)? The narrative states that the Air Controller used the coordinates the plugger displayed. He believed the coordinates displayed on the plugger to be the Taliban Coordinates for a number of reasons:

- The Air Controller could not tell the Soldier's Coordinates from the Taliban Coordinates.

- The Air Controller was predisposed to believe that the Displayed Coordinates were Taliban Coordinates.

As always we check with the Counterfactual Test:

- Had the Air Controller been able to tell the Taliban Coordinates from the Soldier's Coordinates, had he believed that the Displayed Coordinates were Taliban Coordinates? No.
- Had the Air Controller not been predisposed to believe, had he believed that the Displayed Coordinates were Taliban Coordinates? No.

We introduce our new Factors into the WBG, see Figure 10.

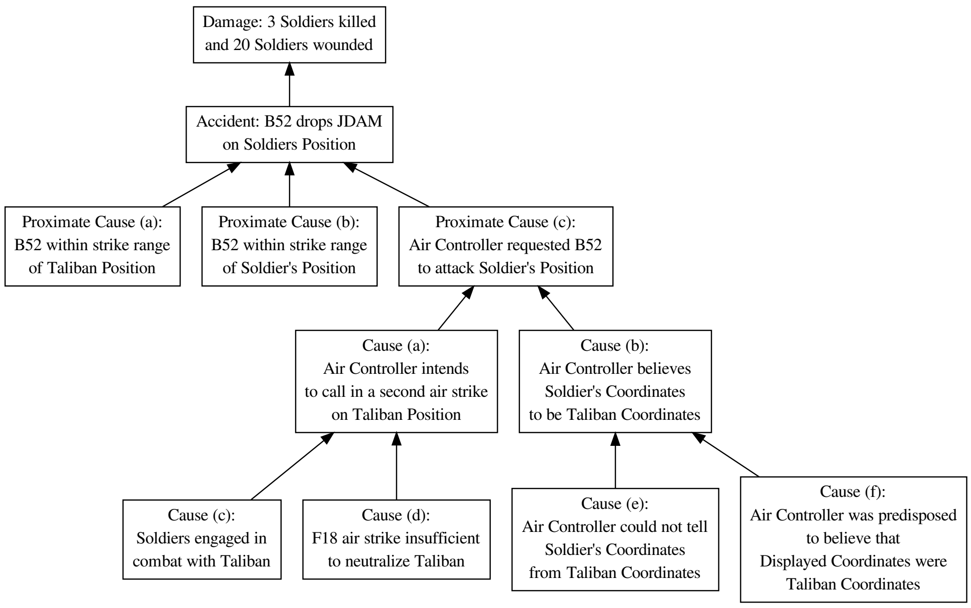

Damage: 3 Soldiers killed and 20 Soldiers wounded
Accident: B52 drops JDAM on Soldiers Position
Proximate Cause (a): B52 within strike range of Taliban Position
Proximate Cause (b): B52 within strike range of Soldier's Position
Proximate Cause (c): Air Controller requested B52 to attack Soldier's Position
Cause (a): Air Controller intends to call in a second air strike on Taliban Position
Cause (b): Air Controller believes Soldier's Coordinates to be Taliban Coordinates
Cause (c): Soldiers engaged in combat with Taliban
Cause (d): F18 air strike insufficient to neutralize Taliban
Cause (e): Air Controller could not tell Soldier's Coordinates from Taliban Coordinates
Cause (f): Air Controller was predisposed to believe that Displayed Coordinates were Taliban Coordinates

Figure 5.13: Friendly Fire WBG; Version 4.

Cause (c):Soldiers engaged in combat with Taliban Following this one up is of little value for our analysis. The soldiers were supposed to take part in combat operations.

5.4.4 Unknowns in the Analysis

Cause (d):F18 air strike insufficient to neutralize Taliban Unfortunately we do not have enough information to completely follow up on Cause (d). We know that there was an air strike, had there not been one it could not have been insufficient:

Cause (g): F18 air strike on Taliban Position.

We do not know the strength of the Taliban, the precision of the strike nor if the bomb performed below its expected effectiveness. From the narrative we know that the strike happened and from the need for a second strike we can infer that the F18 strike was insufficient. But we cannot tell why. Instead of a NFC we will introduce a Node into the WBG clearly marking our lack of information. We know that there was a Cause for this. We have no reason to assume that this unknown Cause is beyond our ability to control, neither that it would not be a valuable insight. Needless to say that this Factor will not be further causally analyzed and will remain a leaf Node, as in Figure 12: *Unknown (a): Strength of Taliban, Performance of JDAM and strike precision are unknown.*

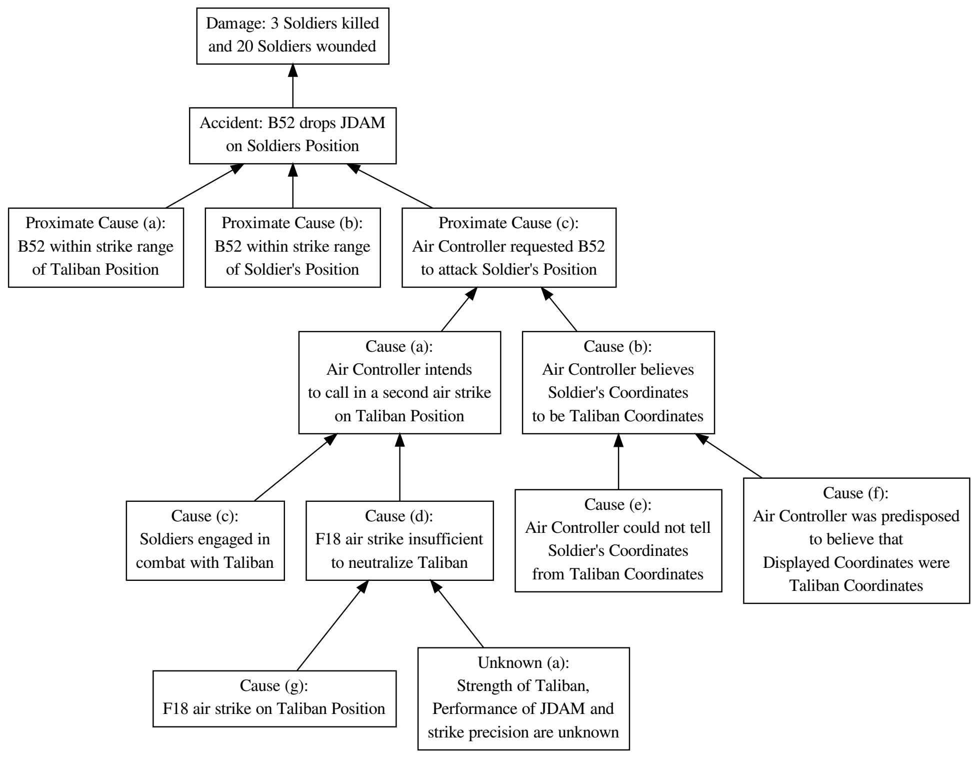

Damage: 3 Soldiers killed and 20 Soldiers wounded
Accident: B52 drops JDAM on Soldiers Position
Proximate Cause (a): B52 within strike range of Taliban Position
Proximate Cause (b): B52 within strike range of Soldier's Position
Proximate Cause (c): Air Controller requested B52 to attack Soldier's Position
Cause (a): Air Controller intends to call in a second air strike on Taliban Position
Cause (b): Air Controller believes Soldier's Coordinates to be Taliban Coordinates
Cause (c): Soldiers engaged in combat with Taliban
Cause (d): F18 air strike insufficient to neutralize Taliban
Cause (e): Air Controller could not tell Soldier's Coordinates from Taliban Coordinates
Cause (f): Air Controller was predisposed to believe that Displayed Coordinates were Taliban Coordinates
Cause (g): F18 air strike on Taliban Position
Unknown (a): Strength of Taliban, Performance of JDAM and strike precision are unknown

Figure 5.14: Friendly Fire WBG with the first indicated Unknown Factor.

5.4.5 Technical Term: Assumption

Cause (e): Air Controller could not tell Soldier's Coordinates from Taliban Coordinates We know that this must have happened, even though the narrative does not explicitly state this. But why did the Air

Controller not recognize the difference? The different coordinates mapped on different locations, so they had to be different. Nevertheless we can safely assume that the visual difference was not that great. Degrees latitude and longitude, the left most digits in GPS coordinates irrespective of format, are the most prominent. They mean the greatest difference in physical location relative the the digits more on the right. If the distance between the Soldier's Position and the Taliban Position is not very great the difference in the sequence of digits may be very small. The Air Controller was using different formats for different calls for air strikes, which may have contributed to his inability to have a clear memory of the Taliban's Coordinates. Please note that, although this may qualify as another Assumption, it is more speculative.

Assumptions are also tested using the Counterfactual Test:

- Had the Taliban Coordinates and the Displayed Coordinates been visually sufficiently different, had the Air Controller been able to tell the Taliban Coordinates from the Displayed Coordinates? Yes[10].

That is why I would not introduce it into the WBG, but leave it here as an explanation for the original Assumption.

We introduce our first Assumption into the 6th version of our WBG, see Figure 13: *Assumption (a): Taliban Coordinates and Displayed Coordinates did not visually differ sufficiently.*

Cause (f): Air Controller was predisposed to believe that Displayed's Coordinates were Taliban Coordinates. According to the narrative the Air Controller was working on the Taliban Coordinates for the second air strike. The Air Controller obviously did not expect the plugger to display different coordinates from the ones entered after a power cycle.

The Counterfactual Test:

- Had the Air Controller not been entering the Taliban Coordinates into the plugger, would he have been predisposed to believe that the Displayed Coordinates were Taliban Coordinates? No.
- Had the Air Controller known that entered coordinates would not survive a power cycle, would he have been predisposed to believe that the Displayed Coordinates were Taliban Coordinates? No.

We introduce two new Causes:

- *Cause (h):Air Controller entered Taliban Coordinates into the plugger.*
- *Cause (i):Air Controller did not know that power cycling the plugger changed Displayed Coordinates.*

[10]To pass the Counterfactual Test this time the answer must be Yes, because the second part of the question was rephrased from "could not tell" to "been able to tell".

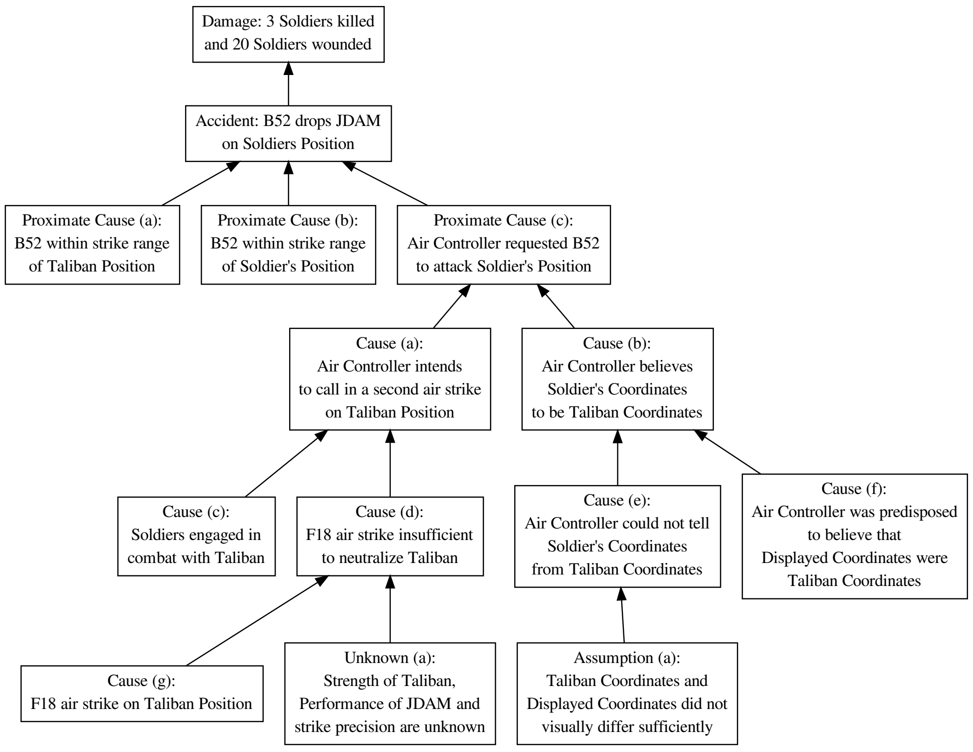

Damage: 3 Soldiers killed and 20 Soldiers wounded
Accident: B52 drops JDAM on Soldiers Position
Proximate Cause (a): B52 within strike range of Taliban Position
Proximate Cause (b): B52 within strike range of Soldier's Position
Proximate Cause (c): Air Controller requested B52 to attack Soldier's Position
Cause (a): Air Controller intends to call in a second air strike on Taliban Position
Cause (b): Air Controller believes Soldier's Coordinates to be Taliban Coordinates
Cause (c): Soldiers engaged in combat with Taliban
Cause (d): F18 air strike insufficient to neutralize Taliban
Cause (e): Air Controller could not tell Soldier's Coordinates from Taliban Coordinates
Cause (f): Air Controller was predisposed to believe that Displayed Coordinates were Taliban Coordinates
Cause (g): F18 air strike on Taliban Position
Unknown (a): Strength of Taliban, Performance of JDAM and strike precision are unknown
Assumption (a): Taliban Coordinates and Displayed Coordinates did not visually differ sufficiently

Figure 5.15: Friendly Fire WBG with Assumption.

Our new WBG looks like Figure 14.

5.4.6 Not a Tree anymore

One of the Causes of our newly introduced Cause (h) is already part of the WBG.

- Had the Air Controller not intended to call in an air strike on the Taliban Position, had he entered the Taliban Coordinates into the plugger? No.

Cause (a) is already part of the WBG. All we need to do is add an additional Edge, giving the WBG its new form, as depicted in Figure 15.

Cause (i):Air Controller did not know that power cycling the plugger changed Displayed Coordinates. No information is given in the narrative why this would be the case. Following this Factor would only lead

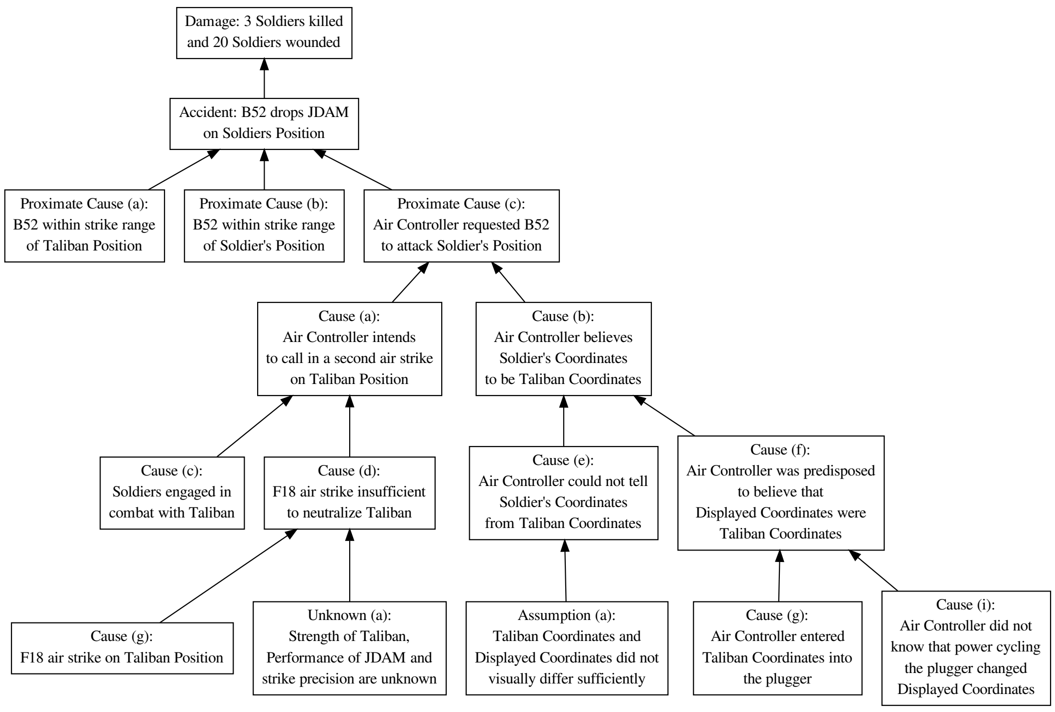

Damage: 3 Soldiers killed and 20 Soldiers wounded
Accident: B52 drops JDAM on Soldiers Position
Proximate Cause (a): B52 within strike range of Taliban Position
Proximate Cause (b): B52 within strike range of Soldier's Position
Proximate Cause (c): Air Controller requested B52 to attack Soldier's Position
Cause (a): Air Controller intends to call in a second air strike on Taliban Position
Cause (b): Air Controller believes Soldier's Coordinates to be Taliban Coordinates
Cause (c): Soldiers engaged in combat with Taliban
Cause (d): F18 air strike insufficient to neutralize Taliban
Cause (e): Air Controller could not tell Soldier's Coordinates from Taliban Coordinates
Cause (f): Air Controller was predisposed to believe that Displayed Coordinates were Taliban Coordinates
Cause (g): F18 air strike on Taliban Position
Unknown (a): Strength of Taliban, Performance of JDAM and strike precision are unknown
Assumption (a): Taliban Coordinates and Displayed Coordinates did not visually differ sufficiently
Cause (g): Air Controller entered Taliban Coordinates into the plugger
Cause (i): Air Controller did not know that power cycling the plugger changed Displayed Coordinates

Figure 5.16: Introducing Causes (h) and (i).

to speculation, which is what happened in the original narrative by Vernon Loeb. Insufficient training and stress during battle were candidates for Causes for Cause (i). In an ongoing analysis this indicates where further investigation is warranted. The Causes of Cause (i) could provide valuable insight into the way the plugger is used by soldier in the field and it could provide a good point for introducing Countermeasures.

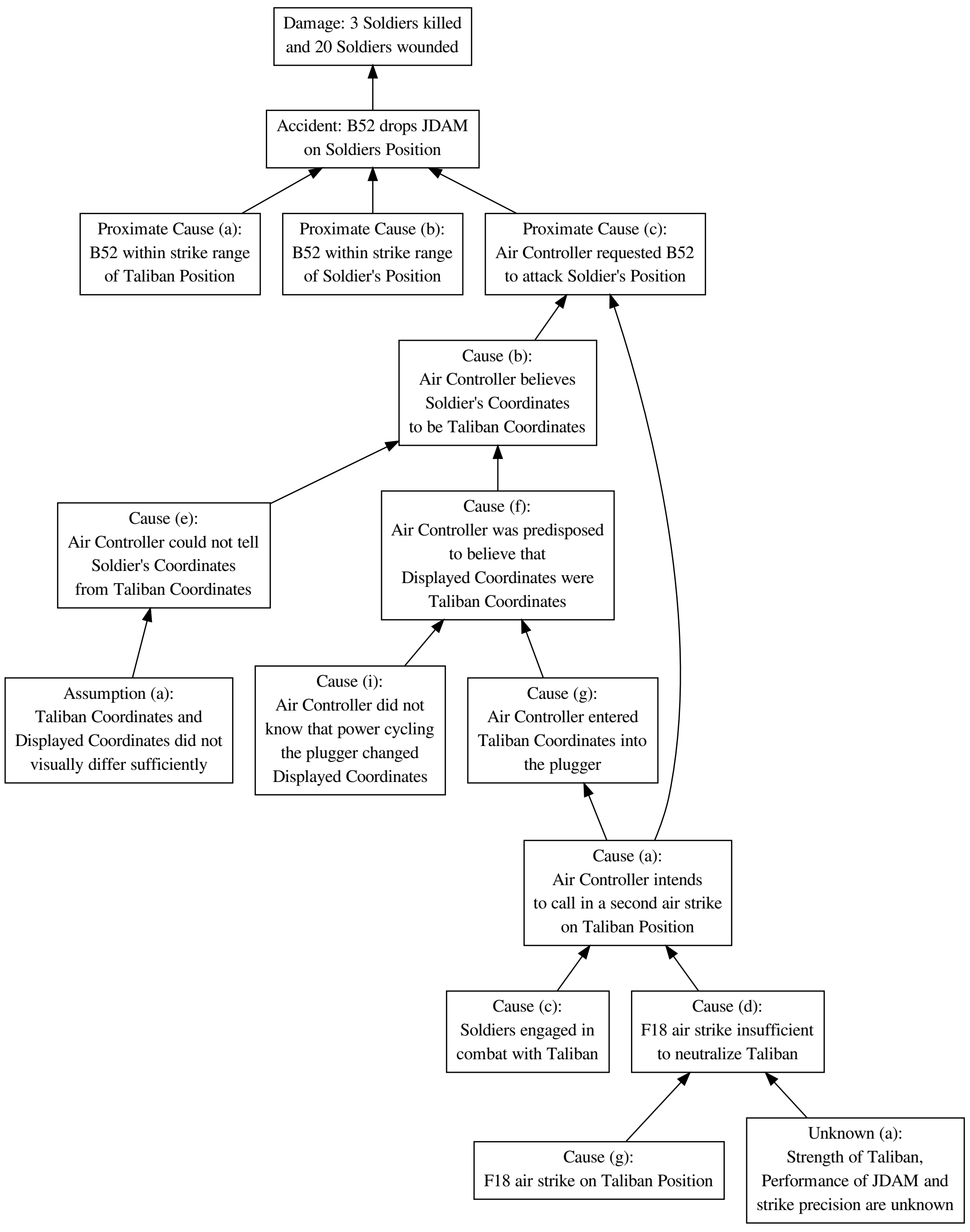

Damage: 3 Soldiers killed and 20 Soldiers wounded
Accident: B52 drops JDAM on Soldiers Position
Proximate Cause (a): B52 within strike range of Taliban Position
Proximate Cause (b): B52 within strike range of Soldier's Position
Proximate Cause (c): Air Controller requested B52 to attack Soldier's Position
Cause (b): Air Controller believes Soldier's Coordinates to be Taliban Coordinates
Cause (e): Air Controller could not tell Soldier's Coordinates from Taliban Coordinates
Cause (f): Air Controller was predisposed to believe that Displayed Coordinates were Taliban Coordinates
Assumption (a): Taliban Coordinates and Displayed Coordinates did not visually differ sufficiently
Cause (i): Air Controller did not know that power cycling the plugger changed Displayed Coordinates
Cause (g): Air Controller entered Taliban Coordinates into the plugger
Cause (a): Air Controller intends to call in a second air strike on Taliban Position
Cause (c): Soldiers engaged in combat with Taliban
Cause (d): F18 air strike insufficient to neutralize Taliban
Cause (g): F18 air strike on Taliban Position
Unknown (a): Strength of Taliban, Performance of JDAM and strike precision are unknown

Figure 5.17: The WBG looses its tree structure.

5.5 Sufficient, but Not Neccessary

There are cases where one Factor is a Cause, but does not pass the Counterfactual Test. Consider the following: Two stone throwers destroy a window. Both are throwing their stones simultaneously. Had stone thrower A not thrown his stone, would the window have been destroyed? Yes, because stone thrower B would have thrown a stone to destroy the window. And vice versa.

Both Factors are sufficient in themselves. One stone thrower alone would be enough to destroy the window. Indeed, one stone thrower would also pass the Counterfactual Test, were it not for the other stone thrower.

5.5.1 Similar Factors

In the above example both Factors were very similar. The action describted for both stone throwers is essentially the same. The difference is only that there are different persons throwing stones.

We need not make the life of the analyst unneccessarily difficult. In this case we can simply combine the two factors into one. This violates the general rule, that Factors should not be further divisible[11]. But it is justified here, for the following simple reason: One of the aims of the analysis is to develop Countermeasures. In this case the two Factors are so similar that we can reasonably expect them to be handled by the same Countermeasure. In most cases both Factors may be a result of a so called common Cause, which in turn suggests that a common Countermeasure may fix the problem. Installing tempered glass may be an effective Countermeasure to both thrown stones. Improving riot prevention may be an effective Countermeasute to the emergence of stone throwers.

5.5.2 Dissimilar Factors

Let's consider a different story with not-so-similar Factors.

This one is a little artificial. There is still a stone thrower who wants to destroy the window. The same instant that the stone thrower throws the stone an earthquake happens. The earthquake is sufficiently powerful to destroy the window.

