Safety Critical Software
: Status Report

By
_200711429 박상욱
_200711439 송인하
_200711444 양동은

B class 3T

Contents

Section Header

1. INTRODUCE

1.1 Purpose of this report

- To bring together concepts necessary for the development of software in safety-critical systems.
- To know the role of safety-critical software in requirements engineering.
- To deal of recent activity in the application of formal methods to safety-critical software development.
- To introduce the classes of formal methods and how they may be used.

1.2 Requirements Engineering and Safety

+ The expensive required development techniques
+ A limited number of the staff

+ Minimize the proportion of the developed system according to safety standard

+ The Requirements engineer has to manipulate the requirements to minimize the safety-critical subsystems while maintaining an overall required level of safety for the entire system

1.2 Requirements Engineering and Safety

- Safety concerns often conflict with other development concerns, such as **performance or cost.**

>> **After performing an analysis of the risk associated with the resultant system, decision should be made during development for reasons of performance or cost that compromise safety**

- The safety of a system is considered by understanding **the potential hazards of the system.**

>> **If the system is analyzed iteratively in terms of the safety hazards, it leads to a hierarchy of safety specifications**

1.2 Requirements Engineering and Safety

- A well-designed system will have few safety-critical components in proportion to the total system.
- Safety-critical components require **a system-level**, rather than a component-level.

>> **Safety must be considered from the start in the development of system**

1.2 Requirements Engineering and Safety

- A requirements engineering has to eliminate **some errors** in the requirement caused by misunderstanding customer desires or poorly conceiving customer requests.

>> **The requirement engineering process must analyze the requirements for both desirable and undesirable behaviors**

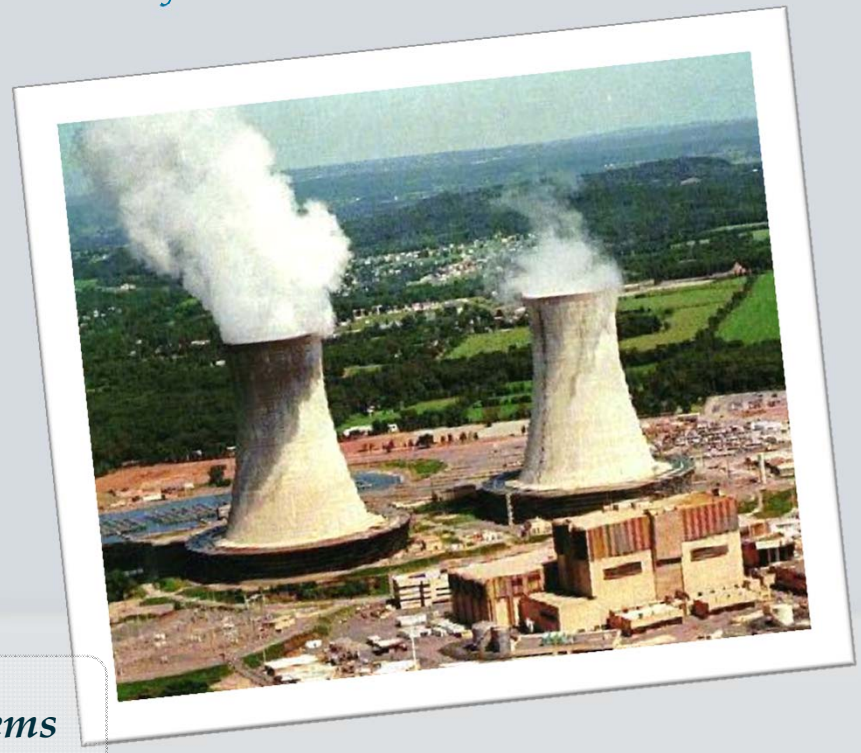
1.2 Requirements Engineering and Safety

- Safety can't be determined by examining the safety of the components in **isolation.**

>> **If not, the system will never be safe since the model is used as the basis for analysis and further development. Although each of the individual components may be safe, the integrated system may not be safe**

1.2 Requirements Engineering and Safety

- Some systems may be untestable for safety in a live situation.