Now we have two events, dissimilar, without a common cause, both of which destroy the window. Installing tempered glass still counters the stone thrower. It may even prevent the glass from shattering during an earthquake. But there is not Countermeasure that will prevent both Factors from happening.

Aggregating these two Factors into one only complicates things. The

[11]As with every good rule there is an exception.

cleanest way[12] would be to make two WBGs. One analysing the stone thrower scenario and one analysing the earthqake scenario. This assumes that both Factors are not dependend on one another and do not share a common cause. If they do both WBAs will reveal that the independence assumption did not hold.

But making two WBAs is more work that one. To cope with this, rather strange and in my experience very rare, situation, we introduce a way to state the fact that there are two Causes that are sufficient, but in this very situation, not neccessary. Keep in mind that this is a substitute for having two WBGs and that the Counterfactual Test and Causal Sufficiency Test should hold for the virtual WBGs.

[12]From an academic point of view

Chapter 6

Accident Analysis Methods - An Overview

For this thesis the accident analysis method "Why-Because Analysis" was not exactly chosen to be the method used to solve the problem of automating common cause analysis of complex incidents and accidents. It was the other way round. Over the years WBA has seen some significant progress, especially in computer support for the method. After there were a couple of software products to help with graph drawing, soon a common data exchange format was needed. CausalML filled that role. On another front the RVS was experimenting with quality control measures. One approach was to have several groups analyse the same case using WBA and then compare the results. There were some very basic rules how to do that [37] [33] and it was a bit of an effort in record keeping, but quite effective. Students who had just been introduced to the method were capable to produce quality WBA graphs just by comparing results and a small set guiding rules. A new software was written [45] to help analysts compare two graphs with each other.

Then in 2007 an A320 overshot the runway of Congonhas-Sao Paulo. One of the causes was an inoperative thrust reverser. There is one thrust reverser on each engine, but in this case only one was working. How the inoperative thrust reverser contributed to the runway excursion of flight TAM3054 is a rather complex story. The crew knew that one of their thrust reversers was inoperative, indeed they knew it before they departed from Porto Alegre. In between the pilots and the engines and their respective thrust reversers is a lot of automation.

There had been two other cases of A320s with inoperative thrust reversers leading to so called runway excursions in which the pilots also knew of the fact. This already means that some factors of all three accidents were the same:

- The accidents happened to A320 aircraft.

- In all accidents one thrust reverser was inoperative and the other one was working properly.
- The crews all knew of the one inoperative thrust reverser of their aircraft.
- All three accidents resulted in runway excursions.

One question in all three accidents was: "Did the design of the aircraft, including automation and operating procedures, unnecessarily provoke runway excursions given the initial situation?" Some may argue *YES* simply from the fact, that the initial situation and the outcome were the same for all three accidents. Aircraft with one thrust reverser inoperable are still considered safe to fly. If the initial situation would be considered unnecessarily dangerous, then authorities would have to issue rules and recommendations that prohibit flight with one thrust reverser inoperative.

Such rules would affect all aircraft, and the operators of aircraft other than A320 would protest, that operations were safe in the past, for other types. This would clearly put an unnecessary burden on aircraft with good safety records when it comes to flying with one thrust reverser inoperative. Would such rules affect only A320 type aircraft, then there has to be a justification why this type has been singled out. The statistics may speak against the A320, but that does not necessarily mean that the A320 is unsafe flying with one thrust reverser inoperative. A320s safely fly and land with one thrust reverser inoperative, so this alone cannot be a sufficient causal factor for runway excursions.

The question is, which other factors have to come together in order to lead to a runway excursion. If in all three accidents the other contributing factors[1]ontributing in the sense, that they are sufficient for the runway excursion to occur together with the one inoperable thrust reverser. turn out to be A320 specific, then A320-specific action is warranted. If they turn out not to be A320-specific, e.g. pilots are tired or environmental conditions were precarious, then no A320-specific action should be taken.

It should also be noted that the presence of same factors in a number of accidents alone is not enough for accidents to be similar. The causal relations should also correspond. With WBA comparison and correlation of factors and their causal relations across accident Why-Because Graphs was relatively easy. The work was done manually, though there already existed one piece of software trying to automate the comparison between two graphs [41].

Form this point on, automatic comparison of a large number of Why-Because Graphs seemed within reach, so I began to devise a way of automated comparison, build on the experience with WBA so far.

[1]C

So far, other accident analysis methods have not been assessed, if there are ways to automatically extract common causes from a number of accidents. The following is a brief overview of accident analysis methods. The methods are given a short introduction, their main ideas and structuring elements are described and then their potential for automatic data extraction from pools of accident data is assessed. Throughout it will be assumed that there exist data formats for these methods, so that tool-availability is not an issue in this assessment.

The methods have been compared with each other on other occasions, especially mention worthy are the Bieleschweig Workshops [31]. Over the course of Bieleschweig Workshops number 2 and 3, members of academia and industry undertook a comparison of methods to assess their respective strengths and weaknesses. The workshop participants agreed on a number of prominent accidents from different domains and each accident should be analysed with each of the selected methods. There were analyses done using methods that were not designed to be accident analysis methods, e.g. Petri-Nets or Fault-Trees. These methods will not be described here, because they lack application specific semantics. It would also be more appropriate to refer to such analyses not as "Accident Analysis using Petri-Nets / Fault-Tree", but instead to "Accident Analysis using structural elements of Petri-Nets / Fault-Trees".

The methods analysed which were designed to be accident analysis methods were:

- STAMP
- MES
- SOL
- WBA
- AcciMaps
- ECF

These methods all feature different approaches to analysing accidents. In this chapter we will have a look at each of the methods:

- What is the general approach to accident analysis in these methods?
- How is data, especially results, structured?
- Which principles of these methods allow for conclusions or inferences that could be used in automatic accident analysis?

Tool support or the level of standardisation of data structures will not be an issue.

6.1 STAMP

In [38] Nancy Leveson, the inventor of STAMP, describes it as follows:

"The[.] STAMP (Systems-Theoretic Accident Model and Process) model of accident causation is built on these three basic concepts - safety constraints, a hierarchical safety control structure, and process models - along with basic systems theory concepts.[...] In STAMP, systems are viewed as interrelated components kept in a state of dynamic equilibrium by feedback control loops. Systems are not treated as static but as dynamic processes[...]. Accidents are the result of flawed processes involving interactions among people, societal and organisational structures, engineering activities, and physical system components that lead to violating the system safety constraints. The process leading up to an accident is described in STAMP in terms of an adaptive feedback function that fails to maintain safety as system performance changes over time to meet a complex set of goals."

In [38] the author also describes the process of how to apply the model STAMP to actual accident analysis, a process called CAST, the "Causal Analysis based on STAMP". For the purpose of this thesis, CAST is not of importance, since all analysis is done within the STAMP framework. CAST's main purpose is to define a way to get data into STAMP so that the object of interest can be analysed.

In STAMP an accident is not the object of analysis per se. A STAMP model of the environment in which an accident happened is described in terms of control-feedback-loops, which are hierarchically structured. For example: physical systems are controlled by operators or computers. The state of the physical systems is fed into the computers or made available to the operators. Operators and computers in turn have control and feedback relations with other operators, superiors, organisations, etc.

The analysis of an accident is done by examining the narrative of an accident and checking if all necessary control-feedback-loops of the overall system

1. are in place,
2. worked as designed/specified and
3. have been designed/specified appropriately.

Structuring Elements The purpose, in terms of safety, of the control-feedback-loops is to ensure that the system stays within appropriate safety constraints. The accident is analysed in terms of inadequacies or lack of control-feedback-loops.

The STAMP model of the environment, or enclosing system, in which an accident happened can be denoted by a directed graph, where

- the nodes are systems or objects
- control edges point from hierarchically higher objects or systems to hierarchically lower objects
- feedback edges point from hierarchically lower objects or systems to hierarchically higher objects
- there are no cycles in the graph if only feedback OR control edges are followed
- there has to be a control edge path from the hierarchically top-most object or system to all hierarchically lower objects of systems
- there have to be feedback edge paths from all hierarchically lowest objects or systems to the hierarchically top-most object or system

The structure of the control-feedback-relations is relatively easily modelled and also well described in [38]. The nature of control-feedback-loop-inadequacy is not canonised, which means that the analyst has to deliver an explanation whenever he-she thinks that a loop is inadequate. Also, an explanation is needed to argue for missing control-feedback-loops. While the model asserts that control an feedback always have to go together, there is no canonical way to determine when a system or object should be inserted or eliminated from the structure in order to better maintain the safety constraints.

Automatic Analysis Using STAMP, the accident itself is not analysed within the model, except for the explanations given why the system in which the accident happed was inadequate to prevent it. So, analysing lots of accidents from the same domain will not result in different models, at least insofar as they do not change their topology, but only in the analyses which report the systems inadequacies with regard to an accident.

In order to determine if a number of CAST reports share common causes, the explanations of control-feedback-loop inadequacies must be machine interpretable. Unfortunately, these explanations can consist of fairly long and technical [44] plain text.

Maybe some work could be put into categorising control-feedback-loop inadequacies.

6.2 MES

Multilinear Events Sequencing (MES) [2], as the name suggests, organises events into sequences. The sequences are ordered chronologically, though the method does not explicitly call for modelling of durations or time in

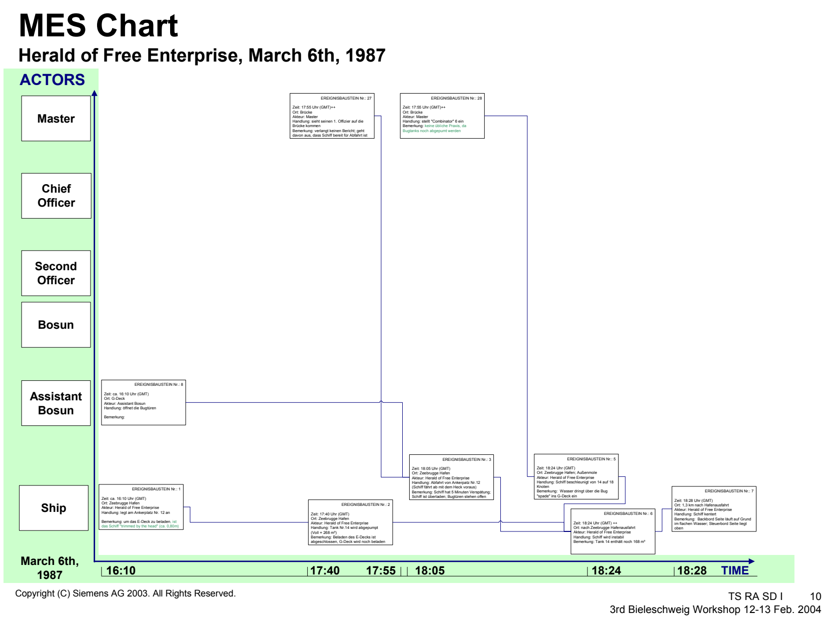

MES Chart
Herald of Free Enterprise, March 6th, 1987
ACTORS
Master
Chief Officer
Second Officer
Bosun
Assistant Bosun
Ship
March 6th, 1987
16:10
17:40
17:55
18:05
18:24
18:28
TIME
Copyright (C) Siemens AG 2003. All Rights Reserved.
TS RA SD I 10
3rd Bieleschweig Workshop 12-13 Feb. 2004

Figure 6.1: Taken from [30]

between events. *Multilinear* means that there is more than one sequence of chronologically ordered events, except for trivial cases. Each event, or *Event Building Block (EBB)*, as an event's description in MES is called, is associated with an actor. For each actor that has been identified, a chronologically ordered sequence of EBBs is constructed. The chronological order of EBBs has to be preserved across all sequences. Additionally the cause-effect-relations between EBBs are included in MES.

According to [2] EBBs should be stated in active voice and EBBs should not denote omissions, unless there are very strong indications that something should have happened. For example, a rule may state *X must follow Y*, but did not, then an omission is in order.

Raising the bar too high to include omissions increases the difficulty to check if the cause-effect-relations between EBBs are correct and complete. The method does not explicitly introduce formal causal reasoning about EBBs, but even if WBA's *Counterfacutal Test* and *Causal Sufficiency Test* were to be applied, the lack of omissions would result in too many failures in the *Causal Sufficiency Test.*

The overall topological structure of EBBs and cause-effect-relations, called "logical links" in [16], is similar to a Why-Because Graph. [2] also calls for simplicity in describing EBBs, so EBBs can be compared to nodes and factors of Why-Because Graphs. The "logic links" between EBBs are comparable to the *Necessary Causal Factor Relation* between factors in Why-Because Graphs.

Missing is the rigour of WBA concerning causal reasoning. This means that MES results could be processed in the same way that Why-Because Graphs can be compared topologically. Computing the MES equivalent of a Causal Commonality Graph may well be a good indicator, that real causal commonalities have been found. But unless the semantics of "logic links" are clear and less open to expert interpretation, drawing conclusions from resulting MES Causal Commonalities Charts still requires additional expert opinion and a review of the cases which allegedly share the Causal Commonalities Chart.

6.3 SOL

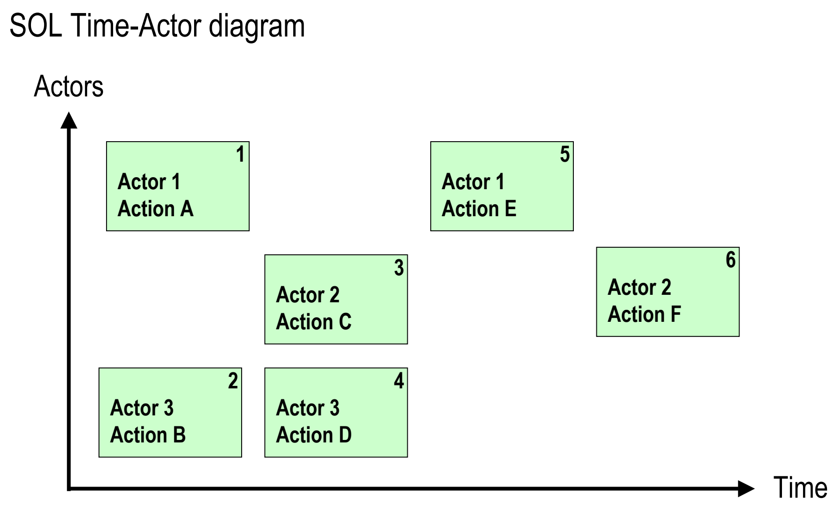
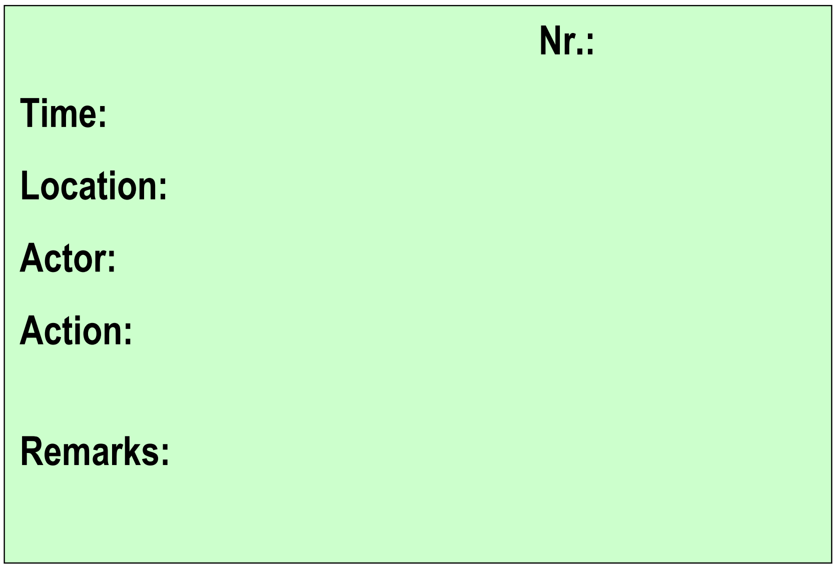


Figure 6.2: SOL Time Actor Diagram: The Time Actor Diagram is very similar in structure to the diagram made for MES analysis.

Safety through Organisational Learning [15] is an accident analysis method that shares many features with MES. Event organisation in multiple event sequences is done in SOL as it is done in MES. The main differences are the data on EBBs and SOL's *identification and analysis aids.*

EBBs Event Building Blocks in SOL are not like factors in WBA or MES, where descriptions of an event should be atomic and simple. EBBs in SOL are more structured, more like a form. SOL's EBBs should have data on:

- A number or ID,
- the date and time the event happened,
- the location where the event happened,



Nr.:

Time:

Location:

Actor:

Action:

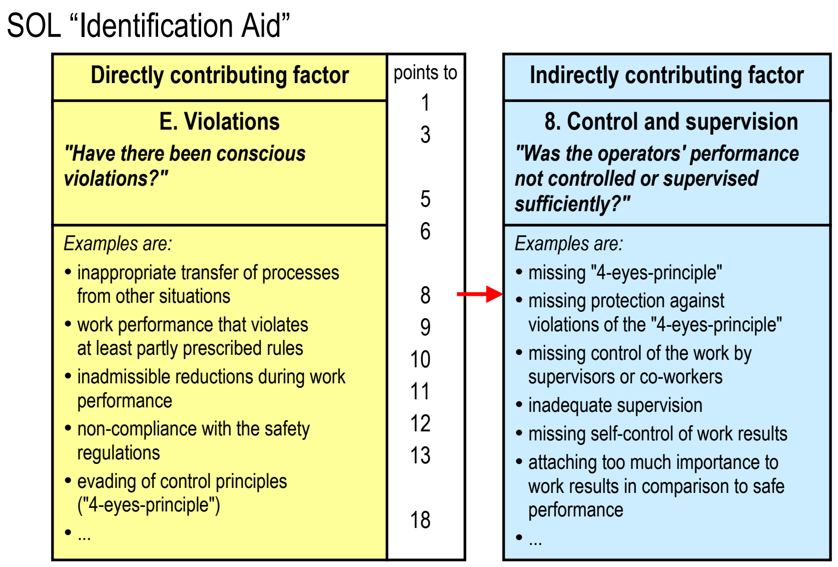
Remarks:

Figure 6.3: SOL Event Building Block: The actual action of the event is described in the "Action" item of the card. "Time", "Location" and "Actor" are difficult to fill out for omissions / non-events.

- actor(s) involved,
- actions involved and finally,
- remarks.

More structured and form-like data makes automatic data extraction easier. The actual event description is contained in the "action" part of the EBB and is subject to the same limitations as other event descriptions. There is no reason not to use guidelines as in WBA or MES, so that the "action" description can be made machine-interpretable. But SOL provides the analyst with additional guidelines, which should help the analyst fill out the EBB form. Additional guidelines would be needed to bring both approaches together, so that the content of the "action" item in a SOL EBB can be stated in a machine interpretable way.

Identification Aid An *Identification Aid* in SOL is meant to help in evaluating an EBB. The *Identification Aid* has two parts. The first part is a list of guiding questions that may be applicable to an EBB. A guiding question may be "Have there been conscious [rule] violations?" The identification aid gives some examples, to help the analyst decide if the guiding question is applicable. If the guiding question is applicable, the *Identification Aid* lists a number of *pointers* to direct the analyst to a list of possible contribut-



SOL “Identification Aid”

Directly contributing factor

E. Violations

"Have there been conscious violations?"

Examples are:

- inappropriate transfer of processes from other situations
- work performance that violates at least partly prescribed rules
- inadmissible reductions during work performance
- non-compliance with the safety regulations
- evading of control principles ("4-eyes-principle")
- ...

points to

1
3
5
6
8
9
10
11
12
13
18

Indirectly contributing factor

8. Control and supervision

"Was the operators' performance not controlled or supervised sufficiently?"

Examples are:

- missing "4-eyes-principle"
- missing protection against violations of the "4-eyes-principle"
- missing control of the work by supervisors or co-workers
- inadequate supervision
- missing self-control of work results
- attaching too much importance to work results in comparison to safe performance
- ...

Figure 6.4: SOL Identification Aid: The Identification Aid supports the analyst to find additional factors. They can be adapted to fit different application domains.

ing factors to the event. An example of a contributing factor to the rule violation guiding question would be

Control and Supervision: Was the operators' performance not controlled or supervised sufficiently?

Again, there are a number of examples given to help the analyst in deciding if a contributing factor applies to an event.

Compared with WBA or MES events are given more meta data in SOL. The description of events using the ”action” item could be changed to be more suitable for machine-interpretation of EBBs. Like in MES there are no rigorous semantics for what constitutes a cause, which has the same implications on deriving causal conclusions automatically. The rigid structure of EBBs in SOL makes it difficult to introduce non-events, especially with regard to stating time, location and also maybe actors.

6.4 WBA and AcciMaps

Why-Because Analysis[54], like SOL and MES, describes accidents in terms of events or factors. Because WBA distinguishes between events, non-events, processes, states and others, *factor* is the preferred umbrella term. WBA places less focus on actor-associated event sequences, but relies on finding and checking causal relations between factors. The relative non-importance of chronological sequences makes introduction of hard to place factors, like non-events or states, easier to integrate.

Factor descriptions should be simple and atomic, like in MES. The causal relations between factors are (5.2.2, p. 38) checked using formal tests, so that there are fixed semantics to the notion of one factor being causal to another factor. Formal semantics make it easier to draw automatic conclusions from given WBA data .

AcciMaps[24] are very similar to WBA [34], but lack the formal notion of causality. Instead factors in AcciMaps are categorised into *physical, organisational, societal and psychological* factors.

As with MES an automatic analysis of AcciMaps would be possible and results would probably good indicators of commonalities in AcciMaps, but the results would have to be checked manually to ensure that they are authoritative.

6.5 ECF

Events and Causal Factors Analysis models accidents with two kinds of factors: Events and Conditions. Factors are very similar to EBBs in MES or SOL or to factors in WBA. Their description should be simple and they should be atomic. Events should denote occurrences and Conditions should denote states or circumstances, but not occurrences. This is more or less the same distinction made between WBA's state- and event-factors. Like WBA ECF makes use of Counterfacutal reasoning to establish if there is a causal relationship between two events. According to [30] causal or Counterfacutal reasoning is not applied to the causal relation between conditions or between a condition and an event.

For automatic analysis this means that at least parts of ECF accidents can be used. The lack of a causal sufficiency criterion leaves much of the quality and (relative) completeness control to the analyst, but there is no reason why ECF could not be improved by additional formal checks. Applying Counterfacutal reasoning to other factors than events, e.g. states/conditions or non-events, has been successfully done in WBA.

6.6 Conclusion

With the exception of STAMP/CAST accident analysis methods all rely on notions of factors or events and on notions of causal relationships between factors. Classification criteria for factors,

- like factor types, e.g. states, events, omissions,

- like actors or
- like organisational classifications, e.g. psychological, organisational, physical,

are usually either translatable from one method into another or can be added to methods that lack them. Again with the exception of STAMP/CAST the accident analysis methods only differ slightly in their approach and do not contradict each other.

This means that, in principle, all these methods can be enhanced with each others features. While this thesis has its focus on finding automatically common causes in a large accident report corpus, other features offer opportunities for additional data mining.

- Classification of factors into organisational-, physical-, legal or psychological factors allows to statistically evaluate not only the prevalence of certain types of factors, but also prevalence in common causes. From this, more general statements can be made as to whether a system is strong or weak in certain aspects of its operation.
- SOL's analysis aids have been derived from organisational research are may offer the opportunity to further subclass factors meaningfully.

What sets STAMP/CAST apart is the focus on control-feedback-loops and the description of the systems and the environment in which accidents occur. The assumption that STAMP is always the appropriate model for accident analysis also means that systems are best described in terms of hierarchy and control-feedback-loops. The focus on control-feedback deficiencies of systems makes STAMP/CAST not a natural candidate for common-cause analysis. It may very well be the case that STAMP/CAST is biased towards certain types of accident causes. Indeed the bias may well have initially led to the development of STAMP/CAST as it is.

Chapter 7

Past Approaches to Automation of Occurrence Data Comparison

7.1 Early Software to Create and Edit Why-Because Graphs

The first pieces of software that automated part of the WBG generation were scripts[51] that ultimately produced a file that could be rendered by the DOT[21] rendering engine of the Graphviz[17] project. One later edition, ybedit[51] by Bernd Sieker would offer a simple graphical user interface.

Limitations in ybedit were its lack of project management and factor accounting. Factors that were not part of the WBG yet had to be kept in a text file.

In his diploma thesis Thilo Paul-Stueve did a task analysis of the WBG generation[47]. He found that lots of time was wasted keeping the rather complex data of a WBA in several different, non-integrated places in sync.

Jan Paller wrote a software, ybfactor[51] that would do the accounting of factors. Both programs used the CausalML[18] file format, whose development was initiated by Oliver Lemke and was a joint effort by Oliver Lemke, Luke Emmet, Mirco Hilbert, Peter B. Ladkin, Jan Paller, Bernd Sieker, Jï¿½rn Stuphorn and Fergus Toolan. I later contributed the inline documentation for the file format, because different tool interpreted the meaning of data structures differently and incompatibly with each other.

ybfactor and ybedit would both use CausalML as their native data format. Jan E. Hennig developed the distributed archiving system VDAS[51], which was used to improve the data exchange between ybfactor and ybedit.

The overall result improved the WBA process considerably, but was still cumbersome, very difficult to deploy, it would only run on GNU/Linux and BSD systems and the there were many things the user of the programs had to know about the data exchange between ybedit, ybfactor and VDAS.

I then started to write a platform independent tool that would integrate all the tasks that were needed to do a WBA. I was later joined by Jï¿½rn Stuphorn, who added PDF exporting functionality. The software was used for research projects inside the group and at Deutsches Institut fï¿½r Luft und Raumfahrt, for courses in WBA and accident analysis, both at the university and at industrial tutorials and by Causalis Limited, a tech-transfer company that specialises in aviation accident analysis.

7.2 Towards Automatic Common Cause Analysis

7.2.1 Manual Occurrence Data Comparison

There are accidents where expert opinions were widely different from one another. It is a variation of the problem, that different experts identify different causes for an accident. The latter problem is solved by WBA. The correctness tests of WBA[54] [36] (see also 5.1.2 and 5.1.2) assure that the analysis is correct, relatively complete and objective. The question of similarity can only be answered if the underlying analyses of two (or more) accidents are correct, relatively complete and objective. This is achieved by comparing factors and subgraphs of two WBGs of different accident analyses. The comparison is done manually. There is no strict rule for assessing the similarity of two factors in different WBGs in manual comparison, but factors are simple enough (5.3.2) that it is obvious whether two factors are similar or not. If the similarity of factors has been established then the similarity of subgraphs can be also established.

Manual comparison turned out to be a very work intensive task. I have done a manual comparison of three supposedly similar aviation accidents[1]. One of the accidents, Bacolod, was done by me, the other two were done by other analysts using WBA. Comparing each pair of accidents took almost as long as it took to do the actual analysis.

7.2.2 Controlled English for WBA, CE4WBA

Within the scope of a diploma thesis Lars Molske developed a software tool, CE4WBA[41], to support the comparison process. The idea was to use

[1]

- TAM Airlines Flight 3045 overran the runway after landing at Sao Paolo airport on 17th of July 2007.
- Philippine Airlines Flight 137 overran the runway after landing at Bacolod airport on 22nd of March 1998.
- Transasia Airways Flight 536 overran the runway after landing at Taipei-Sung Shan airport on 18th of October 2004.

controlled English to describe the factors involved in WBGs. The grammar and vocabulary could be edited in the software. Then all expressions could be typed in and the software would check them for conformity with the vocabulary and grammar. After completely expressing the content of WBGs using CE4WBA all factors could be checked automatically for similarity.

The approach was tried on the above mentioned aviation accidents. The software was performing as intended, but the approach turned out to be infeasible for a number of reasons. The basic grammar, that was designed by Lars Molske, was simple and easy to use, but lacked significantly in expressiveness. Not all factor descriptions could be expressed using the basic grammar. Since the software allowed the grammar to be extended Bernd Sieker, Jï¿½rn Stuphorn and me used the extension feature, which improved expressiveness. Unfortunately the grammar reached the point where it was very complicated to use. The software did not provide any means of checking whether new rules opened the possibility of expressing the same statement in different terms, which conflicted with automatic comparison. We found that the amount of work that went into assuring that automatic comparison could be done on CE4WBA graphs far exceeded the amount of work that was needed to do the comparison manually.

7.2.3 IQualizeIT

Another tool was developed in parallel to CE4WBA which was intended to aid the comparison of WBGs. IQualizeIT[45] was developed by Damian Nowak within the scope of his diploma theses of which I was one of the two thesis reviewers. Its main motivation was to aid in the process of quality management by comparing two WBGs. Students who were taking the Systems Safety classes at our work group were required to compare results with one another. That meant to find all factors that they had in common and then discuss the remaining factors in order to improve the quality of their results. IQualizeIT could read CausalML, but unfortunately had its own data format for writing, because the format was still a work in progress when Damian Nowak was working on his diploma thesis.

7.2.4 Integrating CE4WBA and IQualizeIT

Originally it was thought to integrate CE4WBA and IQualizeIT. Integration would have meant that a system could be built that could automaitcally compare two WBGs that had been built with the CE4WBA grammar and vocabulary. Unfortunately CE4WBA was too unwieldy to use and the integration was not seen as beneficial anymore.

7.2.5 A New Approach

Instead of using CE4WBA another method had to be developed to improve comparisons. This time not only the comparison of two graphs, which was meant for quality assurance purposes, was the goal, but the comparison of a whole corpus of WBGs. The idea was not only to compare graphs and let the analysts figure out the rest in order to synchronize analyses, but to have a fully automatic system that would be able to identify complex behaviour causing occurrences.

Chapter 8

Requirements for Automatic Common Cause Analysis

The accident investigation process presented here has new capabilities that can not currently be found in existing accident investigation report databases.

8.1 Representing Complex Behaviour in Reports

There is nothing fundamentally wrong with describing complex behaviour in natuaral language, humans can understand it. But machines do not understand natural language very well, which makes natural language, the way it is used in all report corpora currently, inadequate. More formal languages such as CE4WBA[41] (7.2.2) are cumbersome and have a very steep learning curve. They may be adequate for demonstrating a principle, but turned out to be unsuitable for to real life application.

Specifying system behaviour more formally,for example with complex mathematical models, is very desirable for data mining, but undesirable from a useablility point of view. It is also very difficult to assure that a model has sufficient expressiveness. Accident data has to be specifiable in a way that an investigator, who usually is no expert in data representation or abstraction, can do it by himself with the confidence that the data entered will yield the expected results.

There is no way around the fact that investigations will primarily be using natural language during the analysis process. Any process that attempts to automate later data extraction therefore must bridge the gap between human-understandable data representation and machine-understandable data representation. The gap should also be bridgeable by analysts in both ways, also taking into consideration that the work of one analyst must be understandable by another analyst and it must also be comparable. Accident investigators must be able to easily convert the natural language representation of a report into a machine-understandable representation. For meta-analysis analysts must also be able to formulate queries to the database and understand the results delivered as a result of the queries.

Accident investigators cannot reasonably be expected to learn formal languages. On the other hand natural language is not suitable for automatic processing. A trade off has to be found.

8.2 Distinction between Cause and Effect

There must be a clear distinction between cause and effect. In current databases reports have chapters on causes and contributions to an accident. This would, in principle, allow text mining to determine single causes to be part of an accident, but the causes and contributions to an accident are not related to one another.

The complete report text could be mined, and indicating phrases such as "leads to", "because of" or "causes" would indicate cause-effect relationships. The statements found with the indicating factors may also be correlated with statements and phrases found in the report sections listing the causes of an accident. Not all causes would necessarily be cause-effect related to one another, but it would still be better than nothing. More importantly, the cause-effect semantics for each relation found may differ from relation to relation. Without such semantics further causal reasoning is problematic. Different authors may have different notions of cause-effect relation semantics. This has implications on causal reasoning, especially when automated. Take transitivity for example: If A is a cause for B and B is a cause for C, if nothing is known about transitivity it cannot be algorithmically determined if A is a cause for C. (see also 9.1.5)

It is also common practice in accident reports to include findings which did not cause the accident that was investigated, but were anyway of importance for safety. Due to the above mentioned problems of causal reasoning, causes found by text mining may be causes for otherwise interesting behavior, but not for the accident in question.

If two or more causes are part of the same causal chain, one cause being another cause's effect, this cannot be inferred from the structure of a text automatically without sophisticated natural language understanding. To understand more complex relations between the causes and contributions, even for a human reader, a thorough understanding of the accident's history is required.

If the cause-effect relations between two factors can be understood by a report database system, then understanding of arbitrarily complex cause effect relations can be built on that.

Example

Consider two hypothetical occurrences in aviation.

- Crew A loses situational awareness. Upon that realisation the crew struggles to regain situational awareness, which turns out to be quite difficult. The crew is anxious which then leads to bad crew resource management and unnecessarily complicates the task.

- Crew B has an inexperienced flight captain. The captain's leadership abilities are inadequate which results in bad crew resource management. During a flight phase with heavy workload the crew suddenly realises that they have lost situational awareness.

In the first case the loss of situational awareness causes bad crew resource management. In the second case loss of situational awareness is an effect of bad crew resource management. In a normal[1] accident investigation report both would be listed in the conclusions part of a report among the causes and contributions. Text mining would not be capable distinguishing between accident investigation reports where A caused B from those were B caused A.

8.3 Distinction between Causes and non-Causes

Another problem for automatic causal reasoning is that accident investigators very seldom use explicit criteria[2] for determining if a factor found is indeed a cause of an accident. This is a shortcoming of the analysis process that generates the reports. Without such explicit criteria[54][36]

- the determination of a cause-effect relationship between a factor and an accident rests solely on the expertise of the analyst,

- criteria are inconsistent across investigation reports and

- no completeness or correctness checks can be done.

The first has implications on the capabilities that a report database can have. Without an explicit rule stating what is a cause and what not analysts tend to include all things that deviate from normal behaviour as causes or contributions. During the Bieleschweig workshops[3] different methods of analysis were discussed[30][11]. Some of which featured (semi-)formal rules on what to include in an analysis and what not.

The quality of a report database obviously depends on the quality of reports contained in that database. Assuring completeness of query results is especially important for all queries to a report database that are aimed at determining the absence of something.

[1]e.g. an aviation accident report according to [26]

[2]falsifiability criteria in the scientific sense

For example: A number of accidents have been analysed and resulted in a change of a system in order to improve safety. After some time the effectiveness of the safety measure should be statistically evaluated. It is expected that the type of accident happens less often, in an extreme case it is eliminated altogether. A report database that contains reports that have been (formally) checked for completeness and correctness can more confidently searched for the absence of something.

8.4 Defining Classes of *Occurrences* and Searching Databases

The problem of defining classes of *occurrences* is related to the problem of representing complex behaviour. In order to define a class there must be a statement that allows the decision if a given *occurrence* is part of the class or not. This is needed for searching and statistics.

In a search for an *occurrence* report in a database the search pattern is a class description. The search result will contain all *occurrences* that match the class description and none that do not match. Search results and their change over time then can be used as the basis for statistical analysis.

In current databases of *occurrence* reports searches and classification are not unified in the way just described. Part of the classification of *occurrence* reports is done before the time the actual search is performed. At the time a database is developed classifications are agreed upon. Then database fields are used to indicate whether an *occurrence* report belongs to a class or not.

Examples from road safety databases would be "Driving Under Influence", "Speeding" or "Technical Fault". These classes come from experience and are known to happen frequently. In a database search the class attributes can be included in the search pattern, so that search patterns are only constructable from predefined classes or patterns and from full text searches. The rate of change in the number of *occurrences* of a specific class can be monitored, e.g. if DUI has bevome more or less common.

That does not imply that classes cannot be changed when new classes of interest emerge. Databases can be changed and data in databases can be reviewed. The problem is only the amount of work that has to go into reclassifying all reports in a database. If a new class is added to the existing ones then everything must be reviewed and changed accordingly. Some databases may be prohibitively large for that, so that only new occurrences will be classified by the new scheme. But the size of an occurrence database in itself is of benefit, because statistical evaluations on large databases will have more impact that evaluations of small ones.

It would be very beneficial to have both flexible classification for large databases. Classification at the time of the evaluation or search eliminates the need for preset classes, which in turn would eliminate the need for reviews if a new class of interest emerges.

8.5 Common Cause Analysis - Automatically Identifying Clusters of Occurrence-Causing Behaviour

Some *occurrences* may share behaviour that leads to unwanted consequences. But common behaviour is not always apparent beforehand. The identification of emerging common behaviour depends on how high the frequency of newly emerging behaviour is and on the severity. More severe common causes will be easier rememberable. Depending on the method used or the rigidity of the rules of an analysis process, analysts may be operationally blind to emerging behaviour.

Automatic identification of emerging unwanted behaviour offers the possibility of ianalysing low severity *occurrences* (incidents), to prevent high severity *occurrences* (accidents). The behaviour may cause different levels of damage depending on factors outside the system in question. The severity of runway overruns in aviation, e.g. depends on the absence or presence of obstacles. On the 14th of September 1993 a Lufthansa A320 overran the runway of Okecie International Airport near Warsaw, Poland, and hit a hill that was built at the end of the runway.[55] The consequences were 2 fatalities and the loss of the airframe. On the 9th of May 2004 an American Eagle ATR-72 had a runway excursion at Luis Munoz MarÃn International Airport, San Juan, Puerto Rico.[6] The aircraft came to a halt on the grass strip beside the runway and suffered no fatalities, though the airframe was lost. On the 17th of July an TAM Airlines A320 overshot the runway of Congonhas-Sao Paulo International Airport, crossed a road and ran into a warehouse.[12] The aircraft was lost and all 199 on board and 12 people on the ground died.

Runway excursions are a well known phenomenon, as are their causes. But unless the same causes lead to other well known phenomena their role in other, less severe occurrences, remains unknown. Changing technology or organisation may give rise to new causes.

It is widely assumed that the rate of *mishaps* is higher than the rate of *incidents* which in turn is higher than the rate of *accidents*[40]. This is often referred to as the safety or accident pyramid. Newly emerging unwanted behaviour will most likely, but not necessarily, manifest in mishaps or incidents before accidents are caused by it. The analysis of low severity occurrences offers the opportunity to identify unwanted behaviour before accidents happen, this way accidents could be prevented.

One problem is that the amount of *occurrences* that must be analysed grows as the severity threshold is lowered to include more *incidents* and *mishaps* into investigation programs. Another problem is that the first time that emerging unwanted behaviour is seen, it cannot be recognised as such.

It cannot be distinguished from a one-off easily. The second time it is seen the question remains how reliable the investigation process is in relating it to the first time. Clearly automation would be of great help. It eases the burden of analysts having to go through a corpus again and again in order to find past behaviour that matches current behaviour. This may either contribute to an increase in the number of *occurrences* that can be analysed, given a fixed amount of resources, and it may lower the adoption threshold enough.

Chapter 9

Algorithmically Finding Common Causes

The purpose of automatic common cause analysis is to find common causes in a large number of accident representations. This method should be capable of clustering accident representations, so that accidents with common causes can be found "close" to each other and accidents which do not have common causes are "far" from each other. To search for common causes we need a causal representation of accidents.

9.1 Causal Analysis Method

9.1.1 Factors

All analysis methods have factors, which are arranged according to the rules (syntax and semantics) of the respective methods. Factors are more deeply analysed in REF (p. PAGEREF). For the time being all we need to know about factors is that

- Factors describe part of an accident history,
- Factors can be compared for identity and
- Factors can be compared for equality and similarity.

9.1.2 Causality

To be generally applicable a method is needed that is making as few assumptions about its application domain as possible. Some methods, e.g. Ishikawa Diagrams [29] or AcciMaps [23] , have emerged as methods which assume the existence of specific classes of factors into which causes can be grouped, such as organisational causes, psychological causes, economic causes, etc. Strict causal analysis methods only assume that the laws of causality [25][39] hold. They therefore are free from unnecessary assumptions.

There are however causal analysis methods, e.g. Ishikawa Models [29], that assume that cause effect relationships form a tree-like or graph-like topology. In other words, effects can have many causes, but causes can only have one effect, which is obviously not true. Just watch a game of billiard. Each time the a ball simultaneously hits two or more other balls and all change momentum, there are more effects for one cause. Other approaches assume only one "root cause", in other words, that there is only one ultimate reason for an accident.

For a survey of causal analysis methods, see Chapter 11 of [32].

We already established that there is a need to distinguish between causes and effects and causes and non-causes (8.4). We want as few assumptions about the application domain and as few non-causality theory of accident causation[1].

This leaves two established causal analysis methods which are good candidates to form the foundation of the automated common cause analysis.

9.1.3 Hopkins' AcciMaps

Hopkins describes an AcciMap to be a causal diagram[23]. "The AcciMap is used to show how factors quite remote from the immediate accident sequence contributed to the accident." The graphical representation shows an acyclic directed graph, where the nodes denote the factors and the directed edges denote the cause-effect relationship or flow of causality. In AcciMaps factors are categorised into five groups of Societal, Governmental/Regulatory, Company, Organisational and Physical factors. While this violates the requirement of unnecessary assumptions it is completely independent of the the causal aspect of AcciMaps. The additional categorisation could be ignored for the purpose of automatic common cause analysis and be left intact for other purposes of analysis.

9.1.4 Ladkin's Why-Because Analysis

WBA is a causal analysis method [35] which is built on the counterfactual theory of causality as first described by [25] and later expanded by [39]. The graphical representation is similar to AcciMaps, an acyclic directed graph, where the nodes denote the factors and the directed edges denote the cause-effect relationship[2]. In WBA factors are categorised into Events, Non-Events, Processes, States, Assumptions, Countermeasures, Countraindications, and Unsepcified factors. As with AcciMaps the categorisation can

[1]For a counterexample see the accident analysis method CAST in Chapter 11 of [32], which is not causality-based or motivated.

[2]For an in depth introduction to WBA see

remain intact, while it can be ignored for the purpose of automatic common cause analysis.

What sets WBA apart from AcciMaps and other causal analsyis methods is the use of formal completeness and correctness criteria. Cause-effect relationships, denoted by edges in a causal graph, have a very specific meaning. In WBA the counterfactural relation [36] between two factors is explicitly established by the Counterfactual Test (see below 9.1.5). This is not only an important quality control in WBA, but has positive implications on representing accident data.

9.1.5 Semantics of Causal Data Representation

AcciMaps offer a bit more informal way of conducting occurrence analysis than WBA. That means results of AcciMaps tend to differ more between authors than results from WBA. The more formal semantics of Why-Because Graphs allows for better semantics in data formats and algorithmic processing. In contrast, non-rigid defined cause-effect relationships, as they are used in AcciMaps, would result in non-rigid defined relationships in data formats. Which means that the comparison of two non-identical syntactical elements, such as edges in a graph, will have different meanings, but would be equal according to syntax.

The causal semantics of WBA, and the semantically well-defined syntax of WBGs, do not impose assumptions other than that systems described using WBGs adhere to the laws of causality[3]. WBAs semantics have a range of benefits[54] [36]. The focus here is to develop algorithms which compare properties in different WBGs. The algorithms are designed to deliver results which also have well-defined semantics, which are based on the semantics of WBA. Important for determining similarity between WBGs are the Counterfactual Test and the Counterfactual Conclusion, see paragraphs below. The Causal Sufficiency Test is not part of the causal reasoning to determine similarities in WBGs.

The Counterfactual Test and Necessary Causal Factors (NCF)

The Counterfactual Test[54] is one of the core principles of WBA.

To determine if two factors are causally related, one being the cause of the other, the counterfactual test asks the following question: If factor A had not been, could factor B have happened?[25] If the answer is No, B could not have happened if A had not happened, then A must be necessary for B to happen.

If the Counterfactual Test has determined that A is necessary for B to happen, we term it that A is a necessary causal factor (NCF) for B.

[3]which is also true for AcciMaps

In the real world A and B did happen. The counterfactual test is a what-if question which assumes that B did not happen contrary to fact, hence the name.

The question assumes that all other circumstances in which A and B take place remain the same, as far as possible. So the question, as stated above should be added by the phrase "all else remaining equal".

This is based on the nearest possible world semantics [39]. The nearest possible worlds semantics assumes the existence of other worlds, some like ours, some not. If we have the ability to compare other possible worlds with our own and order them according to similarity then we can find the nearest possible world W in which B did not happen. If A did not happen in W then B must be necessary for A to happen. This does not mean that there is no other possible world in which A could happen without B, but the further away such a world is from ours the less explanatory power the counterfactual test has. At some distant world from ours things may be completely different.

In a Why-Because Graph the counterfactual or NCF relation is denoted by the directed edges in the graph. The pointy end going into the effect and the stub end coming out of the cause.

Causal Conclusion

WBGs usually consist of more than two factors. Two factors may be causally related without being NCFs of each other. Effects can be causes themselves and so causal paths form in a Why-Because Graph.

Assume 4 factors as pictured in 7.1. Let *ncf (A, B)* be the counterfactual relation between A and B, where A is the necessary causal factor (NCF) and B is the effect. The diagram 7.1 shows

- *ncf(Factor1,Factor2)*,
- *ncf(Factor2,Factor3)* and
- *ncf(Factor3,Factor4)*

What does this say about the causality relation between $Factor1$ and $Factor4$?

- If $Factor1$ had not happened, then $Factor2$ could not have happened.
- If $Factor2$ had not happened, then $Factor3$ could not have happened.
- If $Factor3$ had not happened, then $Factor4$ could not have happened.

Can we follow $ncf(Factor1, Factor4)$? No, we cannot, because the counterfactual test for $Factor1$ and $Factor4$ demands, that all else should remain equal. If $Factor3$ is necessary and sufficient to cause $Factor4$ the presence or absence of $Factor1$ does not matter. Each counterfactual test is conducted in a different nearest possible world. $Factor1$ could be a necessary causal factor for $Factor4$, but we cannot simply draw conclusions from the three worlds and assume that the conclusions hold in a fourth world.

We can draw the conclusion that $Factor1$ and $Factor4$ are causally related, but the relation identified by the Counterfactual Test is not transitive. But nevertheless there is a chain of factors and NCF-relations. $Factor1$ may not be a NCF for $Factor4$, but it is a *Cause*: $cause(Factor1, Factor4)$.

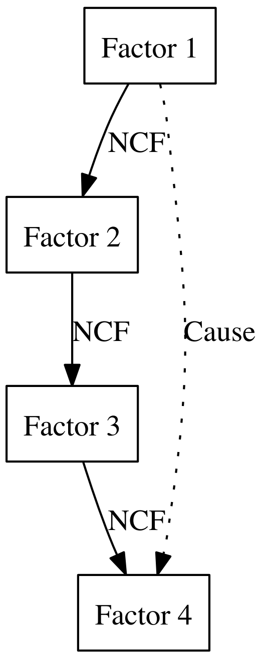


Figure 9.1: Necessary Causal Factor (NCF) chain illustrating the causal relationship between Factors 1 and 4

In 9.1 there is only one causal relation (dotted line) depicted, but of course there are many more, namely between all factors which are on the same causal path: $cause(Factor1, Factor2), cause(Factor1, Factor3), cause(Factor1, Factor4), cause(Fa$
$cause(Factor3, Factor4)$.

Causal Sufficiency and Causal Completeness

There are two other criteria used in WBA. The Counterfactual Test establishes necessary causal relations between cause and effect. The Causal Sufficiency test asks another question. If C1, C2 and C3 are all causes and NCFs for effect E, then the question is if C1, C2 and C3 are sufficient to cause E. If E must happen if C1, C2 and C3 happen, then C1, C2 and C3 are causally sufficient for E.

Both tests together form the Causal Completeness test. A WBG is correct and relatively complete if all its factors pass the Causal Completeness test. For the automatic common cause analysis the Causal Sufficiency Test is only of relevance insofar as I will assume that all WBGs processed are correct according to the Causal Completeness Test.

9.1.6 Comparing NCF Relations

Note: equality versus identity In the following paragraph notions of equality will be introducted. I interested in the comparison of accident data and therefore I need to compare elements of one accident with elements of another. Two elements are identical when they are one and the same (instance) and belong to the same accident. Two elements are equal if they share all relevant properties, but are not identical and do not belong to the same accident. The properties of elements which will matter will be discussed where the different equality measures are introduced.

Equality of NCF Relations Lets assume that there already is an equality measure defined for factors. Assume that factors are tuples of numbers, which are equal if the difference between their respective first number is 0 and which are identical if the difference between both numbers respectively is 0. The first number depicts the factor content, the latter one the accident the factor is part of.