$\overrightarrow{ex}$) *Nuclear power plant shutdown systems*

1.2 Requirements Engineering and Safety

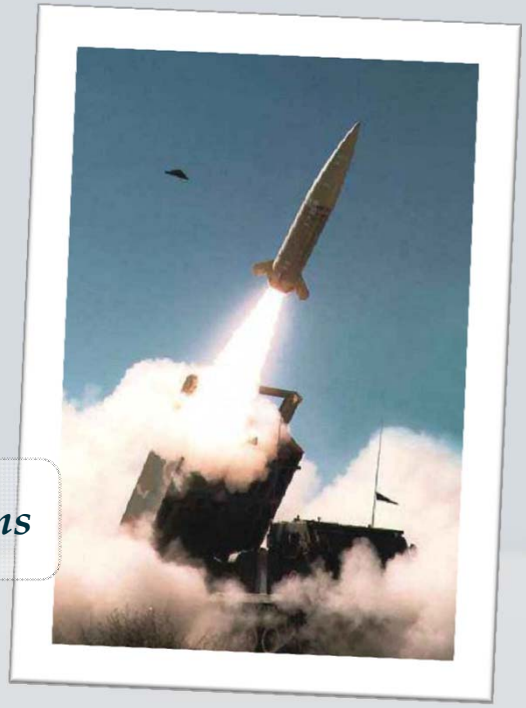
- Some systems may be untestable for safety in a live situation.

$\overrightarrow{ex}$) *Aircraft flight control systems*

1.2 Requirements Engineering and Safety

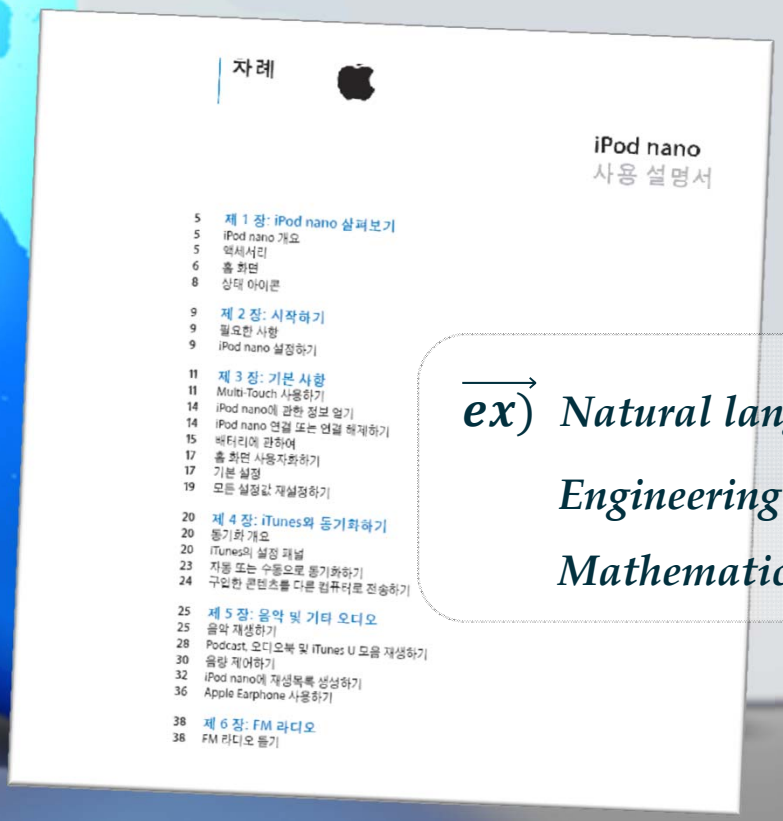
- Some systems may be untestable for safety in a live situation.

$\overrightarrow{ex}$) *Critical components of strategic weapons systems*



1.2 Requirements Engineering and Safety

- Customer's requirements are organized into a coherent form that may be analyzed in a cost-effective manner.


자례

iPod nano
사용 설명서

$\overrightarrow{ex}$) *Natural language descriptions /*
Engineering diagrams /
Mathematics

1.3 Background

- The use of software is increasing in safety-critical components of systems being developed and delivered.
- Therac 25. (a therapeutic linear accelerator)
- Nuclear reactor shutdown systems.

- Software failure is less predictable than hardware failure.

Section Header

2. COMMENTS ON SOFTWARE SAFETY

2.1 Safety is a System Issue

- Software is the control of systems with hazardous components, or the providing of information to people who make decisions that have potentially hazardous consequences, that leads to hazardous systems.