- Factors C_1 and E_1 are in WBG_1,
- factors C_2 and E_2 are in WBG_2,
- factors C_1 and C_2 are equal $eq_1 := equalFactors(C_1, C_2)$,
- factors E_1 and E_2 are equal $eq_2 := equalFactors(E_1, E_2)$,
- C_1 is an NCF for E_1 and $ncf_1 := ncf(C_1, E_1)$,
- C_2 is an NCF for E_2 $ncf_2 := ncf(C_2, E_2)$
- then ncf_1 and ncf_2 are equal $equalNCF(ncf_1, ncf_2)$.

Similar for identity.
With this I can compare if two NCF relations occur in different accidents.

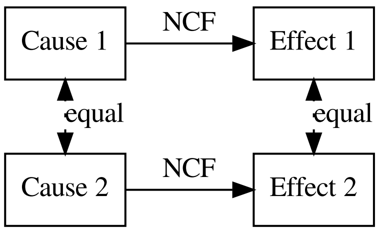

$$\begin{tikzcd}
\text{Cause 1} \arrow[r, "\text{NCF}"] \arrow[d, leftrightarrow, "\text{equal}"] & \text{Effect 1} \arrow[d, leftrightarrow, "\text{equal}"] \\
\text{Cause 2} \arrow[r, "\text{NCF}"] & \text{Effect 2}
\end{tikzcd}$$

Figure 9.2: Necessary Causal Factor (NCF) and equality relations

9.1.7 Equality of the Cause-Relation

Building on the NCF relation is the Cause-Relation9.1.5.

For finding common causes in different WBGs the Cause-Relation relations of all factors involved have to be determined. In WBGs the Cause-Relations are not explicitly included, but can be inferred from the NCF relations.

The reason that the Cause-Relation is needed is, that different levels of detail may express the same situation with different amounts of factors.

If for for WBG_1 holds:

- Factors C_1 and E_1, $F_{1,0}, F_{1,1}, ..., F_{1,n}$ are in WBG_1,
- $ncf(C_1, F_{1,0})$, $ncf(F_{1,0}, F_{1,1})$, ..., $ncf(F_{1,n-1}, F_{1,n})$, $ncf(F_{1,n}, E_1)$,
- then $cause(C_1, E_1)$

and if for WBG_2 holds:

- Factors C_2 and E_2, $F_{2,0}, F_{2,1}, ..., F_{2,n}$ are in WBG_2,
- $ncf(C_2, F_{2,0})$, $ncf(F_{2,0}, F_{2,1})$, ..., $ncf(F_{2,n-1}, F_{2,n})$, $ncf(F_{2,n}, E_1)$,
- then $cause(C_2, E_2)$

Then:

- If $equalFactors(C_1, C_2)$,
- $equalFactors(E_1, E_2)$,
- $cause_1 = cause(C_1, E_1)$ and
- $cause_2 = casue(C_2, E_2)$

- then $cause_1$ and $cause_2$ are equal $causeEqual(cc_1, cc_2)$.

If all equal Cause-Relations in two WBGs have been found, then the Causal Commonality Graph (CCG) containing only mutually equal factors as nodes and their associated Cause-Relations as edges can be drawn. The CCG has very different properties of a WBG.

- The commonatlities found in two WBGs need not be such that the CCG is connected.
- The CCG no longer carries information about NCF relations.
- It is not causally sufficient (9.1.5).

9.1.8 Causal Commonalities Graph

If two WBGs are completely equal, all factors are equal and all NCFs equal, then their CCG has the same number of factors as each WBG. But it has more edges, because there are substantially more Cause-Relations then there are NCF relations. If one WBG is a subgraph of the other WBG their CCG has a number of factors equal the number of factors of the subgraph WBG.

Cause Relation Growth The number of Coun- terfactual Conclusion Relation grows fast. Take for example

- A is an NCF for B,
- B is an NCF for C,
- C is an NCF for D and
- D is an NCF for E.

Then the following Counterfactual Conclusion relations result from this:

- A causes B, C, D and E.
- B causes C, D and E.
- C causes D and E.
- D is an NCF of E, and thereby causes E.

There are 4 NCF relations and 10 Counterfactual Conclusion relations. A linear causal chain of n NCF relations has n(n + 1)/2 counterfactual conclusion relations. Because we can infer all counterfactual conclusion relations from all non-redundant paths it is sufficient to only explicitly depict all non-redundant counterfactual relations. For one, this eliminates the need for a lot of edges to be drawn in graph visualizations and it will also serve as normalizing for edge-count measures used in clustering.

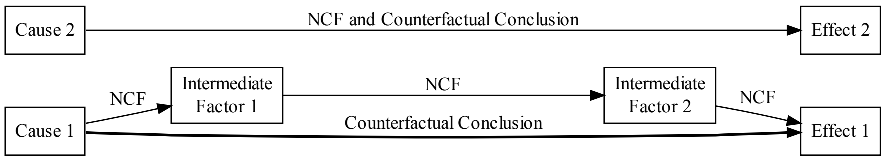


Figure 9.3: Two chains of NCFs of different levels of detail, illustrating the Counterfactual Conclusion Relations compared with NCF relation.

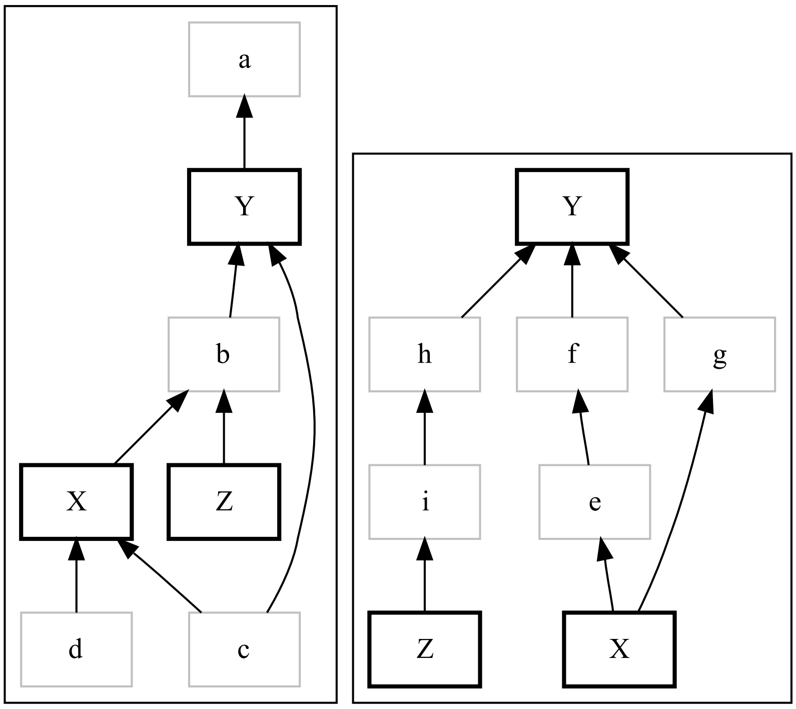


Figure 9.4: Two WBGs. The factors which are equal are denoted by capital letters and with bold factor shapes.

Cause-Relation Redundancy Elimination Take the above example with 5 factors and 4 NCF relations. From the fact that A is a cause for B, which is a cause for C we can follow that A is a cause for C. In a WBG there would be a path from A to B and a path from B to C so there is a path from A to C. If there is a path from A to C then A must be a cause for C, according to the counterfactual relation.

This entails that we can safely eliminate all but one path from A to C as long as we do not eliminate non-redundant paths of other factors.

For each factor F_0 in a CCG if there is more than one path from F_0 to different a factor F_1, then all direct paths of a length of two factors from F_0 to F_1 can be eliminated if there is at least one path from F_0 to F_1 that is longer than two factors.

This will reduce all redundant paths. The reason is, that all paths of length 2 only contain the starting and ending factors. The Cause-Relation is transitive. That means that the Cause-Relation between F_0 and F_1 can be denoted by either a path of length 2 or any longer path between the two factors. The path of length 2 can be eliminated without reducing the information content of the CCG. This automatically eliminates redundant parts of longer paths of which F_0 and F_1 are part.

The factors X,Y and Z should be equal with respect to their counterparts in the other WBG. If only the counterfactual conclusions of mutually equal factors are drawn as a graph then you can see that both WBGs share the CCG 9.5.

Normalizing Effect I want to use the Causal Commonalities Graph to cluster accident Why-Because Graphs according to common causes. There is a minimum number of causal commonality relations in a CCG when all redundant relations have been eliminated. The maximum number of relations is the number if all relations existing are explicitly depicted, as shown in 7.6. In between the number of relations can still be arbitrary without changing the meaning of the Causal Commonalities Graph. If the number of edges in a graph is arbitrary it cannot be used as a distinguishing feature useable for clustering. The minimum and maximum number graphs on the other hand have been normalized, so that graphs of the same meaning would have the same edge count. For obvious reasons I will use the graphs with their minimum edge count, that is with all edges in redundant paths between two factors removed.

A side effect of this is that the amount of data needed to store a Causal Commonalities Graph will be greatly reduced.

9.1.9 Definition of the Causal Commonalities Subgraph

Normally a subgraph of a graph is a subset of the nodes and subset of the edges restricted to the nodes. In WBG terms a WBG subgraph is subset of factors of the original WBG, and a subset of the NCF relations restricted, to the factors. Because the counterfactual conclusion relation extends further then normal edges or NCF relations do, we need a special definition of subgraph for the Causal Commonalities Graph.

A subgraph of a Causal Commonalities Graph is a subset of the factors of the original graph and a subset of all possible counterfactual conclusion relations of the original Causal Commonalities Graph restricted to the factors.

9.2 Causal Commonalities in more than two WBGs

The CCG is a product of two WBG, but the goal is to find causal commonalities in more than one WBG. CCGs can be made for each pair of WBGs. The question remains how to find common causes in of many WBGs when all we have are pairwise comparisons.

A collection of n WBGs results in $n(n-1)/2$ CCGs, one for each pair of WBGs.

The CCGs can be compared if they share common subgraphs 9.1.8.

Empty Subgraph In the case where two CCGs do not share a subgraph, the corresponding WBGs do not have any common causes.

Completely Equal CCGs If two CCGs are completely equal then all corresponding WBGs share the whole CCG as common causes. To have two completely equal CCGs means that there are three or four WBGs, depending on whether the two CCGs share one WBG, which all have common causes.

Shared Subgraphs If two CCGs share a subgraph of the two CCGs this is no different as if the subgraph would be a CCG on its own. The above for identical CCGs applies for the subgraph.

Non-Connected Causal Commonalities Graphs There is one problem with CCGs that is not important for the comparison of two WBGs, but becomes a major problem when comparing many graphs. The way that CCGs are constructed from WBGs does not ensure that the CCGs are connected. Non-connected subgraphs in both WBGs may result in a non-connected CCG. If two non-connected CCGs share a subgraph it may well be the case, that the subgraph is a number of nodes which are not causally related. To overcome this the non-connected subgraphs in CCGs should be made CCGs on their own.

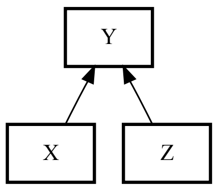


Figure 9.5: The Counterfactual Conclusion Graph CCG resulting from 9.4

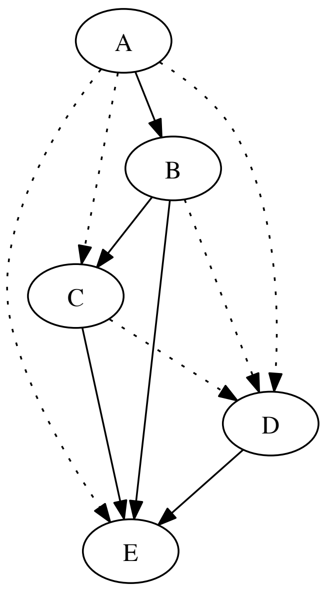


Figure 9.6: The corresponding Causal Commonalities Graph from 9.1.8 on p. 92. The graph is cluttered with redundant edges denoting causal commonalities relations, which could otherwise be inferred. The redundant edges are drawn dotted.

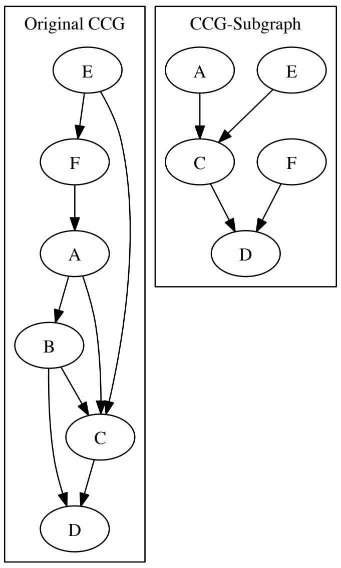


Figure 9.7: The CCG-subgraph is not a real subgraph in a graph-theoretic sense. The transitive nature of the counterfactual conclusion relation demands a different notion of subgraph for Causal Commonalities Graphs.

9.2.1 Clustering Algorithm

At the start of the algorithm we have a collection of WBGs which we want to examine for common causes. The aim to have a database of common causes which can be used to determine the most prominent common causes found among the WBGs. Note that at the moment the exact meaning of "most prominent" remains unspecified at the moment.

Algorithm Overview

Step 1: CCGs For each WBG pair in the collection of WBGs a CCG is constructed. The resulting CCGs are stored in a CCG database, which also lists all WBGs which are associated with the CCGs. After this step each CCG database entry holds one CCG and two WBGs.

Step 2: Non-Connected CCGs Each CCG is checked for connectedness. All entries of non-connected CCGs are eliminated from the CCG database. The connected real subgraphs (not in the sense of 9.1.8 of the non-connected CCGs are put into the CCG database as CCGs in their own respect, each one of which will be associated with the WBGs of the original CCG database entry. A WBG can only occur once in a CCG database entry, if a WBG is to be added to an entry where it is already contained, the add-procedure will be ignored. Each change which has been done in Step 2 must be logged for later use (see Step 4).

Step 3: CCG Subgraph Check All CCGs in the CCG database are compared pairwise.

- If two CCGs do not share a common subgraph nothing happens.
- If two CCGs are completely equal, then their two CCG database entries are merged. One CCG entry remains, while the other is eliminated from the CCG database. All WBGs of the eliminated CCG database entry which are not already part of the remaining CCG database entry are added to the remaining CCG database entry.
- If two CCGs share a subgraph then a new CCG database entry is added. The subgraph will be the CCG of the new CCG database entry and the WBGs, each only once, of the original CCG database entries will be put into the entry.

All changes in Step 3 must be logged for later user (see Step 4).

Step 4: Reiterate Steps 3 and 4 Steps 2 and 3 are repeated until there are no further changes to the CCG database. This ensures that all the new CCGs which have been generated in Step 3 are connected. Each new CCG that will be introduced by Step 2 must be processed by Step 3, because it may still change the database and affect clustering. Steps 2 and 3 must be slightly altered during the reiteration. Every time a need for a change of the CCG database is detected the log must be checked. If the proposed change occurred in the past then it must not be perfumed. Otherwise Steps 3 and 4 may run indefinitely. A CCG entry may be added by Step 2 which is then eliminated in Step 3, leaving intact the sources of the original adding in Step 2.

The final CCG Database When Steps 1-4 have run the CCG database contains a list of CCGs with all the WBGs which share the CCG. WBGs may appear in many CCG database entries, but within one entry a WBG can only occur once. The larger the CCG, in terms of factors, the bigger the commonalities between all the WBGs in the respective CCG database entry. The more WBGs the more prominent the common causes are among the WBG collection.

Change of the CCG Database over time The CCG database is a snapshot at a specific point in time. It can be used to point to common causes which can then be addressed appropriately. Taking into consideration the change of the CCG over time, important results for accident meta-analysis can derived:

- Newly emerging common cause patterns can be found early. New hazards or threats can be addressed as soon as they emerge as a trend, which may not be so prominent as to be observable by human analysts.
- Fast growing common cause patterns can be identified, compared to other trends and allow intelligent intervention.
- Stagnation of formerly growing patterns can be identified which is a good proxy for the effectiveness of interventions.

Practical Considerations

With the CCG database there is, essentially, a two-dimensional value to measure the "prominence" of common cause patterns. First, the size of the CCG in number of factors and secondly the number of WBGs which contain the behaviour described in the respective CCG. To use a CCG database to base real life decisions on, the application domain may impose restrictions on its use.

Cut-Off Not all ranges of the CCG size or the number of WBGs are important. Depending on the relative sizes of CCG or the relative number of WBGs for each CCG a cut-off may be appropriate. For example CCG database entries with CCGs of size two or three may be deemed unimportant. So may all CCG database entries with no more than two or three WBGs associated with them.

Weighing Function Depending on application domain and the set goal either the size of the CCG or the number of WBGs per CCG database entry may be more important. Additionally the introduction of weights for WBGs or CCGs of a specific type (for example high-damage WBGs) may be desirable.

Ignoring Specific Entries Essentially this would be the same as assigning as specific type of WBG or CCG a zero weight. Cases which are known to be unimportant should be filtered out.

Chapter 10

First Accident, continued

Getting back to the example from 5.3 this chapter will demonstrate how to denote the factors in terms of the attribute-value lists as described in 11.1.3 (p. 107). All factors should be expressed in terms of an ACTION, an ACTOR and a list of OBJECTS.

Recall that we already identified the actors of the Friendly Fire Accident:

- *Soldiers* will denote the U.S. Special Forces soldiers as a group.
- *Air Controller* will denote the soldier, member of the Soldiers, who was responsible for calculating targeting coordinates and calling in air strikes.
- *F18* will denote the F/A-18, the aircraft that conducted the first airstrike on the Taliban Position.
- *B52* will denote the B-52 bomber aircraft, that conducted the second airstrike, which hit the Soldier's Position.
- *Taliban Position* will denote the physical location of the Taliban which the Soldiers intended to strike.
- *Soldier's Position* will denote the physical location of the Soldiers.
- *Taliban Coordinates* will denote the GPS coordinates that correspond to the Taliban Position.
- *Soldier's Coordinates* will denote the GPS coordinates that correspond to the Soldier's Position.
- *Displayed Coordinates* will denote the GPS coordinates that appear on the display of the plugger.
- *JDAM* will denote a GPS guided bomb.

Going through the factors of the Why-Because Graph (see 59) we can identify the ACTORs, ACTIONs and OBJECTS for most of them quite easily. For example the factor "Accident: B52 drops JDAM on Soldiers Position"

- has "B52" as its ACTOR,
- the ACTION is "to drop" and
- the OBJECTs are "JDAM" and "Soldiers Position".

Given factors that are formulated as atomic factors (see 105) and stated in active voice the assignment is fairly easy. Subjects will become ACTORs. the predicate will become the ACTION and all indirect an direct objects will become OBJECTs. Adjectives, pronouns and other words will simply be omitted.

Instead of the attribute ACTION the attribute OMITTED_ACTION is used in order to denote that an action that was supposed to have happened did not. For example the factor "Cause (e): Air Controller could not tell Soldier's Coordinates from Taliban Coordinates" would be expressed as

- ACTOR: Air Controller
- OMITTED_ACTION: to distinguish
- OBJECT: Soldier's Coordinates
- OBJECT: Taliban Coordinates

One problem is with factors that are not stated in active voice and/or are not atomic. An example is the factor "Accident: 3 Soldiers killed and 20 Soldiers wounded". The factor could be rephrased, for example to "Accident: JDAM kills 3 Soldiers and injures 20 Soldiers". Atomicity can be easily mended by just splitting the factor in two. The resulting two factors must both be causes and effects of all factors that the original factor was.

Factor one:

- ACTOR: JDAM
- ACTION: to kill
- OBJECT: Soldiers

Factor two:

- ACTOR: JDAM
- ACTION: to injure
- OBJECT: Soldiers

Another example that is more challenging to the active voice problem is the factor "Proximate Cause (a): B52 within strike range of Taliban Position". First, the predicate is missing in this statement entirely, it should be rephrased to "...B52 *is* within strike range...". The factor denotes a state, that is, a property of the accident and its environment that does not change during the course of the accident. There is no actor, so all named entities here should be OBJECTs and the property of the state that is expressed should be mentioned. This factor should be expressed as

- OBJECT: B52
- OBJECT: Taliban Position
- PROPERTY: within (strike) range

The resulting Why-Because Graph would look like 10.1 (p. 104). Two factors have been omitted from the rephrasing. One factor is labelled "Assumption", the other is labelled "Unknown". Both are cases in which there is no definitive knowledge as to what a cause for a factor is. For an automatic analysis these should be omitted, because these factor types are just stand ins for the "real" factors which are unfortunately completely known.

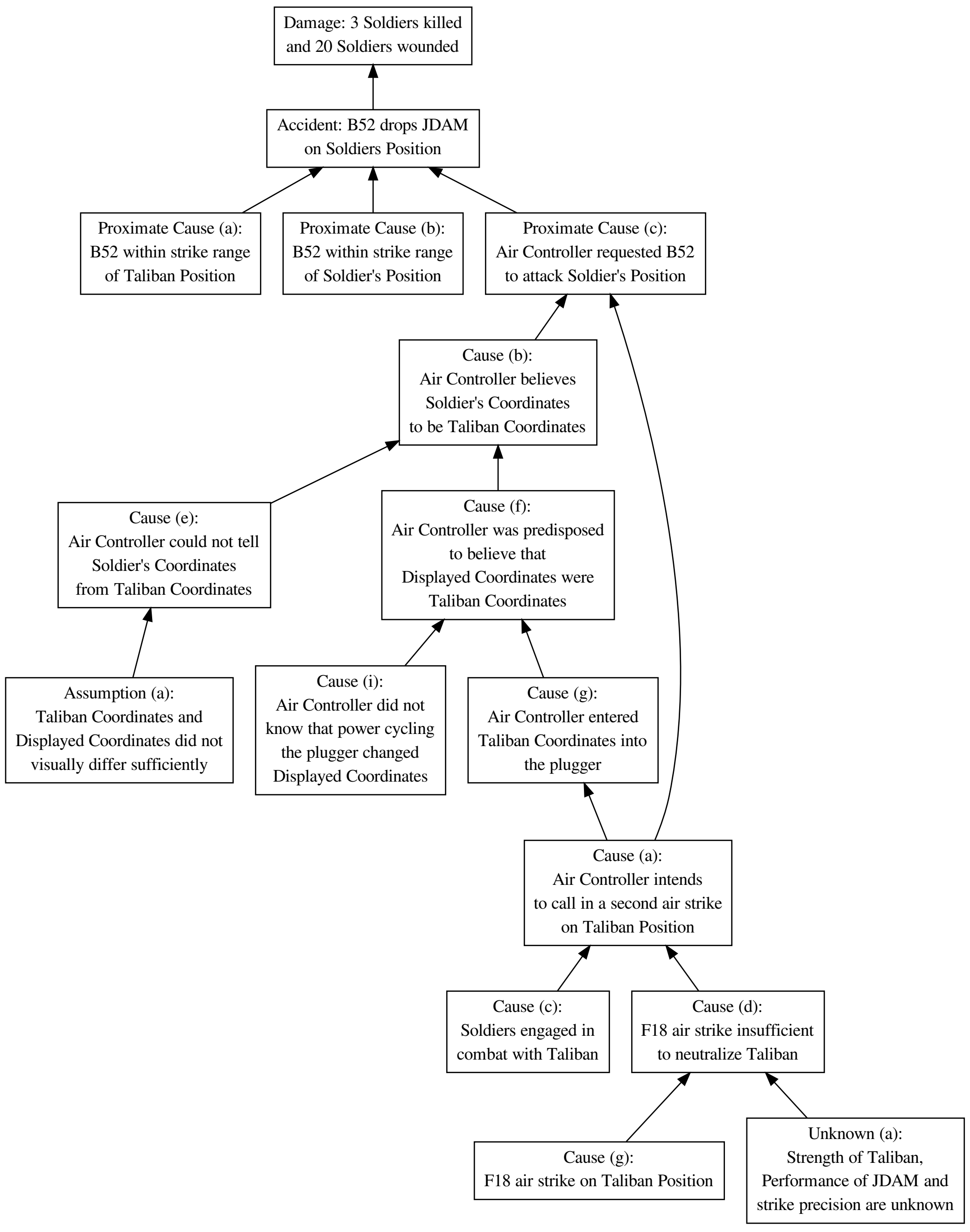

Damage: 3 Soldiers killed and 20 Soldiers wounded
Accident: B52 drops JDAM on Soldiers Position
Proximate Cause (a): B52 within strike range of Taliban Position
Proximate Cause (b): B52 within strike range of Soldier's Position
Proximate Cause (c): Air Controller requested B52 to attack Soldier's Position
Cause (b): Air Controller believes Soldier's Coordinates to be Taliban Coordinates
Cause (e): Air Controller could not tell Soldier's Coordinates from Taliban Coordinates
Cause (f): Air Controller was predisposed to believe that Displayed Coordinates were Taliban Coordinates
Assumption (a): Taliban Coordinates and Displayed Coordinates did not visually differ sufficiently
Cause (i): Air Controller did not know that power cycling the plugger changed Displayed Coordinates
Cause (g): Air Controller entered Taliban Coordinates into the plugger
Cause (a): Air Controller intends to call in a second air strike on Taliban Position
Cause (c): Soldiers engaged in combat with Taliban
Cause (d): F18 air strike insufficient to neutralize Taliban
Cause (g): F18 air strike on Taliban Position
Unknown (a): Strength of Taliban, Performance of JDAM and strike precision are unknown

Figure 10.1: The WBG with factors expressed as lists of ACTOR, ACTION, OBJECTs and PROPERTY.