>> **Software can be considered unsafe only in the context of a particular system.**

- At the system level, software may be treated as one or more components whose failure may lead to a hazardous system condition.

2.2 Safety is Measured as Risk

- We in inherently understand what we mean when we say, "This system is safe." Essentially, we mean that it will not cause harm either to people or property.

- There are many systems that can be made completely safe, but making systems that safe may interfere with their ability to perform their intended function.

$\overrightarrow{ex}$) ***A Nuclear Reactor***

- the system is perfectly safe so long as no nuclear material is introduced into the system.

2.2 Safety is Measured as Risk

- The definition of safety becomes related to risk.

$$Risk = \sum_{hazard} E_{hazard} \; X \; P_{hazard}$$

: E_{hazard} is a measure of the effects that may be caused by a particular mishap

: P_{hazard} is the probability that the mishap will occur

2.2 Safety is Measured as Risk

- The point we must accept is that no system will be wholly safe.

So we have to

- **minimize the risk by either containing the hazard**
- **reducing the probability that the hazard will occur**

2.3 Reliability is not Safety

- It is important to distinguish between the terms reliability and safety.
- According to definitions from Deutsch and Willis,

; Reliability is measure of the rate of failure in the system that renders the system unusable

; Reliability describes how well the system performs its function

; Safety is a measure of the absence of unsafe software conditions

; Safety states that the system functions do not lead to an accident

2.3 Reliability is not Safety

- A system may be reliable but unsafe.
- An example is an aircraft avionics system that continues to operate under adverse conditions such as component failure, yet directs a pilot to fly the aircraft on a collision course with another aircraft.

So

+ The system itself may be reliable

+But, leads to accident

2.3 Reliability is not Safety

- A system may be safe but unreliable.
- An example is a railroad signaling system that may be wholly unreliable but safe if it always fails in the most restrictive way; in other words, whenever it fails it shows “stop”.

So ⇒
- **+ The system is safe**
- **+ But it is not reliable**

2.4 Software need not be Perfect

- We consider software to be perfect if it contains no errors, where an error is a variance between the operation of the software and the user's concept of how the software should operate.

- The notion of perfection considers all error equal.
- Any error means that the software is imperfect.

- From a safety viewpoint, only errors that cause the system to participate in an accident are of importance.

2.4 Software need not be Perfect

- There may be gross functional divergence within some parts of the system, but if these are masked, or ignored by the safety components, the system could still be safe.

An example

+ **Consider a nuclear power plant using both control room software and protection software**

+ **The control room software could, potentially, contain many errors, but as long as the protection system operates, the plant will be safe**

- It may be not economical, it may never produce any power, but it will not be an agent in an accident.

2.4 Software need not be Perfect

- A system such as the protection system, some bugs can be tolerated from the strictly safety viewpoint.

An example

+ **The protection system might always attempt to shut down the reactor, regardless of the condition of the reactor**

+ **The system is not useful, it contains gross functional divergence, yet it is safe**

- This should be contrasted with a protection system that never attempts to shut down the reactor regardless of reactor of reactor condition.

- The view that software need not be perfect to ensure safety of the entire system means that developers and analysts of safe software can concentrate their most detailed scrutiny on the safety conditions and not on the operational requirements.

2.5 Safe software is Secure and Reliable

- If the system is unreliability, a failure could occur such that the system's security is compromised.
- The safety-critical components of a system need to be secure since it is important that the software and data can't be altered by external agents.
- If the system is unreliability, it could fail to perform at any time when the software is needed to avoid a mishap.

2.6 Software should not Replace Hardware

+ One of the advantages of software is that it is flexible and relatively easy to modify
+ An economic advantage of software is that once it has been developed, the reproduction costs are very low

+ Hardware may be quite expensive to reproduce and is, in terms of production costs, the most expensive part of a system

+ Thus, economic viewpoint, there is considerable temptation to replace hardware components of a system with software analogs.

However, there is a danger to this approach that leads to unsafe systems

2.6 Software should not Replace Hardware

- Hardware fails in more predictable ways than software, and a failure may be foreseen by examining the hardware – a bar may bend or show cracks before it fails. These indicators of failure may occur long enough before the failure that the component may be replaced before a failure leading to a mishap occurs.