Chapter 11

Attribute Value List Extractor

11.1 Equality of Factors

AcciMaps, Why-Because Analysis and most other factor-based methods use natural language to define the content of factors.

Natural language is the preferred language of humans, but for automatic processing semantics have to be expressed syntactically. Like the graph representation of accidents with WBGs, the syntax of the graphs denotes cause-effect relations, which can be processed automatically and retain their meaning of the process is meaning-preserving.

The "language" of WBGs are easily understandable for humans and, with the appropriate software to help rendering, can also be easily written. This chapter outlines how to bridge the gap between human understandable factor statements and machine processable factor statements which are meaning-preserving.

The starting point will be the familiar use of natural language by the author of a WBG. Authors can easily adhere to simple rules regarding the phrasing of texts. For example law enforcement must adhere to protocol styles, judges adhere to specific rules when writing legal opinions or judgements.

11.1.1 Atomicity of Factors

As explained in (5.3.2, 47) factor descriptions should be atomic, that is, they should not be further divisible into sensible statements.

An example: The factor "The driver changed gear and accelerated" can be divided into two statements. One "The driver changed gear." and two "The driver accelerated". Both of which cannot be further divided into meaningful statements.

Note: This is different from expressing the same situation in a different level of detail!

If the above is compared for equality with a factor "The driver changed gear and decelerated", the non-atomic statement would not be equal to any of the above, because they are stating different circumstances. But dividing the second factor into two atomic factors, then there are two equal factors "The driver changed gear".

Atomicity helps phrasing a factor as simple as possible. Complex, non-atomic, factors would contain phrases that relate the different sub-statements to one another. As we have noted (9.1.5, p. 87) relations in text cannot be easily discerned by a natural language processor. Having as few relations, between two or more things, as reasonably practical in a factor statement improves makes the task of natural language processing easier. The causal relations can be expressed as WBGs.

11.1.2 Phrasing Guideline

Factor statements resemble statements commonly found in formal protocols such as police protocols.

Use Descriptive Statements Descriptive statements should be used to retain objectivity. This is not a concern for the problem at hand. But, when teaching WBA to students the use of normative statements leads to the use of compound phrases like "wrong coordinates" or "bad data". Not using descriptive statements will reduce the number of compound phrases. Not all compound normative phrases will be eliminated, some will be replaced by compound descriptive phrases.

Use of Active Voice Use of active voice improves the predictability of phrase-function with relation to the position of a phrase (or word) in a statement.

Use of Simple Present Tense Use of only simple present tense also improves the predictability of phrase-function. Other tenses use more complex verb structures which are considerably more difficult to parse. Simple past is similarly simple, but verbs in their past form are often used as adjectives. Having simple past form verbs in different roles makes it more difficult to determine the role of word in a statement. The word function guessing heuristic takes the possibility of simple past tense into account (**??**), but the use of simple past is discouraged nevertheless.

Consistent Naming Consistent naming means, that the same thing is always denoted by the same phrase[43]. This helps in identifying fixed phrase because their recurrence in several factors is a good indication for a fixed phrase. Consistent naming is also an important factor across WBGs. In many application domains with institutionalised accident analysis there already are naming conventions in place, e.g. in aviation. If naming conventions do not exist it is well worth the effort to create one.

11.1.3 Machine Processable Representation

The proposed machine processable representation is easy to write and understand, though it may be a bit cumbersome, which is addressed later. Factors in WBGs, which adhere to the above stated rules, come in two kinds. Factors with actions like events and processes, factors describing states and factors describing omissions.

Actions Actions have one or more actors, an action and a number of objects. Objects could be further distinguished into categories, but this is not done here. The question in how far categorised objects are an improvement over non-categorised objects remains open. All events and processes can be described as attribute value lists with the attributes ACTOR, ACTION and OBJECT. There can only be one ACTION attribute per factor, there must be at least one ACTOR attribute and there can be any number of OBJECT attributes.

For example the statement ""The aircraft touched down on runway 22" would be denoted as an attribute value list:

ACTOR	aircraft
ACTION	to touch down
OBJECT	runway 22

Of more than one object is part of a statement it is simply added to the attribute value list, regardless of the kind of object. "The aircraft touched down on runway 22 with the main landing gear" would be:

ACTOR	aircraft
ACTION	to touch down
OBJECT	runway 22
OBJECT	main landing gear

This may be interpreted as "The aircraft touched down on the main landing gear with runway 22". When comparing the two sentences for equality they will appear equal according to the attribute value lists, but for most statements the attribute value list representation should be free of such collisions. If a known case does happen often, then a second action can be used to distinguish the different roles of the associated objects. In this case there would be "to touch down (1)" and "to touch down (2)". The first meaning touching down on a runway and the second meaning touching down an ones main landing gear.

Omissions Omissions describe actions, events and processes, that should have happened, but did not. Describing omissions resembles the description of actions. All omissions can be described as attribute value lists with the attributes ACTOR, OMITTED ACTION and OBJECT. Like with actions, there can be only one OMITTED ACTION attribute per factor, there must be one ACTOR attribute and there can be any number of OBJECT attributes.

ACTOR	crew
OMITTED_ACTION	to complete
OBJECT	checklist

States States describe circumstances that do not change over the course of the events described in a WBG. States describe which system or object has which property. They can be described using the attributes OBJECT and PROPERTY.

For example the statement "The runway was too short." would be denoted as an attribute value list:

OBJECT	runway
PROPERTY	short

Approximation of Natural Language Meaning The attribute value lists do contain the complete meaning of the sentences from which they are derived. But they provide a good proxy for determining if two factors describe equal circumstances or not. The concept is easy to understand, so that the author of a WBG should not have problems writing the attribute value lists for each factor once the factor statements are set. During lectures I tried the attribute value list representation on students. They were given a WBG not containing the usual natural language descriptions, but instead the attribute value lists. Understanding took considerably longer than normal and unknown words and phrases could not be guessed based on context. Otherwise the attribute value representation was understandable and the students were able to give a narrative of the accident based on the WBG shown

Determining Similarity or Equality Now that the statements are in the form of attribute value lists and the types of statements (actions, omissions and states) can be inferred from the attributes, similarity and equality can be determined algorithmically.

Two attribute value pairs are equal if the attributes are of the same kind and if the values are the same. Two factors are equal if they are of the same type and if for each attribute value pair in one factor there is exactly one equal attribute value pair in the other factor.

For any deviation from total equality a dissimilarity score can be awarded, based on rules. All factor pairs which are over a set dissimilarity threshold are not considered similar enough to qualify as equal factors for the purpose of 9.2 (p. 91). Rulesets should take into account the number of cases to consider and the number of tolerable near hits and near misses.

11.2 Automatic Guessing of Attribute Value Lists

Automatic guessing of the attribute value lists, based on the natural language factor statements, is a function intended to help the author of a WBG to prepare a WBG for use in 9 (p. 85).

11.2.1 Guessing Actors, Actions, Omitted Actions, Objects and Properties

Guessing the correct attributes is done by a multi-pass scoring algorithm. The algorithm relies on a dictionary which was derived from an English language thesaurus **??**.

Pass 1 The first pass of the algorithm determines the predominant tense the factors of a graph are stated in. This helps in determining word types in cases where there are multiple possibilities. If the predominant tense is simple present tense for example the odds that a verb encountered in simple past form is an action are lowered. Instead it becomes more likely that a simple past verb form acts as an adjective.

In a sentence like "The aircraft flies damaged" the knowledge that the predominant tense is simple present tense lets the heuristic rule in favour of "damaged" as an adjective.

Pass 2 The second pass looks for signs of non-atomicity. The algorithm works on the assumption that factor texts are atomic (11.1.1). If there are indications that a factor text is non-atomic, then the user will be reminded to check the atomicity of the factor text statement. For example, phrases like "following" or "because of" are indications that the factor text is a conjunction of more than one atomic statement. While there may be instances where non-atomic statements are necessary, all causal relations should only be expressed through the semantics of the WBG.

Pass 3 The third pass establishes which words in a factor are composite words. They are marked as such and will be treated like singular words in later passes.

Some composite words can be inferred from the thesaurus used (). Frequently co-occurring words within a WBG will be treated as composite words.

Pass 4 The fourth pass simply finds out all the possibilities of word types that a word can have. For example the word ”pilots” could mean a plural noun, the pilots of an aircraft, or a simple present tense verb, the action of someone at steering an aircraft. The words or phrases are compared with an a dictionary of word forms. Each time a match is found the word form is registered.

For example the sentence ”Pilots fly aircraft” would be classified as follows in pass 4:

pilots	plural noun	simple present verb
fly	singular noun	simple present verb
aircraft	singular noun	plural noun

Pass 5 The fifth pass establishes for each factor the likelihood it is an ACTOR, ACTION, OMITTED ACTION, OBJECT or PROPERTY. The fifth pass is the first which takes into consideration the order and context in which words appear in a factor text, e.g.:

- In an english sentence Actors come first, then Actions then Objects.
- Adjectives and articles indicate Actors and Objects, while adverbs indicate verbs.
- If a word is the only noun/verb/etc. in a factor text it must fill a specific role. The only noun must be the subject, the only verb must be the predicate.
- Statements with auxiliary verbs as only verbs indicate state descriptions.
- The phrase structure noun phrase - verb - adjective indicates state descriptions.
- An auxiliary verb followed by ”not” indicates omission phrased as omitted actions like in ”pilots do not take off”.
- ”No” before a noun indicates omissions by absence of an actor like in ”no pilot takes off”.
- Words following an article are most likely nouns.

The heuristics are helped by the explicitly set factor types each factor has.

If a word fits a heuristic it is awarded a score to indicate its probability to fit a specific role in a sentence.

Pass 6 Finally, based on the scores awarded, the attributes and values are put together. Subjects of sentences will be ACTOR values in actions and omissions and OBJECT values in state descriptions. Predicates of sentences will be ACTION value in actions and OMITTED ACTION values in omissions. Objects of sentences will be OBJECT values in actions and omissions. Adjectives of state description sentences will be PROPERTY values. The words are transformed to their base form to make them comparable with other attribute value lists form factors in other WBGs. ACTOR and OBJECT values are singular nouns, ACTION and OMITTED ACTION values are infinitive verb forms without the leading "to" and PROPERTY values are adjectives in their normal form.

Chapter 12

Why-Because Analysis Software Toolkit Design Choices

The toolkit's reference manual can be found in B (147). How WBAs can be conducted using the toolkit is explained in the Why-Because Analysis introduction in A (139).

The toolkit aids in authoring Why-Because Analyses, storing and managing Why-Because Graphs and offers automatic common cause analysis. This chapter describes the architecture, selected festures and design choices of the toolkit.

12.1 Constituent Systems

The software is written mainly using the programming language Java [13]. Java is, with some exceptions, operating system independent. The toolkit uses the dot [46] rendering engine, which is part of the Graphviz [17] project. The toolkit depends on the presence of dot, which is not written in Java, but is available for all popular operating systems. The toolkit will use dot from a locally installed Graphviz instance. The toolkit interfaces with the dot rendering engine using Grappa[42], a Java library.

The toolkit also needs an SQL database if automatic common cause analysis is required. The user can choose which SQL database to use, but has to configure the SQL interface of the toolkit, which can be done by an XML file. The toolkit uses the Java library MyBatis[53], with which the chosen SQL database has to be compatible.

12.2 Graphical User Interface

The toolkit's appearance is different from classical UI designs. The toolkits GUI is designed with the following paradigms in mind:

- Monotony
- Modelessness
- Visibility
- Conditional Availability

12.2.1 Monotony

All actions that can be done can be done in one way only. The reverse is also true. The button column on the right has the main controls to eidt WHy-Because Graphs. There are no shortcuts, context menus or drop down menus. This has been criticized by some users, who are mostly power users with a computer science background. Having a monotnous interface makes the UI simpler to use and assures, that all users use the UI in the same way. This has implications on debugging and ond supporting susers. Debugging is easier, becasue it is easier and less ambiguous to describe the actions that lead to a software fault. In support all users do actions tha same way, which reduces the gap between power users and normal users. Power useres tend to make use of all shortcuts available, but then lose their ability to support users who prefer simpler, but slower access to actions[48].

12.2.2 Modelessness

Buttons obviously are not entirely modeless. They are enables if the toolkit is in a state that allows the action a button offers and they are disabled if the state does not allow the button's action. But all actions offered by the buttons always work in the same way. There is no button which changes its behaviour in dependence of the state of the toolkit.

Modelessness ensures that a selected action will conform to the expectations of the user, because it will not choose between different functions depending on the state of the software.

12.2.3 Visibility

There are no drop-down menus and no context menus. The reason is that all actions should be visible to the user. There are exceptions, such as selection using the mouse. The software also has as few modal windows as possible. There are no actions, except for file operations and colour choosing, where modal windows are used. Java's Swing GUI framework provides ready to use colour chooser and file operations dialogues, so there was comparatively less utitily in reimplementing them.

The main advantage over conventional drop-down and context menus is that users can see at a glance all options they have. They do not need to search menus and submenus and then remember the paths to find a specific action. And users do not need to search every submenu to be sure that an action they search for is not present.

The same goes for properties of data objects. All properties can be viewed and edited using the tabbed panes in the several subviews. That way no popup wizards are needed and the user can change faster from one view into another, because everything is directly accessibele.

Having no popup windows means that there are no "are you sure" dialogues, which is compensated for by the Undo/Redo functions (see following paragraphs).

12.2.4 Conditional Availability

Buttons and widgets, with which data objects can be manipulated, are only enabled and editable if the selection state of the data objects is such that the actions executed by the buttons or widgets is sensible. Each time the selection state, or sub-selection states, change all buttons and widgets re-evaluate the preconditions neccessary for their execution. If all preconditions are met, then the button or widget becomes enabled or editable. This eliminates the need for popup windows informing the user that some condition neccessary for a requested action is invalid. A list of preconditions for all buttons and widgets can be found in the software toolkit's reference manual in B.4.3.

12.2.5 Undo and Redo

All graph authoring actions are undoable and redoable if undone. The undo history is not limited, but is reset every time the software is quit or a project is activated.

Having an Undo relieves the user of all inhibitions to just use actions for fear of destroying work unrecoverably. Editing actions with far reaching effects usually check the user's intention by modal dialogues featuring "are you sure?" questions. Havin a reliable Undo/Redo lets the user see what has actually happened and then just revert the last state (or anyone before that) if the user is not satisfied with the result.

12.2.6 Graph Rendering

The software uses the dot rendering engine to draw the Why-Because Graph. The dot rendering engine is an automaitc layout engine, which takes a specification of nodes and which noded to connect with edges and then computes a graph with a minimum of edge crossings. [21] The user does not have to arrange the nodes and edges, which is very tedious work if a graph gets larger than approximately 20 nodes. Dot itself is not an interactive tool. The dot interface library Grappa is used to read and write data to and from dot, the interactivity is implemented in the toolkit itself.

From personal conversation I know that students at the TU Braunschweig used MS Visio to draw graphs. In one special instance a student drew a graph of circa 80 nodes, which took several days just to lay out. Just layouting the same graph using the toolkit takes less than one hour.

12.2.7 User Reception

The software has been used in industrial tutorials and in the University courses. As mentioned, some power users missed shoftcuts, but otherwise all users could fully concentrate on their analyses, and not on operating the software, after they had a brief introduction which consisted of no more than 15 minutes of performing a small Why-Because Analysis. Questions on how to use the software to achieve certain goals were virtually zero.

12.3 Project and File Management

As has been described earlier CausalML [18] is used as the file format for the WBA toolkit. A WBA project is contained in a single file in the CausalML format. User specific project management data is put in the user's home directory. Neither projects nor project management data is supposed to be user serviced, but nevertheless all data will be stored in human interpretable form.

12.3.1 Project Overview

The WBA toolkit is aware of all WBA projects that have been opened in the past. The software has an integrated project browser, which will give a brief overview of all projects. Unlike other software the user does not need to do file management himself. The projects properties are displayed in the browser, so that the user is easily able to recognize projects at a glance. This eliminates the need to remember filenames and places and open the files in order to check on their content.

12.3.2 Saving Progress

The WBA toolkit will save the whole project after every single user action. The user is freed from the need of saving regularly, one of the major reasons work is lost. The user can simply exit the application and the last state will be saved to the project file.

A failure during the saving process could easily lead to significant data loss, because the Undo/Redo history will also be gone. To mitigate the risk of data loss the software keeps the last successful version and loads it in case the original file is broken.

There is still the risk that an action is performed, which alters large parts of the project. It the software crashes on saving after such an action, the action cannot be undone because the Undo/Redo history is not saved with the project. Persistent Undo/Redo is a feature that will be implemented in a future version.

12.3.3 Sharing

Because the WBA's toolkit takes complete responsibility of file management the user needs additional functionality to share projects with other users and to clean up the project list.

12.3.4 Collaboration

All projects have an owner and a lock-state. The WBA-toolkit is not a networking application, but projects that a users works on will be locked by the user to enable collaborative working on shared filesystems. Users working on projects together will see which other user currently holds a lock on a file.

12.4 Action Management

Action management executes all actions, keeps track of past actions and handles undos and redos.

All actions go through Action Management. All actions which can be accessed by user gestures, button presses or mouse clicks or drags, consist of subactions. If an action is executed it provides an ordered list of subactions to Action Management which then does the actual data manipulation. All data that is needed as context for subactions is also provided by the actions.

All subactions are also their own reverse subsection, that is their undo-subactions. If the user requests an Undo, the Action Manager retreives the last action from the action stack and executes all subaction's undo-functionality in reverse order. All subactions are designed so that this is always possible.

On a Redo the Action Manager just re-executes actions that have been undone.

12.5 Button and Widget State Management

Not only buttons generate actions, but all editable widgets do, such as text fields or checkboxes. Such widgets server a double function. One is the display of objects properties and the other is the manipulation of an objects properties. When an objects property is edited through such a widget there is no "commit" dialogue or button. The object is changed as soon as the widget looses its focus. As said earlier, unwanted changes can be undone, so there is no need for a "commit" facility.

As with buttons other widgets also are only enabled and editable if the selection state of the objects is such that the action triggered by a widget can be executed. Every time an action is executed by Action Management the software's State Management checks if the preconditions of widgets are met or not. The state of buttons and widgets is changed accordingly.

State Management is compartmentalized, so that a change in the selection state of factors will not normally trigger a reevaluation of preconditions for widgets only manipulating actors or groups. State Management also manages when to rerender the graph, which can be a time consuming process and this is only done when neccessary.

12.6 Linguistic Analysis of Factor Texts

The WBA toolkit can analyse factor texts to help in the transition between natural language and machine interpretable language. The transition is done by a heuristic which tries to determine which attribute value entries a factor should have, based on the natural language description of a factor, see 9 (p. 85). The word corpus that is used for this has been derived form the University of Princeton's WordNet 3.0 14.1.

12.7 Database

The toolkit includes an interface to an HSQLDB [22] SQL database. The interface uses the MyBatis [53] library and can be easily adapted to other database engines by configuring the SQL config file. The database schema defined by the config closely resembles the CausalML data structures. The toolkit allows easy and quick conversion between CausalML and the database. The database is not meant to be used for active projects, but to hols all finished projects which are subject to the automatic root cause cluster detection.

12.8 Root Cause Cluster Detection

The root cause cluster detection runs on the procjects stored in the database. The root cause cluster detection compares WBGs that are stored in the database pairwise and rates their similarity with one another. The more nodes and subgraphs two or more WBGs have in common the more similar they are and are assigned a similarity score. If the score is above user defined threshold it will be reported, so the user can browse the results.

To find WBGs with certain behaviour the user simply has to create a WBG that describes the behaviour and let the similarity scoring algorithms find which WBGs match the search pattern. The search pattern WBG will be treated as a Causal Commonalities Graph.

Chapter 13

Implementing the Comparison

13.1 Overview

This chapter describes the way in which the graph comparison is implemented. Intuitively one would start from the simple task of comparing nodes, get to comparing two graphs, then to comparing a number of graphs, every time building on top of the methods that have been implemented. The problem with that approach is, that it will result in a very inefficient way to solve the problem. Instead the process has to be designed with the final goal in mind from the start.

Where ever possible data structuring is designed to save as many search or compare operations as possible.

Comparing two graphs, one with x number of nodes and one with y number of nodes, results in comparing $x \times y$ comparisons. Comparing g graphs means that each graph will have to be compared to all other graphs, so there are $g - 1$ graph comparisons for each of the n graphs, resulting in $g\ times(g-1)$ graph comparisons. Let n_{avg} be the average number of nodes in a graph then the upper bound for the number of graph comparisons would is $g \times (g - 1) \times {n_{avg}}^2$.

The best way to reduce the number of comparisons is to reduce the number of graph comparisons. It would be desirable to only have to compare those graphs, that share at least two nodes each. This can be achieved if some effort is put into preselection and structuring data so that searching and comparing can be sped up using indexing[1].

[1]The indexing has not been implemented in the Why-Because Analysis Toolkit, but the indexing and table clustering features of the HSQLDB database engine have been used.

13.2 Storing Graphs, Nodes and NCF Relations

Graphs contain nodes and NCF relations between the nodes. The database table T_GRAPH is used for storing the graph handles, the table T_NODEKIND is used for storing the *node kinds* and the table T_NCF is used to store NCF relations. Two nodes are of the same *Node kinds* if they share all their properties, meaning they have the same set of attributes (ACTOR, ACTION, OMITTED_ACTION, PROPERTY and OBJECT) and the same values for all occurring attributes, where the order of OBJECT attributes does not matter.

The table T_NODEKIND is used for gathering information on the occurrence of nodes of a kind in the graphs. When a new graph is entered into the database for each node of that graph one of the two following steps happen:

1. If the table T_NODEKIND already contains a node of the same kind, then only a reference to the graph will be added to the entry of the respective node kind. An inverse reference to the T_NODE entry is stored in the respective T_GRAPH entry.
   (a) If the new graph reference is the second reference to be stored with the T_NODEKIND entry, then the inverse reference for the first graph has to be set also.
2. If the table T_NODEKIND does not contain a node of the same node kind a new entry will be created with a reference to the graph. No inverse reference will be stored in the respective T_GRAPH entry.

The idea is that two graphs need only be compared if they are co-referenced in at least two entries in the table T_NODEKIND. This property can be easily checked by counting the number of references a graph has to entries in the T_NODEKIND table.

And only those nodes which are denoted by a T_NODEKIND entry with at least two graph references have to be taken into account for comparison.

13.3 Selecting Comparison Candidates

The goal is not only to compare all graph pairs for which we know that there is a chance of having causes in common. The goal is also, for a given Causal Commonality Graph 9.1.8 (p. 92) to find all graphs in the database which contain the causal commonality graph.

The following steps will be performed on duplicates of the T_GRAPH and T_NODEKIND tables, which will be called TD_GRAPH and TD_NODEKIND. When creating the duplicate tables all graphs will be ignored that do not have at least two references to node kinds.

Going through all the entries in the TD_GRAPH table the following steps are performed:

1. All node kinds that are referenced by a graph are examined. All graph references to other graphs are written to a special table T_CANDIDATES, which will hold one entry for each graph that came up during the examination of TD_NODEKIND and a list of references to entries in TD_NODEKIND for each occurrence.
2. The nodes of the graph that have been identified to have equal nodes in the other graphs will be examined pair-wise for their Cause-Relations 5.2.2 (p. 38). The results can be
   (a) The nodes have a Cause-Relation, then the direction is noted in TD_GRAPH as an ordered couple of node kind references.
   (b) The nodes no not have a Cause-Relation, then their are eliminated from further investigation and references to their respective counterparts are deleted from T_CANDIDATES. If after this a graph ends up with an empty list of node kind references in T_CANDIDATES, then the graph will be eliminated from the table.
3. For each remaining graph in T_CANDIDATES the same Cause-Relation will be done. The result will be kept in the respective graphs TD_GRAPH table entry, to prevent double work.
   (a) Node kind references that have no Cause-Relation will be eliminated from the T_CANDIDATES table and also from the TD_GRAPH table.
   (b) Node kind reference pairs that have the wrong order of Cause-Relation will be removed from the T_CANDIDATES table, but not from the TD_GRAPH table.
   (c) Node kind reference pairs that have the right order of Cause-Relation will be noted in T_CANDIDATES.
4. If there are at least two Cause-Relations in the graph that remain after the elimination steps, it must be analysed if there are Causal Commonality Graphs bigger than only two nodes are shared by the graphs in T_CANDIDATES:
   (a) All Cause-Relations that have been found to be common to all graphs, are examined if a Cause of a Cause-Relation is equal to an Effect of a Cause-Relation. For all that are found the respective two-node Cause-Relations are put together to bigger Causal Commonality Graphs.

(b) All Causal Commonality Graphs that are bigger than just two nodes are compared, the ones in the graph under examination and the ones in the graphs referenced in T_CANDIDATES, for their maximum overlap.

(c) The results are written to the results table T_RESULTS, which holds the maximum overlap or the two-node Causal-Commonality Graphs an a list of references to the graphs which share the Causal Commonality Graph.

13.4 Example

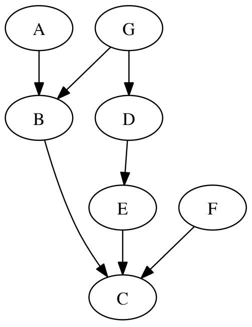


Figure 13.1: Graph 1

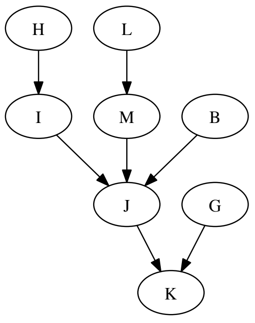


Figure 13.2: Graph 2

Node Kind	Graph ID
A	1,4
B	1,2,4
C	1,4
D	1,4
E	1,4
F	1
G	1,2,4
H	2,3,4
I	2,3,4
J	2,3,4
K	2,3
L	2,3
M	2,3,4
X	5
Y	5
Z	5

Table 13.1: T_NODEKIND example for Graphs 1 to 5

Graph ID	Node Kind
1	A, B, C, D, E, G
2	B, G, H, I, J, K, L, M
3	H, I, J, K, L, M
4	A, B, C, D, E, G, H, I, J, M
5	

Table 13.2: T_GRAPH example for Graphs 1 to 5. Graph 5 can be eliminated from further analysis.

Graph ID	Common Nodes	Cause-Relations
2	B, G	NONE
3	NONE	NONE
4	A, B, C, D, E, G	AB, CB, DB, EB, GB,CA, DA, EA, GA, DC, GC, GD, GE
5	NONE	NONE

Table 13.3: T_CANDIDATES example for Graphs 1 derived from cross referencing T_NODEKIND and T_GRAPH. Graph 4 matches the Cause-Relations AB, GB, DC, GC, GD and GE with Graph 1. This results in the Causal Commonality Graphs pictured in **??**

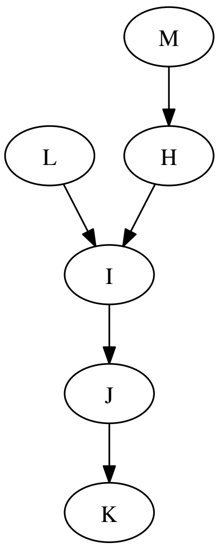


Figure 13.3: Graph 3

13.5 Tweaks

In order to filter out common nodes that are not relevant for comparison, but are for example needed for other functions, these can be marked and then be ignored. A good example for this would be if there was a node labelled ACCIDENT in some way. All graphs are analyses of accidents, but the conceptual node denoting the accident is not relevant for common *cause* analysis.

Other often occurring nodes may be marked to be ignored or filters may be installed to weed out too common nodes. In these cases some common cause analysis results may be lost.

All comparison intermediate results can be kept, given enough room for data storage, to automatically update a completely analysed set of data in case where one graph is added to the database.

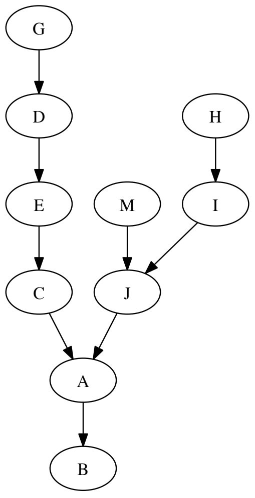


Figure 13.4: Graph 4

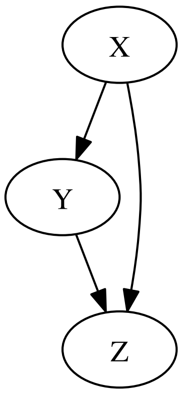


Figure 13.5: Graph 5

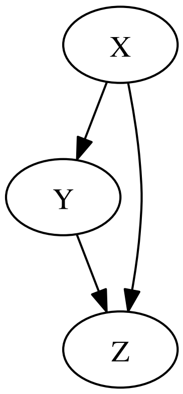


Figure 13.6: Two resulting Causal Commonality Graphs from Graph 1 and Graph 4

Chapter 14

Notes on the Software Implementation

14.1 Using the WordNet 3.0 Thesaurus

The basis for the dictionary used in 11.2 was the University of Princeton WordNet 3.0 Thesaurus see [13] and [39].

The WordNet dictionaries have been processed to be more suitable for use in the software. WordNet's verb, noun and adjective dictionaries for most of the basis for the word corpus used by the software. WordNet's index files are converted into the csv files, which can be found in the 'dict' directory of the software.

The rest of the files for pronouns, prepositions, articles and conjunctions have been created by hand, because they were not part of the thesaurus in a suitable form.

Additionally there is the option to add user-defined nouns, verbs and adjectives.

14.1.1 Word Corpus Data Flow

The index files have been filtered using shell scripting to filter out all unique words from an index file. The index files contain many cross-referencing data, which is not relevant for use in the software. The software read the processed index files and built the final files. This was implemented in Java and is part of the source code distribution of the software, but the functionality is not accessible from within the software.

The functions used to build the final word corpus can be found in the Java class com.causalis.textutils.MkWordList. The MkWordList class reads the base forms and generates a large number of inflections for the base forms. Verbs for example will be in simple present, simple past, past, continuos and infinitive form in the database. The verb 'to write' can be found as write, wrote, written, writing and (again) write. Depending on the knowledge about irregular forms, inflections of the base forms are generated either procedurally according to rules of english grammar, or by looking up the irregular forms. The lookup of irregular forms is done using the thesaurus, which lists all irregular forms. The absence of irregular forms in the thesaurus is taken as an indication that a word is formed regularly. Creating the inflections of regular words is done by the software.

The resulting files are all located in the software distributions 'dict' directory. The generated files are

- nountable.csv
- verbtable.csv and
- adjectivetable.csv.

Other files have been created by hand:

- conjunctions.lst,
- prepositions.lst and
- pronountable.csv.

If the files exist the software will also read user-defined verb, noun and adjective files:

- verbtable.userdef.csv,
- nountable.userdef.csv and
- adjectivetable.userdef.csv

At system start the word corpus files are read and an in-memory database is built by the com.causalis.textutils.WordCorpus class.

File Formats The csv files contain table data as comma separated values. The lst files are simple lists, where each line has only one entry. All user defined corpus files must be in the same format as the normal word corpus files.

All formats share the convention that all entries are done using lowercase letters.

verbtable.csv Format The verbtable.csv file has 5 entries per line. Each line holds different inflections for one verb, the order of appearance in a line determines the flection type.

1. infinitive, without 'to'
2. simple present tense
3. simple past tense
4. past tense
5. continuous tense

Composite verbs are denoted with an underscore '_'. For example 'to wish well' is denoted as

```
wish_well,wishes_well,wished_well,wished_well,wishing_well
```

nountable.csv Format The nountable.csv file has 2 entries per line. Each line holds the singular and plural inflections of a noun, the singular form is the first one, the pluralform the second. Composite nouns are denoted with an underscore '_'. For example 'landing gear' is denoted as

```
landing_gear,landing_gears
```

adjectivetable.csv Format The adjective.csv file has 3 entries per line. Each line holds different inflections for one adjective, the order of appearance in a line determined the flection type.

1. adjective
2. comparative
3. superlative

pronouns.lst, conjunctions.lst and prepositions.lst Format The conjunctions.lst and prepositions.lst files contains a list of pronouns, conjunctions and prepositions respectively. There is only one entry per line.

Looking up Words with the WordCorpus

The in-memory database reads each entry from the various sources and stores the associated form-type with the entry. For example the verbtable.csv entry for 'to pilot' is

```
pilot,pilots,piloted,piloted,piloting
```

WordCorpus would store the following in the in-memory database:

- pilot is an infinitive form
- pilots is a simple present form
- piloted is a simple past form
- piloted is (also) a past form
- piloting is a continuous form

From the nountable.csv entry for 'pilot'

```
pilot,pilots
```

WordCorpus would store in the in-memory database:

- pilot is (also) a singular noun
- pilots is (also) a plural noun

If WordCorpus is queried for the word 'pilot' it would deliver as a result that the word can be

- an infinitive form verb and
- a singular noun.

This information is later used to determine the function of a word in a factor text.

14.2 Testing the Attribute Value List Generation

To test the attribute value list generator Christoph Goeker and Tim Schuermann provided me with english language graphs of theirs. The graphs were build before they could have known of the attribute value list generator, which also means that the phasing does not match the phrasing of factors as proposed in 11.2 (p. 109).

The detection rate is mediocre with a little better than 50% of word functions correctly detected. The detection rate improved to 82% once the phrasing was brought more in line with the phrasing guideline (11.2). I did the conversion myself. I tried to be as strict with the conversion as possible, retaining as much of the original phrasing as possible. But since I have written the software and also used the WBGs provided by Chritoph Goeker and Tim Schuermann as development aids the real-world detection rate is probably lower than 80%. However, the amount of time saved by the actual performance is still worth it. Starting from scratch takes considerably longer than just going through the factors and correcting the attribute value list.

The process of guessing the attribute value list for single factors is still opaque to the user. The user is just presented with the result and has no means of looking behind the reasoning for the results. It cannot be expected from each user to try to understand the attribute value list generation, but the possibility to adapt ones phrasing of factor statements will probably improve the detection rate.

14.3 Testing the Clustering

To test the clustering (9, p. 85) data is needed on which the algorithm can be tested. Because the complete analysis of a significant amount of WBGs is out of the scope of this dissertation, randomly generated WBGs have been used.

14.3.1 Random Generation of Why-Because Graphs

The random graph generation algorithm aims to generate graphs with topologies similar to those that have been made for real accidents. While there is no reason that a specific topology should not or cannot be a WBG most real-world WBGs share most of the following features:

- More than five causes for a factor is considered many.
- More than five effects for a factor is also considered many.
- WBGs tend to be deeper than wide.
- The number of edges is about a factor of 1.5 higher than the number of factors.
- There is usually one accident factor at the top, but there is also one WBG of the Lake Constance Midair Collision [19], which has two [52].

It is assumed that the way that systems are created is responsible for the fairly regular shapes the WBGs take. If a system is assumed to be inherently safe, with a low probability of failure, then there is not much need to design additional safety measures into the system. If however the inherent safety proves insufficient, then there is at least one cause for the failure. If a system is not assumed to be safe enough inherently, then additional safety measures are taken. Either the system becomes sufficiently safe after a number of safety measures have been introduced into its design or the system design becomes too expensive and is given up. For each additional safety measure that fails in the case of an accident there is one more cause for a given failure. But the number of safety measures is limited, mostly due to cost, so there is an upper bound on the number of causes for a failure.

Random Generation Parameters The random graph generator first creates a number of factors. For the purpose of the clustering algorithm test instead of factor statements only attribute value lists are generated. For both the attributes as for the values numbers are used. Attributes are countable, so representing them with numbers accurately captures their function within the clustering algorithm. The values of attributes are also countable, so they are also represented as numbers for the same reason.

The ranges of the current implementation of the random graph generator for randomly generated factors and edges are:

- The number of entries per factor ranges from 3 to 7.
- The number of different attributes is 10 and the number of different values is 1000.
- The number of generated factors per WBG ranges from 20 to 90.
- The number of generated edges is not fixed. It starts at the number of edges necessary to construct a spanning tree over all factors, then between 25% and 60% the number of nodes additional edges are inserted between random factors, with their direction adjusted such that no circles are created.

14.3.2 Using the Random WBGs

The random generated WBGs have been used to test the software implementation of the clustering algorithm. Sufficient numbers are needed to make sure that the predicted result, namely the identification of common cause clusters, can be achieved. The current implementation can be found in the software under the "Cluster Detection" tab. At the moment there is only a table view. The resulting dataset is too complex to be easily browsable in a simple table, but the visualisation of the dataset is not trivial and not yet part of the software. I hope to include it in a future version, so that the results are easily accessible and understandable enough that they can in principle be acted upon.

Chapter 15

Conclusion

15.1 Causal Commonality Graphs

Emerging Accident Classes With the methods described in 8 and 7 the most prominent common causes in Why-Because Graphs can be found. New Causal Commonality Graphs, which emerge after the addition of Why-Because Graphs to a database, indicate a new class of accidents. New accident classes in conventional accident databases would only be introduced if the common causes of the accidents belonging to that class became prominent. This depends strongly on the number of accidents occurring and on the severity and thereby memorability of the accidents. For each Causal Commonality Graph there are at least two Why-Because Graphs, which also means two accidents. Depending on the number of accidents processed this may not be much, but the alert-threshold can be set to another number.

Freely Definable Accident Classes or statistical analysis it is still helpful to have predefined classes. These can be expressed in the form of Causal Commonality Graphs. Classes can be easily introduced if there is a need for new classes and complete reclassification of a Why-Because Graph database can be done automatically.

This is only one instance in which the dynamic classification of data is a good thing.

15.2 Computer Aided Why-Because Analysis

The software helps the author of a Why-Because Analysis is different ways. The Why-Because Analysis projects can be managed, there is even simple support for collaborative work on the same project. The lay outing of the Why-Because Graph is done automatically, as is the creation of a time line representation of the data. The automatic lay out is a great time saver and one of the factors that make processing large numbers of Why-Because

Graphs feasible. Other users of Why-Because Analysis use Microsoft Visio for lay outing. For one, try to download a Why-Because Graph from the RVS homepage and stop the time it takes to draw a visually appealing graph. After that stop the time it takes do the same in the Why-Because Analysis software toolkit. If you draw graphs with programs like Microsoft Visio all your data is in the form of graphical data formats. None is in a form that allows easy processing by other software.

Saving time means more analyses in the same amount of time. Specialised data formats means easy to process by other Why-Because Analysis software. The data format CausalML is open and everybody can use it.

Having supporting software for performing Why-Because Analyses can easily make the difference between the feasibility and in feasibility of more advanced forms of data analysis.

15.3 Further Improvement

Visualisations The search for common causes in accident data results in large datasets, which are not easily parseable. The results pose a challenge to effectively visualise them. In the current implementation only rudimentary visualisation has been done, mainly to monitor the behaviour of the underlying algorithms by their author. It is still an open question how to best present the results to the user.

Improving Attribute Value List Generation For one the ratio of incorrect attribute values entries in factors is still rather high. Finding a better set of scores and additional helpful rules are venues that must be explored. A specialisation of vocabulary may also be feasible in cases where the application domain is well known and parts of the dictionary can be eliminated.

Translating the attribute value list generator into other languages is especially worthwhile for languages of speakers in regions where the English language is not as common as it is here. This also also involves specifying new phrasing rules, finding or building an appropriate dictionary for the chosen language and developing a new scoring scheme.

To better achieve this goal the attribute value generation program code must be more modular or even scriptable. The current implementation has no provisions for easily changing this part of the software.

Integration with other Safety Data There are computer tools for diverse safety analyse methods. A priori methods estimate the safety of a system before it is build or operational. Fault-Tree Analysis for example. Much of a priori safety analysis is probabilistic assessment. How likely is it that a system fails, that it results in serious damage. The factors in Fault-

Trees are stated in the same way that the factors in Why-Because Graphs are. The relations between Fault-Tree Factors are also well defined. With the combination of accident data and Fault-Tree data the estimates taken in a system's Fault-Trees are relatively easy to check. They can be revised on the bases of actual safety performance data and the safety of the system can be reassessed more accurately.

Appendix A

Advanced Concepts in WBA

Some concepts are not immediately relevant for the automatic analysis of Why-Because Graphs. Nevertheless they are explained briefly in this Appendix. For a textbook introduction on Why-Because Analysis see [50].

A.1 Factor Types

Before you start on Factor Types you should have your causal analysis close to being finished. If you are learning WBA you should first focus on learning the Counterfactual Test and Causal Sufficiency Test. Get familiar with basic concepts that were presented during the GPS Friendly Fire case before moving on.

A.1.1 Which Factor Types are there?

Do you need Factor Types? Factor Types are classifications of Factors[1]. The classification presented here is not neccessarily the only one. You may come up with any other classification of it suits the goal of an analysis. The classification presented here consists of the folloging Factor Types, which will be discussed in detail:

- Event
- Process
- UnEvent
- State
- Assumption
- Contraindication
- Countermeasure

The WBG software tool supports all of the above. Each of the Factor Types corresponds to a node shape, so they are easily identifiable at a glance.

Event

An Event in WBA is a change of the state of the world. Unfortunately, taking this literally is not very helpful. For the purpose of Accidnet analysis, Events are changes in states of things smaller than the world. These things, commonly called systems, are not definable in advance. Usuall an Accident analyst will choose a system definition implicitly when analysing the systems behaviour, which is a sequence of Events.

If I go 50 km/h in my car and accelerate to 60 km/h it may be suffiently details to say that there was one Event, acceleration from 50 km/h to 60 km/h. When my car moves at 50 km/h it constantly chagnes its position. Each change in position may be an Event under the definition, but it may

[1]Remember: All Causes are Factors, but not all Factors are Causes.

not be one in the idealizes system used to describe Events that make sense from an Accident analysis point of view.

If more detail does not give us additional insight into an Accident, we should omit it.

Process

Processes are sequences of similar events. In our above example the acceleration of the car from 50 km/h to 51 km/h to ... to 60 km/h, can be described as a series of more detailed Events. Processes are used to describe the Process nature of a Factor. The above described Event may well be described as a Process. If the acceleration of the car is described as an Event or a Process depends on the relative analytical value of the two descriptions. If the Process nature should be emphasized it should be described as a Process.

In other cases the choice for classifying a Factor as a Process is simpler. If other Factors ocurr during a Process, which may still be going on after the Accident that is analysed, it is much easier to classify a Factor as a Process, than it is to describe the Process in terms of Events and correlate them with other Factors. During the meltdown of a nuclear power plant there are other Factors going on. Safety barriers fail, heat and pressure build, radiation emerges, monitoring and control devices fail. All these things may happen during a meltdown, and the meltdown may, at least in part, be a Cause for all of them.

UnEvent

An UnEvent is an Event that should have happened, but did not. A car that crosses a red light is an Event. But there is also an UnEvent in there. The rules of the road say that cars should stop before a red light. The car did not stop, but it should have been, so the UnEvent is that the car did not stop before the red light. Intuitively there is not much of a difference between saying *car crosses red light* and *car did not stop at red light.* It is clear to almost everybody that there are rules. The UnEvent explicitly states that

- something did not happen and
- that it should have happened.

When classifying a Factor as an UnEvent there must be a Factor that says *there is a good reason that something should have happened.* In most cases this *good reason* is some kind of rule. How explicit this rule is may well be a Cause.

State

States are true over the whole of an Accident. The rules mentioned above are good examples of States. The rule that says *stop before a red traffic light* holds throughout the Accident. There is no point in time where it does not hold. If you check your WBG for plausibility be reminded that an UnEvent usually has a State as one of at least two NCFs.

Another plausibility check that involves States is that no non-State Cause should only have States as Causes. If that were true, then the Cause would be true all the time and be a State itself.

A.1.2 Assumption

If there is not enough evidence to support a Cause, but it is clear from the Causal Sufficiency Test, that something is missing, the missing bit may be introduced as an Assumption. The Assumption should state what is assumed[2] Ideally, in an ongoing investigation, Assumptions may reveal loose ends and be resolved to be "real" Factors.

Countermeasures

In an analysis is finished Countermeasures can be implemented. To illustrate the effectiveness of Countermeasures they can be part of a WBG, but they are not first class citizens, as the Causes are. Including Countermeasures as Factor Types helps illustrate the way the Countermeasure would affect and prevent or mitigate an Accident.

Contraindication

Contraindications, like Countermeasures, are also second class citizens. They are not Factors or Causes, but in a sense, the opposite.

In an ongoing investigation there may be more than one hypothesis for the Causation of parts of an Accident. All findings that support one hypothesis go into the WBG and become Causes. But findings that challenge or contradict the main hypothesis should also be presented, as it is not helpful to exclude challenging Factors just because they don't fit in the Counterfactual Test. Ideally, like with Assumptions, these can be eliminated after an invesigation has been completed.

[2]It could also state that the only thing known is that there is something missing.

A.2 Continuous Functions

A.2.1 A Train Derailment

On the 15th of November 2004 the *City of Townsville*, a diesel tilt train, derailed near Berajondo in Queensland, Australia. One of the Factors was speed. The train was going at a speed of 112 km/h into a curve that was limited to 60 km/h. There are other Factors but in this section we want to examine continuous functions.

To illustrate the point we do the Counterfactual Test naively:

- Had the train not been travelling at 112 km/h, would it have derailed?

There are a number of possible answers to that question, depending on how to interpret it. Had the train been travelling at 113 km/h then it would have derailed. Travelling at 113 km/h is not-travelling at 112 km/h, which satisfies the constraints put by the Couterfactual Test. So the answer to the Counterfactual Test is Yes? This does not seem to be right and we would correctly point to the fact, that increasing speed increases the likelyhood of derailing. A rephrasing of the Conterfactual Test would be in order.

- Had the train not been travelling too fast, would it have derailed? No.

Is speed really a Cause?

We know for certain, that taking the curve at 60 km/h would have derailed the train. Would travelling at 61 km/h have derailed the train? We do not know, but intuitively we would say that this is close enough to 60 km/h and the train would have safely passed the curve. What speed is the limit for safely passing the curve? Do we need to know that if we want to determine if speed was a Cause?

A.2.2 Continous Values vs. Discrete Values

Let's take a step back and have a look at the general problem. The Counterfactual Test asks a Yes-No question. It is used to give an answer to the original question "Is it a Cause or not?", which is also a Yes-No question. But train speed is not an on-off issue. Train speed can be any number between 0 and the train's top speed. It need not even be an integer. What we need is a way to map the continuous values, such as train speed, to the Yes-No answers[3].

Intuitively we would select a point in the continuum, a speed number in the derailing train case, which separates the Yeses from the Nos. For example:

[3]Mathmaticians and Computer Scientists call this Discretization, just in case you'd like to know more about the general problem.

- If the train travels at or slower than 60 km/h then speed is not a Cause.
- If the train travels faster than 60 km/h then speed is a Cause.

60 km/h is an obvious candidate. First we know from experience that 60 km/h is safe, because other trains went through the curve at 60 km/h without incident. Second, there is a speed limit.

A.2.3 Back to Reality

The above mapping from speed to Yes-No, simplifies matters a little to much. Imagine that something brought the train to derail and it was traveling at 63 km/h. In court it is argued that speed was a Cause and so the driver is to blame.

No engineer would design the track without a safety margin. The train probably derails at higher speeds than 60 km/h, so is it right to put the blame on the driver for going 63 km/h[4].

Let's assume that the speed at which the train derails is 90 km/h. If we know that then we could change the mapping to

- If the train travels at or slower than 90 km/h then speed is not a Cause.
- If the train travels faster than 90 km/h then speed is a Cause.