- But, Software does not exhibit physical characteristics that may be observed in the same way as hardware, making the failures unexpected and immediate; thus may be no warning of the impending failure.

2.7 Development Software is also Safety-Critical

- Safety analysis of a system is performed on a number of artifacts created during the development of the system. Later stages in the development need not be analyzed under the following circumstances.

  + **The analysis of the current stage of the development shows that a system performing according to the current description is safe**

  + **There is certainty that any artifacts created in subsequent development stages precisely conform to the current description**

Section Header

3. HAZARD ANALYSIS TECHNIQUES

3. Hazard analysis Techniques

- Two aspects of the effort of performing a hazard check of a system.

Hazard identification	+ Delphi techniques + Joint application design (JAD) + Hazard and Operability Analysis
Hazard analysis	+ Fault tree analysis + Event tree analysis + Failure modes and Effects Analysis (FMEA)

- The analyst has to perform all hazard identification and subsequently analyzes the system to determine whether or not the hazards can occur or lead to a mishap, the two activities may well be mixed.

3.1 Hazard Identification

+ There does not appear to be any easy way to identify hazards within a given system

+ We have to attempt to develop a list of possible system hazards for identification before the system is built

+ A thorough understanding of the history of failures in the given domain is a necessary prerequisite to the development of the preliminary hazard list

+ And the experts need to understand the differences between the new system and previous systems so that they can understand the new failure modes introduced by the new system

3.1 Hazard Identification

- The project management may have to use some approach to ensure that they have the greatest likelihood of listing the system hazards.

+ To use “Brainstorming”

+ To use some guidelines to know

_ The Delphi Technique

_ Joint Application Design (JAD)

3.1 Hazard Identification

- The Delphi Technique.

Process

+ Send out a questionnaire to all members of the group

+ After the responses to the questionnaire have been received, the opinion are reproduced in such way that the author's identify is obscured and the opinions are collated

+ The collated opinions are sent out to the experts who justify any outlying opinions

3.1 Hazard Identification

- The Delphi Technique.

Key ideas

+ Anonymous responses
+ The connection is only through the questionnaires

3.1 Hazard Identification

- The Delphi Technique.

Advantage	+ Someone particularly strong personality can't sway the opinion of the entire group through force of will + When the group is unable to attend a meeting in a JAD method, this method overcomes the issue of group consensus
Disadvantage	+ This method makes for slow communication and it may take several weeks to arrive at consensus

3.1 Hazard Identification

+ Joint Application Design.

Process

+ The group must be made up of people with certain characteristics

  - members must be skilled and empowered to make decisions
  - the right number of people between six and ten is involved in a JAD session

+ Be led by a facilitator who should have no vested interest in the detailed content of design and be chosen for reasons of technical ability and skills in communication and for ability to maintain control over a group

+ The ideas must be captured immediately and become owned by the group rather than individuals

  - the facilitator must capture any ideas and display for all to see Take place in a neutral location

3.1 Hazard Identification

- Joint Application Design.

Key ideas

+ Appropriate number's members with certain characteristics
+ A neutral facilitator
+ A neutral place
+ An immediately captured idea and a display

3.1 Hazard Identification

- Joint Application Design.

Advantage	+ Make for an alternative fast communication + Take fewer interruptions
Disadvantage	+ All ideas of group members don't affect decisions of group + The facilitator can become a bottleneck

3.1 Hazard Identification

- Hazard and Operability Analysis.

+ This applies at all stages of the development life cycle and is used to ensure a systematic evaluation of the functional aspects of the system

  First, the designers identify their concepts of how the system should be operated

  Second, the designers determine when the identified conditions can become safety-critical

This analysis is an iterative process that should be started before any detailed design. It should be continually updated as system design progresses

3.2 Hazard Analysis

- The purpose is to examine the system and determine which components may lead to a mishap.
- Two basic strategies.

+ The deductive techniques

- Fault tree analysis

+ The inductive techniques

- Event tree analysis

- Failure modes and effects analysis

- The inductive methods are applied to determine what system states are possible and the deductive methods are applied to determine how a given state can occur.

3.2 Hazard Analysis

+ Deductive techniques : Fault Tree Analysis.