This way a driver going 63 km/h would still be liable for exceeding the speed limit, but speed would not be a Cause for a derailment.

But is it a clear cut mapping with 90 km/h? The derailment speed may be dependend on a number of factors. The distribution of the train's mass affects its center of gravity. Environmental influences like gusts or precipitation affect train performance. Wear and tear do so, too.

All these factors may shift the derailment speed in one or another direction. In this example the shift may not be great, but we cannot assume that to be generally the case.

At different speeds speed has a different state as a Cause:

- **0 km/h - 60 km/h:** Speed is not a Cause.
- **60 km/h - lowest derailment speed (LDS):** LDS is the speed at which all environmental influences must work together in order to derail the train. From 60 km/h up to LDS the train would not derail and speed is not a Cause.

[4]We will get back to the issue of Cause verus Blame. They are not the same, even if my story seems to suggest that.

- **LDS - lowest certain derailment speed (LCDS):** LCDS is the lowest speed at which the train derails given no additional environmental factors. In between LDS and LCDS speed is a Cause, but it is not sufficient. Other Causes are neccessary.
- **LCDS - maximum speed (MS):** In this case there is no more need for external influence. Speed is sufficent as a Cause[5].

A.2.4 Conclusion

The train derailment is only an illustrative case. We have mapped a continuous function on three different results (see above). In principle we would have to map on four different results. We have not examined the case where a Factor is sufficient, but not neccessary. This would have complicated things a bit and there will be a section devoted to it. I do not see a situation where a continuous function would have such a result. Here we only have examined one-dimensional continuous functions, there may be cases where multi-dimensional continuous functions may have to be mapped on four results: neither sufficient nor necessary, necessary but not sufficient, sufficient but not necessary, necessary and sufficient.

[5]Remember that Causes are necessary by definition.

Appendix B

The WBA Toolkit Manual

B.1 Introduction

The Why-Because Analysis (WBA) Software Toolkit aims to help the WBA user to perform an incident analysis. The software has been developed as part of my doctoral thesis on advances in incident analysis. This manual provides help for admins and users of the software, but assumes that the user is familiar with WBA. For admins there is one chapter on platform and installation issues. For users there is a quick start guide, a more comprehensive guide to the softwaer and a reference guide to the functions available in the software.

The whole software, including this manual, is subject to the so called 3-clause BSD-style license[1].

B.1.1 License

[1]http://www.opensource.org/licenses/BSD-3-Clause

OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

B.2 Installation

B.2.1 Installation Steps in Short

1. Download and install the Java Runtime Environment version 1.5 or greater
2. Download and install the latest version of GraphViz
3. Download the WBA Toolkit Software
4. Copy the WBA Toolkit Software to a location of your choice
5. Start the WBA Toolkit Software with
    - ybt2.sh on GNU/Linux, OS X and other Unices
    - ybt2.bat on Windows

B.2.2 Prerequisites

All prerequisites are the same for the Operating Systems Windows, GNULinux and OS X. I have not tested installing the software on other BSD flavours. But it should be clear from the other OSs prerequisites what can and should be done.

Java Runtime Environment

The platform will need the Java Runtime Environment installed. The JRE version should be 1.5 or greater. If the JRE does not come packaged with the respective OS distribution it can be downloaded from Oracles website *http://java.com/en/download/index.jsp* .

GraphViz

GraphViz is needed as the graph layouting engine. If it does not come packaged with the respective OS distribution it can be downloaded from the AT&T Research's GraphViz website *http://www.graphviz.org/Download.php*.

The WBA Software Toolkit

At the time of writing the software is only available inside Bielefeld University's Technical Faculty network in *vollehresyssafeybt2.DATE* . The software does not need installation in the classical sense. It is sufficient to just copy the software someplace and directly execute it from there. There can be multiple instances in parallel installed, but they should not be running in parallel.

B.2.3 First Run

After first use

After starting the software for the first time there will be a new directory in the user's home directory. The directory will be called *YBT2Projects*. Inside the directory all files concerning the software will be stored. If the directory is removed the software will behave as if it started for the first time.

Testing the Reporters Functionality

Change into the tab labeled *Graph*. Click on the button on the left labeled *Node*Â inside the box labeled *Create...*. Then select the created node in the canvas on the center on the screed by once left-clicking it with the mouse. Node should be highlighted. Now click on the button on the left labeled *Cause* inside the *Create...* box. Now you should see two node that are connected by an arrow. If that is not the case please check your *GraphViz* installation especially the *dot*Â command.

B.3 Quick Start Guide

This quick start guide is intended to get you an overview over what the software can do. It is not a comprehensible guide that will explain everything to you quickly. Throughout the Quick Start Guide it will be assumed that the software started for the first time.

B.3.1 Starting the Software

You can start the software with the scripts provided.

- *ybt2.sh* on GNULinux, OS X and other Unices
- *ybt2.bat* on Windows

After starting up, which may take a while depending on the computing power of your machine, a full-screen application will be visible. On top you can see that you are currently viewing the leftmost tab called *Projects*. On the left side there are some button grouped according to function. For the moment the *Projects* view should not concern us.

B.3.2 The *Graph* View

Let's change the tab to the one on the immediate right of the *Projects* view, the *Graph* view. On the left there are grouped buttons, at the bottom is a set of tabs and in the middle is the graph although on first start there is not yet much of the graph to be seen.

Creating, Deleting and Changing Nodes in the Graph

- **Click on the button labeled *Node* in the box labeled *Create...***
- **You should see a diamond with the number 1 in it**

From here on we will call the diamond a *Node*. There are other kinds of *Nodes* that have different shapes. If you click the same button a second time then another *Node* will appear right beside the first one. To construct a graph we will not only need *Nodes*, but also *Edges*.

- **Click on the *Node* using the left mouse button**
- **The *Node*'s border should be highlighted in red**
- **Click on the button labeled *Cause* in the box labeled *Create...***
- **You should see another *Node* and a directed *Edge*[2] pointing from the newly created *Node* to the one already existing**

[2]Commonly known as an arrow

The selection of any number of *Nodes* in the graph and then using one of the *Create ... Cause Effect* buttons will

1. Insert a new *Node* into the graph
2. Connect the new *Node* with each of the selected *Nodes* with an *Edge*
3. The new *Edges* will always be pointing from an inserted *Cause* to a selected *Node*[3] and vice versa.

To remove *Nodes* from the graph use the left mouse button plus the SHIFT key to highlightselect a number of nodes. A click on the button labeled *ompletely* in the box labeled *Remove* deletes the selected *Nodes* as well as all belonging *Edges*.

- **Add additional *Causes* and *Effects* to the graph**
- **Select a number of *Nodes* of which each has at least one incoming or outgoing *Edge***
- **Click on the button labeled *ompletely* in the box labeled *Remove***
- **The *Nodes* have been deleted and all associated *Edges* have been completed with them**

Edges and *Undo/Redo*

There is one other way to introduce *Edges* into a graph. To connect two *Nodes* already in the graph

- **Click and hold the left mouse button on one *Node***
- **Drag the mouse to an other *Node***
- **Release the left mouse button above the other *Node***
- **A new *Edge* will have been drawn between the two *Nodes***

There is no visual feedback to indicate the beginning or the origin of the mouse draw gesture. On a side note: There are no functions implemented for the right mouse button (no context menus) or keyboard shortcuts.

Now you have seen the basic graph drawing capabilities of the WBA Software Tool. It will automatically layout the graph and you now know how to change the graph layout. Some additional notes for beginners:

- After every step the changes that have been made to the graph will be saved.

[3]Making this one an *Effect*

- After having done a few graph manipulations try the *Undo/Redo* buttons. *Undo/Redo* remembers all steps from either starting the software or activating a new project (which has not been discussed).
-

Labeling *Nodes*

The graph drawing capabilites of the WBA Software Toolkit would not be much withouth labeling *Nodes*. On the bottom of the screen there are a number of tabs. Select the tab labeled *Node Properties and Text*. When you single-select a *Node* using the left mouse button, then you can see that the input fields in the tab come to life.

- **Select one *Node***
- **Click inside the big text area labeled *Text* in the *Nodes Properties and Text* tab**
- **Type some text into the text area.**
- **Click back into the graph area**[4]
- **The graph will be redrawn and you should see the text appear inside the selected *Node***

The *Name* of a *Node* can be changed in a similar way.

Not only the *Node* label (or text) can be changed. The same goes for the shape of the *Node*. If you single-select a diamond shaped *Node* the *Node Kind* combo box should show the label *Unspecified*.

- **Single-select a *Node***
- **Change the *Node Kind* combo box to *Event***
- **The shape of the selected *Node* changes from diamond-shaped to box-shaped**

B.3.3 The Factor View

The graph view has some additional features to offfer. To use them effectively we need to have a look on the *Factor* view. The *Factor* view is located under the tab labeled *Factor*[5]. The view resembles the em Graph view in that there is a panel of buttons on the left, a set of tabs on the bottom and a main view in the middle. The main view in this case is a table. If you have

[4]The changes will be made after the text area looses the mouse focus; therefor clicking anywhere will assign the changes; ther is no "Assign" button for this

[5]I assume you already guessed that

a look at the contents of the table you may recognise the textual contents of *Nodes* that already are part of the graph.

Please note that there is a distinction between *Node* and *Factor*. *Nodes* are *Factors* that are part of the graph.

The *Factor* view lists all *Factors*, including those that are not part of the graph. A quick check whether a *Factor* is also an *Node* is provided by the *In Graph* colums of the *Factor* table.

- **Click on the button labeled *Factor* in the box labeled *Create...***
- **A new *Factor* should appear at the bottom of the table**

The *Factors* in this view can be manipulated in the same way as in the *Graph* view. The tab labeled *Factor Properties and Text* on the bottom of the screen can be used to change *Factors* has the same functions that the *Node Properties and Text* tab has in the *Graph* view.

Newly created *Factors* can be used directly in the *Graph* view.

- **Change into the *Graph* view**
- **From the bottom tabs select the one labeled *Factors not in Graph***
- **Select the *Factor* in the table at the bottom**
- **Select any number of *Nodes* in the graph**
- **Click on either of the buttons inside the *Insert Factor* box**[6]
- **Depending on the button you clicked the *Factor* has been inserted as a *Node*, a *Cause* or an *Effect*.**

More on *Undo/Redo*

Now change back into the *Factor* view. Delete a *Node* inside the *Factor* view and change back to the *Graph* view. When you hit the *Undo* button you will see that the action from the *Factor* view is undone. At the beginning it may be a bit confusing to undo actions that have occurred in a different view than the actual one, but this behaviour is much preferred over constantly changing views in order to undo a series of actions. Also note that undoing an action may not result in any visible change.

[6]use *Undo/Redo* to observe the different functions of the *Insert Factor* buttons

B.3.4 Printing

Printing can be done in the tab labeled *Report Generation.* On the left is a panel of buttons and on the right is a table. The document to be printed is represented as the table on the right. The first column, the one with the checkboxes, is used to determine which parts should be included and which not. The second column contains the names of the document parts. You can select a document part on the right and shift its position relative to other document parts. That way you can change the order of document parts in the resulting PDF file. Clicking on the *Save as* button in the *Report* box will open a dialog to choose a file for output. The software will remember the last save action and clicking on the *Save* button will export a PDF to the previously specified location. With the *External PDF* button in the box labeled *Create...* you can include PDF files in the report. A file dialog will open, asking for the location of the external PDF, and then a new document part will appear in the table on the right.

B.3.5 Last Words

This concludes the quick start guide. If you have gained some familiarity with the interface it should not be difficult to grasp the functions of the remaining parts of the software. For a full feature by feature explanation please use the reference manual. If you have questions or suggestions regarding this manual or the software please send me an email to jsandersTechFak.Uni-Bielefeld.DE.

B.4 Reference

B.4.1 General Notes

Starting the Software

Please use the scripts provided. For GNU/Linux and other Unices use the *ybt2.sh*[7] script. For Windows systems use the *ybt2.bat* script.

You may also want to invoke the software directly with Java. For details please consult the shell a/o bat file. If you have the source code than you may also use *ant*. There is a *build.xml* which will let you

- *ant dist*: will build the binary distribution in a directory called *dist*
- *ant run*: will run *ant dist* and then execute the program
- *ant exec*: will execute the program assuming the binary distribution has been built

Function Invocation

For all functions of the software there only one way to invoke it. That does not mean that there are no alternative ways in the software to get to the same end. It means that there are

- no keyboard shortcuts,
- no context menus and
- no main menus

in the software each providing its own way to invoke a specific function. For example there is a button that will create a node. If you click on the button the node creation function will be executed and a node will be created. The button is the only way the function can be invoked. Since there are no menus, context menus or shortcuts all functions are visible, meaning not hidden in submenus. The software offers different views of the project, because of the complexity of the whole, but within a view all functions are easily visible and accessable.

Selection State and Fuction Availability

Buttons, text fields, text areas and other interactive GUI components can change their state. Some times the components are active and can be used, at other times they are inactive. The components are always active if their function can be executed on the current selection. If a function works on one or more *Nodes* but only *Edges* are selected, then the component providing the function will be inactive.

[7]uses /bin/sh

Undoing

Nearly all functions can be undone and redone. Because the majority of functions can be undone this reference will only explicitly mention those functions that cannot be undone. Having undoable functions saves the user from popup dialogues that ask for explicit acknowledgement of functions that entail huge changes. On top of that there are no *Apply* buttons anywhere. If a change is made in a part of the user interface the changes are applied immediately. If they are not desired they can easily be undone. Selecting (or highlighting) an object is also an undoable and redoable function.

The undo history will go back to the time the active project has been loaded. That happens on startup, when a project will be loaded and every time the active project is changed.

Saving

After nearly every function invocation the whole project is saved. If the machine crashes the user can continue to work on a very recent version. When the software starts it will automaically load the last project worked on. Note that even the selection state of the project is saved, aiding the user is taking up hisher work where heshe left it.

The Log File

There is a log file called *ybt2.log.* It is written into the softwares main directory (which is the same where *ybt2.sh* und *ybt2.bat* reside). If you are unfamiliar with the source of the software or even unfamiliar with the workings of Java, you may not want to read the logfile. The value of the logfile for the normal user is to have a log of the inner workings of the software that can be sent to the developer (me[8]) along with an error description.

Please note that the log file may contain sensitive information. If you are unsure do not send the logfile! Just send an error description.

B.4.2 The Views

Generally all *Views* are divided into three parts. One panel on the left side with buttons, one tabbed pane on the bottom and one main component filling the rest.

The Project View

The *Project View* lets you select the project to work on. There is a table containing all projects and switching projects can be done with few mouseclicks. There are provisions to facilitate collaborative work on projects.

[8]jsanders@TechFak.Uni-Bielefeld.DE

The Graph View

The *Graph View* allows the display and manipulation of the Why-Because Graph. After each change to the WBG the display is refresehed and the *dot*[9] graph layout engine renders a new graph.

The Factor View

In the *Factor View* all *Factors* can be edited. Some of the functions in the *Factor View* have effects on the *Graph View*, but do not immediatly trigger the layout engint. Only after the *Graph View* is selected the changes to the *Graph* are implemented by the layouting engint.

The Actors View

Actors are used to formalize the partaking of an *Actor* in a *Factor*. *Actors* defined and described in the *Actors View* can be assinged to *Factors* in the *Factors View* and the *Graph View* as well. The *Factors Actors* relation will be used in the *Timeline View*.

The Timeline View

The *Timeline View* lists all *Factors* in chronological order and also describes which *Actors* did take part in *Factors*. The *Timeline View* uses the date and time information of *Factors* as well as the *Factors Actors* relationships to generate the *Timeline*.

The Groups View

Groups can be defined to further indicate special *Factors*. *Groups* can be associated with colours, so that the *Graph View* can visualize *Factor's* by colouring them.

The Reporter View

The *Reporter View* is designed to be a first reporter tool. Whenever there is an incident some one will be the first to report it. Because these reports are seldom in the form usable directly in a WBG this tool aims to help. A text, e.g. a protocol, can be factorized to make it a suitable starting point for a WBA.

[9]Part of the GraphViz project, see www.graphviz.org

The Report Generation View

This view provides PDF generation functions. There is a standard report form that the software can print into a PDF file. Other PDF files can be inserted in the document to increase report generation flexibility a little.

B.4.3 The Project View Reference

The Projects Table

The table shows which projects are known to the program. At least one project is alwayt the *Current Project*, which is indicated in the tables rightmost column. If on startup no project can be found a new project is created and made the *Current Project*. The software only knows the location of projects on the file and does not keep an own database. If a project file is not found at the expected location this is indicated in the *File Found* column. To facilitate collaborative work on a project a project will be marked as *locked* with the username of the locking user. The lock is not a hard lock, but only provides an indication that the file may be in use.

Buttons

Projects: Activate

- The button is enabled if a project is selected in the table on top-right, that is not markes as the *Current Project* in the last column.
  - a project is selected in the table on top-right, that is not markes as the *Current Project* in the last column,
  - the selected project file can be found as indicated by the *File Found* column,
  - and the selected project is not locked, as indicated by the *Locked By* column.
- If this button is clicked the selected project will be activated and become the *Current Project*.
- The new *Current Project* will be locked, the former *Current Project* will remain locked.
- The *Undo/Redo* history will be reset.

Projects: New

- This button is always enabled.
- If this button is clicked a new, blank project will be created.

- The new project will not be automatically activated.
- The new project will automatically be locked by the current user.

Projects: Lock

- The button is enabled if
  - the selected project is not the *Current Project*,
  - the selected project is not already locked by the current user
  - and the project's file location is known.
- If the button is clicked then a lock will be set.
- The lock is not a hard lock, it is intended to indicate that the project file is in use by another user to facilitate collaborative work on a project.

Projects: Unlock

- The button is enabled if
  - the selected project is not the *Current Project*,
  - the selected project is not already unlocked
  - and the project's file location is known.
- If the button is clicked then the lock, set by a different user, will be r emoved.
- The lock is not a hard lock, it is intended to indicate that the project f ile is in use by another user to facilitate collaborative work on a project.

Modify: Unregister

- The button is enabled if the selected project is not the *Current Project*.
- If the button is clicked the projects entry will disappear from the list of projects.
- The project's file on disk will not be affected by this. The file has to be removed separately.

Modify: Fork

- The button is enabled if a project is selected.
- If the button is clicked a copy of the selected project will be made.
- As with the *New* button the forked project will not be automatically activated.

External: Export Import

- The button in enabled if
  - a project is selected
  - and the project's file could be found.
- If the button is clicked
  1. The user is prompted to select a file from the file system.
  2. The file will be checked if it is a valid project file.
  3. If the file is a valid project file it will be registered in the project view.
  4. Work on the project will be done directly on the selected file.

External: Import

- The button is enabled if
  - a project is selected
  - and the project's file could be found.
- If the button is clicked
  1. The user is prompted to select a file location.
  2. The selected project will be copied to the location specified by the user.

Other Functions

Selecting a Project Selection of a project is done with the left mouse button in the projects table. Only single selection is allowed.

Project Location This label shows the file location of the *Current Project*. It cannot be edited. If the project should reside in a different location the use of *Modify: Fork* is suggested.

Author This, by default, is filled with the user name (or logname) of the current user. It is encouraged to change this to the actual authors' names.

Master Date Time The date and time in this field are used when creating new *Nodes* or *Factors*. Newly created *Nodes* or *Factors* have a datetime field on their own. The *Master Date Time* sets the default on creation.

Title The title or short description of the project.j

Description A longer description of the project. This field only accepts plain text, there are no formatting or highlighting provisions.

B.4.4 The Graph View Reference

The Graph

The graph is displayed in the top-right part of the display. The graph is re-layoutet every time a change to the layout is done.

Selection *Nodes* and *Edges* can be select with the left mouse button. There is no function on the right mouse button. Multi-selection is possible using the left mouse button and holding the CTRL key pressed during the mouse click. The same goes for deselecting an element from a selection. If the left mouse button is clicked on the backgroud all objects will be deselected. Selecting and deselecting are undoable functions.

Drawing an *Edge* A new *Edge* between two *Nodes* can be drawn with the mouse by dragging the mouse with the left mouse button from the *Cause Node* to the *Effect Node*.

Create: Node

- This button is always enabled.
- Clicking on this button will create a new *Node* in the graph.
- The new *Node* does not have any *Edges* created with it.
- After creation the new *Node* can usually be found on the bottom-right or bottom-left of an already exising graph.

Create: Effect

- This node is enabled if
  - a number of *Nodes* have been selected
  - and no *Edges* have been selected.
- Clicking on this button will create a new *Node*.
- Additionally new *Edges* will be created along with the new *Node* making the new one an *Effect* for all selected *Nodes*.

Create: Cause

- This button is enabled if
  - a number of *Nodes* has been selected
  - and no *Edges* have been selected.
- Clicking on this button will create a new *Node*.
- Additionally new *Edges* will be created along with the new *Node* making the new one a *Cause* for all selected *Nodes*.

Remove: Completely

- This button is enabled if a number of *Nodes* a/o *Edges* has been selected.
- Clicking this button will delete the selected *Nodes* and *Edges*.

Remove: From Graph

- This button is enabled if a number of *Nodes* a/o *Edges* has been selected.
- Clicking this button will
  - remove the selected *Nodes* from the graph, but will retain them as *Factors* and
  - delete all selected *Edges*.

Insert as: Node

- This button is enabled if a number of *Factors* from the *Factors Not in Graph Table* have been selected.
- Clicking on this button will insert the selected *Factors* as *Nodes* into the graph.
- The new *Nodes* do not have any *Edges* created with them.
- After insertion the new *Nodes* can usually be found on the bottom-right or bottom-left of an already exising graph.

Insert as: Effect

- This button is enabled if
  - a number of *Nodes* have been selected,
  - a number of *Factors* from the *Factors Not in Graph Table* have been selected,
  - and no *Edges* have been selected.
- Clicking on this button will insert the *Factors* as *Nodes*.
- Additionally new *Edges* will be created along with the inserted *Factors* making all *Effects* for all selected *Nodes*.

Insert as: Cause

- This button is enabled if
  - a number of *Nodes* have been selected,
  - a number of *Factors* from the *Factors Not in Graph Table* have been selected,
  - and no *Edges* have been selected.
- Clicking on this button will insert the *Factors* as *Nodes*.
- Additionally new *Edges* will be created along with the inserted *Factors* making all *Causes* for all selected *Nodes*.

Subgraph: Collapse

- The button is enabled if one *Node* is selected.
- If the button is clicked the graph below the selected *Node* will be substituted by a placeholder *Node*

Subgraph: Expand

- The button is enabled if one *Node* is selected which is an *Effect* for a subgraph *Node*.
- If the button is clicked the placeholder *Node* will be expanded into the full graph

Zoom: Zoom In (+)

- The button is always enables.
- Clicking the button will enlarge the graph displayed in *Graph View*

Zoom: Zoom Out (-)

- The button is always enables.
- Clicking the button will scale down the graph displayed in *Graph View*

Zoom: Scale to Fit

- The button is always enables.
- Clicking the button will scale the graph so that it can be seen completely in the *Graph View*.
- Note that nodes may be too small to read their containing text.

Zoom: Normal (1:1)

- The button is always enables.
- Clicking the button will display the graph at its normal scaling factor.

Factors not in Graph

- This table shows all *Factors* that are not *Nodes* in the graph.
- Each table row denotes one *Factor*
- The selection of the *Factors* shown in the table is used by the *Insert Factors* button group.
- A comprehensive view of all *Factors* and *Nodes* in the project can be found in the *Factors View*.

Node Properties and Text

Node Proerties and Text: ID

- This field is not editable.
- The ID is a unique identifier mainly used internally by the software.
- It is provided here in order to facilitate debugging in case the project file gets corrupted.

Node Properties and Text: Name

- This text field is enabled if one *Node* is selected.
- This text field contains the name of a *Node.*
- The name of a node is usually automatically set by the software on creation on the *Factor* or *Node*
- In a *Node* the name is displayed on top of the node text in brackets.

Node Properties and Text: Date/Time

- This text field is enabled if one *Node* is selected.
- The *Node's* timestamp can be manipulated with this text field.
- The text field's content is not show within the *Node.*
- Date/Time is used to create the *Timeline* in the *Timline View.*

Node Properties and Text: Node Kind

- This combo box is enabled if one *Node* is selected.
- The node kind determines the *Nodes* shape in the graph.
- At the time of writing the following *Node Kinds* exist:
  1. Unspecified - the default, denoting that no choice has been made which *Node Kind* the *Node* should be assigned.
  2. Event - denoting an event in the sense of a point in time where a system changes its state.
  3. UnEvent - denoting a point in time where a system should have changed its state but did not.
  4. State - denoting a state that is true througout the whole of the incident .

5. Process - denoting a series of sufficiently similar event so that they can be aggregated as a process.
6. Assumption - denoting a *Node* for with at best circustantial evicence exists, but is believed to be true.
7. Countermeasure - denotes a measure to counteract a specific *Node* in order to defeat the incident.
8. Contraindication - denotes the fact that there may be conflicting information about a *Node* to be true or false.

Node Properties and Text: Text

- This text area is enabled if one *Node* is selected.
- The text in this text area is the *Node's* main text which appears inside the node.

Annotations, Actors, Groups

Annotations, Actors, Groups: Annotation

- This text area is enabled if one *Node* is selected.
- The text in this text area is the *Node's* additional information and is not part of the visible text in the graph. ide the node.

Annotations, Actors, Groups: Actor Assignment

- This table is enables if one *Node* is selected.
- This table lists all *Actors* that have been defined int the *Actors View*.
- By clicking on a checkbox in an *Actor's* row an *Actor* is assigned to the selected *Node*.
- A *Node* may have any number of *Actors* associated to it.
- Changing this table has no effects on the graph.
- The Actor Assignment is used in the *Timeline View* to construct the *Timeline*.

Annotations, Actors, Groups: Group Assignment

- This table is enables if one *Node* is selected.
- By clicking on a checkbox in a *Group's* row this group is assigned to the selected *Node*.

- One *Node* can only be in one *Group*, therefor selecting a *Group* will automatically unselect a previously selected *Group*.
- The *Node* in the graph will be coloured with the associated *Group's* colour.

Edge Detail

Edge Detail: Justified

- This button is enabled if one *Edge* is selected.
- Clicking this button will mark the selected *Edge* as justified[10].
- The Edge will be highlighted blue when it is marked as justified.

Edge Detail: Not Justified

- This button is enabled if one *Edge* is selected.
- Clicking this button will mark the selected *Edge* as not justified[11].
- The Edge highlighting will be turned off and the *Edge* will appear black again.

Edge Detail: Edge Label

- This text field is anabled if one *Edge* is selected.
- Editing this text field will change the label of the selected *Edge*.
- Items will be displayed halfway between the *Edge's* tail and head.

Edge Detail: Edge Justification

- This text area is enabled if one *Edge* is selected.
- The text entered into the text area does not affect the layout of the graph.
- The text will be used in *Report Generation*.
- The text is meant to explain why two *Nodes* are causal factors.

[10]It passed the Counterfactual Test
[11]It did not pass the Counterfactual Test

B.4.5 The Factor View Reference

The *Factor View* contains an overview of all *Nodes* and *Factors* of the project. The main difference to the *Graph View* is the tabular representation of data instead of the graphical representation used in the *Graph View*.

The Factor Table

Create... Factor

- This button is always enabled.
- Clicking this button will create a new *Factor*, which will appear as the bottom most *Factor* in the table.

Remove completely

- This button is enabled if at least one *Factor* or *Node* is selected.
- Clicking this button will delete all selected *Factors* or *Nodes*.
- Deleting *Nodes* will lead to a redraw of the graph after changing into *Graph View*

Remove from Graph

- This button is enabled if at least one *Node* is selected.
- Clicking this button will remove all selectd *Nodes* from the graph and make them *Factors*.
- Removing *Nodes* from the graph will lead to a redraw of the graph after changing into the *Graph View*.

Factor Properties and Text

Factor Properties and Text: ID

- This field is not editable.
- The ID is a unique identifier mainly used internally by the software.
- It is provided here in order to facilitate debugging in case the project f ile gets corrupted.

Factor Properties and Text: Name

- This text field is enabled if one *Node* or *Factor* is selected.
- This text field contains the name of a *Node* or *Factor*.
- The name of a *Node* or *Factor* is usually automatically set by the software on creation on the *Factor* or *Node*
- In a *Node* the name is displayed on top of the node text in brackets.

Factor Properties and Text: Date/Time

- This text field is enabled if one *Node* or *Factor* is selected.
- The *Node* or *Factor's* timestamp can be manipulated with this text field.
- The text field's content is not show within the *Node*.
- Date/Time is used to create the *Timeline* in the *Timline View*.

Factor Properties and Text: Node Kind

- This combo box is enabled if one *Node* or *Factor* is selected.
- The node kind determines the *Nodes* shape in the graph.
- At the time of writing the following *Node Kinds* exist:
  1. Unspecified - the default, denoting that no choice has been made which *Node Kind* the *Node* should be assigned.
  2. Event - denoting an event in the sense of a point in time where a system ch anges its state.
  3. UnEvent - denoting a point in time where a system should have changed its s tate but did not.
  4. State - denoting a state that is true througout the whole of the incident .
  5. Process - denoting a series of sufficiently similar event so that they can be aggregated as a process.
  6. Assumption - denoting a *Node* for with at best circustantial evicence exists, but is believed to be true.
  7. Countermeasure - denotes a measure to counteract a specific *Node* in o rder to defeat the incident.
  8. Contraindication - denotes the fact that there may be conflicting informati on about a *Node* to be true or false.

Factor Properties and Text: Text

- This text area is enabled if one *Node* or *Factor* is selected.
- The text in this text area is the *Node's* main text which appears inside the node.

Annotations, Actors, Groups

Annotations, Actors, Groups: Annotation

- This text area is enabled if one *Node* or *Factor* is selected.
- The text in this text area is the *Node's* or *Factor* additional information and is not part of the visible text in the graph.

Annotations, Actors, Groups: Actor Assignment

- This table is enables if one *Node* or *Factor* is selected.
- This table lists all *Actors* that have been defined int the *Actor s View*.
- By clicking on a checkbox in an *Actor's* row an *Actor* is assigne d to the selected *Node* or *Factor*.
- A *Node* or *Factor* may have any number of *Actors* associated to it.
- The Actor Assignment is used in the *Timeline View* to construct the *Timeline*.

Annotations, Actors, Groups: Group Assignment

- This table is enables if one *Node* or *Factor* is selected.
- By clicking on a checkbox in a *Group's* row this group is assigned to the selected *Node* or *Factor*.
- One *Node* or *Factor* can only be in one *Group*, therefor selecting a *Group* will automatically unselect a previously selected *Group*.

B.4.6 Actors View Reference

In the *Actors View Actors* can be managed. *Actors* can be assigned to *Nodes* or *Factors* are are used to construct the *Timeline*.

Actor Table

Create... Actor

- This button is always enabled.
- Clicking this button will create a new *Actor*.
- Properties of a newly created *Actor*Â will be filled with values indicating that the *Actor* has not been worked on yet.

Remove Actor(s)

- This button is enabled if at least one *Actor* is selected.
- Clicking this button will delete the selected *Actors*.
- References of *Nodes* or *Factors* to a deleted *Actor* will be automatically deleted.

ID

- This field is not editable.
- The ID is a unique identifier mainly used internally by the software.
- It is provided here in order to facilitate debugging in case the project f ile gets corrupted.

Name

- This text field is enabled if one *Actor* is selected.
- This text field contains the name of a *Actor*.

Description

- This text area is enabled if one *Actor* is selected.
- The text in this text area is the *Actor's* additional information.

B.4.7 Timeline View Reference

The *Timeline* is derived from the data given in the *Actors View* and the Graph View or *Factors View*. *Nodes* and *Factors* can be associated with *Actors*. Each row in the *Timeline* represents a *Node* or *Factors*. Each column, with the excption of the first three colums, represents an *Actor*. If the checkbox in an *Actor* column is set, then the respective *Actor* is taking part in the event(s) described in the *Node* or *Factor*. The first columns are

- The *Factor* column displays the text of the *Node* or *Factor*.
- The *Date/Time* column displays the date and time at which the event(s) described happen.
- The *Duration* column displays the duration of the described event(s).

In contrast to most other tables the checkboxes in the *Timeline* table are editable. They serve the same function as the small *Actor* tables in the *Graph View* or the *Factor View*.

B.4.8 Group View Reference

In teh *Group View Groups* can be managed. *Groups* can be assigned to *Factors* and *Nodes* and affect their colouring in the graph.

Group Table

Create... Group

- This button is always enabled.
- Clicking this button will create a new *Group*.
- Properties of a newly created *Group*Â will be filled with values indicating that the *Group* has not been worked on yet[12].

Remove Group(s)

- This button is enabled if at least one *Group* is selected.
- Clicking this button will delete the selected *Groups*.
- References of *Nodes* or *Factors* to a deleted *Groups* will be automatically deleted.
- This may affect the graph and force a redraw when the *Graph View* is next displayed.

ID

- This field is not editable.
- The ID is a unique identifier mainly used internally by the software.
- It is provided here in order to facilitate debugging in case the project f ile gets corrupted.

[12]The default colour is white.

Name

- This text field is enabled if one *Group* is selected.
- This text field contains the name of a *Group*.

Colour Chooser

- This button is enabled if one *Group* is selected.
- Clicking this button will open a colour chooser dialogue from which the *Group's* colour can be selected.
- This may affect the graph and force a redraw when the *Graph View* is next displayed.

Description

- This text area is enabled if one *Group* is selected.
- The text in this text area is the *Group's* additional information.

B.4.9 Reporter View Reference

The *Reporter* is a tool to assist in the beginning on an analysis. It is an aid to transform a natural language text into chunks, each chunk being fit as a first approximation to a *Factor* text.

The Reporter Text Area

The text area is where the input to the *Reporter View* is put. To achieve the best results the text provided here should be

- in simple present tense,
- and in active voice.

Most protocols will be in this form.

Factorize

- This button is always enabled.
- Clicking the button will factorize the text provided in the text area, which will subdivide the text into *Factorized Items* using indicators from punctuation and phrasing, each *Factorized Item* should be a good approximation for an atomic[13] expression.

[13]Not further subdivisible and still making sense.

- For each *Factorized Item* a new row will be inserted into the table at the bottom of the text area.
- All *Factorized Items* in the table will be deleted on factorizing.

Delete Items

- This button is enabled if one or more *Factorized Items* in the table are selected.
- Clicking this button will delete all selected *Factorized Items* in the table.

Create Item

- This button is always enabled.
- Clicking this button will insert a new, empty *Factorized Item* into the table.

Clone Items

- This button is enabled if one or more *Factorized Items* in the table are selected.
- Clicking this button will duplicate all selected *Factorized Items* in the table.

Merge Items

- This button is enabled if two or more *Factorized Items* in the table are selected.
- Clicking this button will delete both slected *Factorized Items* and create a new one containing the text of both previously selected *Factorized Items.*
- The order in which the *Factorized Items* appear in the table determines the order in which the texts are concatenated.

Make New Analysis

- This button is always enabled.
- Clicking this buttion will delete all a projects *Nodes* and *Factors* and create new *Factors* from the *Factorized Items.*
- *Factorized Items* marked as *Damage, Incident* or *Proximate Cause* will be converted to *Nodes* and a preliminary graph will be created.

Factorized Item Table

The table contains all *Factorized Items*. The first column contains the textual descriptions, the second column the type of a *Factorized Item*. The table itself is not editable.

Factorized Item Text Area

- The text area is enabled if one *Factorized Item* is selected.
- The text in this text area is the *Factorized Item's* main text which appears in the table.

Factorized Item Combo Box

- The combo box is enabled if one *Factorized Item* is selected.
- The combo box changes the selected *Factorized Item's* type to one of the following:
  - *None*: This is the default upon creation of a *Factorized Item*.
  - *Damage*: This indicates, that the *Factorized Item* describes the damaga in a reported incident.
  - *Incident*: This indicates, that the *Factorzied Item* describes the incident event in a reported incident.
  - *Proximate Cause*: This indicates that the *Factorized Item* describes a cause leading to the incident event.
- When creating a preliminary graph from the *Reporter View* all *Proximate Causes* will be causes for all *Incidents*, which will be causes for all *Damage*.

B.4.10 Report Generator Reference

The table in the *Report Generator View* shows the *Report Generator Parts* that will be included in a PDF export. *Report Generator Parts* can be

1. A *Cover*, usually for the front page of a *Report*.
2. The *Graph* on one page, which may leave *Node* texts unreadable.[14]
3. The *List of Factors*, which contains all *Nodes* and *Factors* of the active *Project*.
4. The *Causal Justification List*, similar to the *List of Factors*.

[14] An alternative way to display the graph is in progress.

5. The *Group List*, similar to the *List of Factors*.
6. The *Actor List*, similar to the *List of Factors*.
7. The *Timeline* which is a chronological table of all *Nodes* and their respective *Actors*.
8. Additionally other PDF documents can be used as sources for a *Report*. See *Create ... External PDF*.

The order as shown in the table will be the order in which the *Report Generator Parts* will appear in the finished PDF. *Reports Generator Parts* can be excluded from the PDF export by unselecting the checkboxes in the table.

Report: Save As

- This button is always enabled.
- Clicking this button will open a file output dialogue.
- After a file location has been selected the PDF export will write the *Report* to that file.
- The file location will be stored in the *Project*.

Report: Save

- This button is always enabled.
- The PDF export will write the *Report* to a previously selected file location.
- If there is no previously selected file location the behaviour will be as in *Report Save As*.

Move Parts: Move Up

- This button is is enabled if one *Report Generation Part* is selected.
- Clicking this button will move the seleveted *Report Generation Part* one row up in the table, with repect to the other *Report Generation Parts*.
- If the selected *Report Generation Part* is already the top most, nothing is done.

Move Parts: Move Down

- This button is is enabled if one *Report Generation Part* is selected.
- Clicking this button will move the seleveted *Report Generation Part* one row down in the table, with repect to the other *Report Generation Parts*.
- If the selected *Report Generation Part* is already the bottom most, nothing is done.

Create: Cover

- This button is enabled if one *Report Generation Part* is selected.
- Clicking this button will insert a *Report Cover* below the selected *Report Generation Part*.

Create: Graph

- This button is enabled if one *Report Generation Part* is selected.
- Clicking this button will insert a *Report Graph* below the selected *Report Generation Part*.

Create: Factor List

- This button is enabled if one *Report Generation Part* is selected.
- Clicking this button will insert a *Report Factor List* below the selected *Report Generation Part*.

Create: Causal Justification List

- This button is enabled if one *Report Generation Part* is selected.
- Clicking this button will insert a *Report Causal Justification List* below the selected *Report Generation Part*.

Create: Group List

- This button is enabled if one *Report Generation Part* is selected.
- Clicking this button will insert a *Report Group List* below the selected *Report Generation Part*.

Create: Actor List

- This button is enabled if one *Report Generation Part* is selected.
- Clicking this button will insert a *Report Actor List* below the selected *Report Generation Part.*

Create: Timeline

- This button is enabled if one *Report Generation Part* is selected.
- Clicking this button will insert a *Report Timeline* below the selected *Report Generation Part.*

Create... External PDF

- This button is enabled if one *Report Generation Part* is selected.
- Clicking this button will open a file open dialogue, to select a PDF for inclusion in the *Report.*
- A new *Report Generation Part* will be inserted into the table below the selected *Report Generation Part.*

Bibliography

[1] U.S. National Aeronautics and NASA Space Administration. Aviation Safety Reporting System. http://asrs.arc.nasa.gov/, 2012.

[2] Ludwig Benner. Accident investigation: Multilinear events sequencing methods. *Journal of Safety Research*, 7(2), March 1975.

[3] Bieleschweig Workshop participants. The Bieleschweig Workshops on Systems Engineering. http://www.rvs.uni-bielefeld.de/Bieleschweig/ and https://www.tu-braunschweig.de/ifev/veranstaltungen/bieleschweig, 2002 - 2011.

[4] Bieleschweig Workshop participants. Comparison Criteria. http://www.rvs.uni-bielefeld.de/Bieleschweig/criteria/, 2003. Criteria for comparing different methods of root-cause analysis.

[5] U.S. National Transportation Safety Board. Transportation safety databases. Technical Report NTSB/SR-02/02 PB2002-917004, U.S. National Transportation Safety Board, 2002.

[6] U.S. National Transportation Safety Board. Crash during landing - executive airlines (doing business as american eagle) flight 5401, avions de transport regional 72-212, n438at, san juan, puerto rico. http://www.ntsb.gov/doclib/reports/2005/AAR0502.pdf, 2004.

[7] U.S. National Transportation Safety Board. NTSB Accident Reports. http://www.ntsb.gov/investigations/reports.html, 2012.

[8] Boeing. http://www.boeing.com/news/techissues/pdf/statsum.pdf.

[9] U.K. Air Accidents Investigation Branch. AAIB Formal Report Archive. http://www.aaib.gov.uk/sites/aaib/publications/formal_reports/formal_report_archive.cfm, 2012.

[10] Australian Transportation Safety Bureau. ATSB Research and Analysis Reports. http://www.atsb.gov.au/publications/publications-list.aspx?mode=all&publicationType=Research%20and%20Analysis%20Report, 2012.

[11] Claire Blackett (UC Dublin). Analysis of the Royal Majesty Grounding Using SOL. http://www.rvs.uni-bielefeld.de/Bieleschweig/third/Blackett-B3-2004.pdf, 2004.

[12] Centro de Investigacao e Prevencao de Acidentes Aeronautocos Commando da Aeronautica. Final report a - no 67/enipa/2009. http://www.cenipa.aer.mil.br/cenipa/paginas/relatorios/pdf/3054ing.pdf, 2009.

[13] Oracle Corporation. Java runtime environment (jre) and java development kit (jdk). http://www.java.com.

[14] French Bureau d' Enquêtes et d' Analyses pour la securite de l' aviation civile. BEA. http://www.bea.aero/index.php, 2012.

[15] Dr. Babette Fahlbruch, Dipl.-Psych. Rainer Miller. Safety through Organizational Learning (SOL) - an in depth event analysis methodology. http://www.rvs.uni-bielefeld.de/Bieleschweig/first/Fahlbruch_Miller_SOL-Handout.pdf, 2002.

[16] Dr Dmitri Zotov, MBE, PhD, MAv, FRAeS. Multi Linear Events Sequencing. http://www.asasi.org/papers/2011/MES Analysis - Dmitri Zotov.pdf, 2011.

[17] John Ellson, Emden Gansner, Yifan Hu, Arif Bilgin, and AT&T Research Dwight Perry. Graphviz - graph visualization software. http://www.graphviz.org, 2012.

[18] Luke Emmet, Mirco Hilbert, Peter B. Ladkin, Jan Paller, Jan Sanders, Jörn Stuphorn, Bernd Sieker, and Fergus Toolan. Causalml language definition xs:schema file. http://rzv113.rz.tu-bs.de/dlfo/CausalML/CausalML_1.1.0_nd.xsd, 2006.

[19] Bundesstelle für Flugunfalluntersuchung. Investigation report ax001-1-202. Technical report, 2010.

[20] Bundesstelle für Flugunfalluntersuchung. BFU. http://www.bfu-web.de/, 2012.

[21] E. R. Gansner, E. Koutsofios, S. C. North, and K.-P. Vo. A technique for drawing directed graphs. *IEEE Trans. Softw. Eng.*, 19(3):214–230, March 1993.

[22] HSQL Development Group. www://hsqldb.org.

[23] Andrew Hopkins. An accimap of the esso australia gas plant explosion. http://www.qrc.org.au/conference/_dbase_upl/03_spk003_Hopkins.pdf, 2000.

[24] Andrew Hopkins. *Lessons from Longford: The Esso Gas Plant Explosion.* CCH, Sydney, 2000.

[25] David Hume. *An Enquiry Concerning Human Understanding.* Oxford University Press, 1777/1975. Third Edition.

[26] International Civil Aviation Organization, ICAO. *Annex 13 to the Convention on International Civil Aviation: Aircraft Accident and Incident Investigation.* ICAO, 2010.

[27] International Electrotechnical Commission. Committee Draft, IEC 62740 Root Cause Analysis (RCS), Edition 1, IEC TC 56 WG3 PT3.23. June 2012.

[28] Canadian Transportation Accident Investigation and Safety Board. Transportation Safety Board. http://www.tsb.gc.ca/eng/publications/rmr-dpr/2004/rmr-dpr-2004-6.asp, 2012.

[29] Kaoru Ishikawa. *What is total quality control?* Prentice-Hall, 1985.

[30] Jens Braband, Ernesto de Stefano, Sonja-Lara Kurz (Siemens TS). Comparison of Event-Based Root Cause Analysis Models. http://www.rvs.uni-bielefeld.de/Bieleschweig/third/Stefano-Kurz-Braband-B3-2004.pdf, 2004.

[31] Jens Braband, Luke Emmet, Peter B. Ladkin, Claire Blackett, I Made Wiryana, Fergus Toolan, Oliver Lemke, Jan tecker-Gayen, Timm Grams. Comparison Criteria. http://www.rvs.uni-bielefeld.de/Bieleschweig/criteria/, 2002.

[32] Chris W. Johnson. University of Glasgow Press, 2003.

[33] Peter B. Ladkin. http://www.rvs.uni-bielefeld.de/publications/Papers/ladkin_WBG_Comparison.pdf.

[34] Peter B. Ladkin. http://www.rvs.uni-bielefeld.de/publications/Papers/Ladkin-Glenbrook.pdf.

[35] Peter B. Ladkin. Causal analysis of aircraft accidents. *Lecture Notes in Computer Science*, 2000.

[36] Peter B. Ladkin. Causal system analysis. http://www.rvs.uni-bielefeld.de/publications/books/CausalSystemAnalysis/index.html, 2001.

[37] Oliver Lemke. http://www.rvs.uni-bielefeld.de/research/WBA/Neufahrn_WBA-Vergleich_1.0.0.pdf.

[38] Nancy G. Leveson, 2012.

[39] David Lewis. *Counterfactuals.* Oxford University Press, 1973.

[40] Lyle Masimore. Proving the Value of Safety - Justification and ROI of Safety Programs and Machine Safety Investments, 2007. Publication SAFETY-WP004B-EN-P July 2007.

[41] Lars Molske. Controlled english for why-because analysis software. http://www.causalis.com/index.php?id=65&file=1A3B0&no_cache=1&uid=57, 2005. This is a link to the software. Lars Molske's Diploma Thesis is not available to me.

[42] John Moncenigo. http://www2.research.att.com/ john/Grappa/, 2012.

[43] Roger Needham. Naming. *Distributed Systems*, pages 318 – 327, 1993.

[44] Paul S. Nelson. http%3A%2F%2Fsunnyday.mit.edu%2Fpapers%2Fnelson-thesis.pdf, 2008.

[45] Damian Nowak. Iqualizeit software. http://www.causalis.com/iqualizeit.zip, 2005. This is a link to the software. Damian Nowak's Diploma Thesis is not available to me.

[46] Bureau of Transportation Statistics. Safety in numbers - safety data action plan. Technical report, U.S. Department of Transportation, 2000.

[47] Thilo Paul-Stueve. A practical guide to the why-because analysis method - performing a why-because analysis. www.rvs.uni-bielefeld.de/research/WBA/WBA-Guide.pdf, 2005.

[48] J. Raskin. *The Humane Interface: New Directions for Designing Interactive Systems.* ACM Press Series. Addison-Wesley, 2000.

[49] J. Reason. *Human Error.* Cambridge University Press, 1990.

[50] Jan Sanders. http://www.rvs.uni-bielefeld.de/research/WBA/WBA_Introduction.pdf.

[51] Bernd Sieker, Michael Hoehl, Jan Paller, and Jan E. Hennig. cid2ft, cid2dot, wb2dot, ybedit, ybfactor, vdas. http://www.rvs.uni-bielefeld.de/research/WBA/#Older, 2000.

[52] Joern Stuphorn. http://www.rvs.uni- biele-feld.de/Bieleschweig/5.5/Stuphorn Ueberlingen WBA.pdf.

[53] MyBatis Team, 2012.

[54] Karsten Loer Thorsten Gerdsmeier, Peter B. Ladkin. Formalising failure analysis. http://www.rvs.uni-bielefeld.de/publications/Reports/AMAST97.html, 2003.

[55] Main Commission Aircraft Accident Investigation Warsaw. Report on the accident to airbus a320-211 aircraft in warsaw. http://www.rvs.uni-bielefeld.de/publications/Incidents/DOCS/ComAndRep/Warsaw/warsaw-report.html, 1994. As made available in digital form by Peter B. Ladkin on the RVS Website in 1996.