+ It starts with a particular undesirable event and provides an approach for analyzing the cause of this event

+ It is a depiction of the logical interrelationships of basic events (component failures, human failures, some random event in the environment and etc.) that may lead to a particular undesired event

+ It can be an expensive and time-consuming process

+ Domain expertise is necessary since provides the knowledge of how similar systems have failed in the past

+ It is possible to create a fault tree template that may be used as necessary within a fault tree

3.2 Hazard Analysis

- Deductive techniques : Fault Tree Analysis.

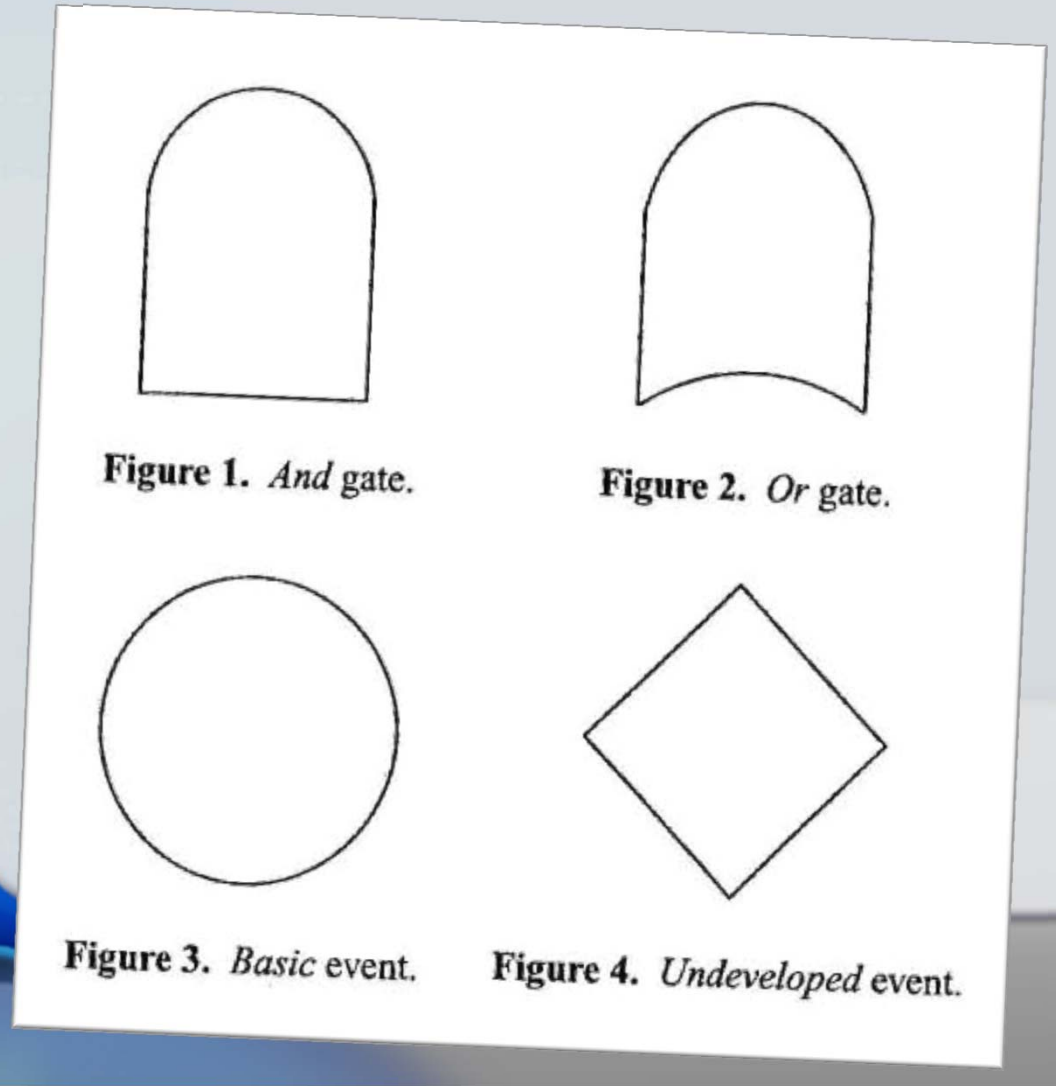


Figure 1. And gate.

Figure 2. Or gate.

Figure 3. Basic event.

Figure 4. Undeveloped event.

3.2 Hazard Analysis

- Deductive techniques : Fault Tree Analysis.

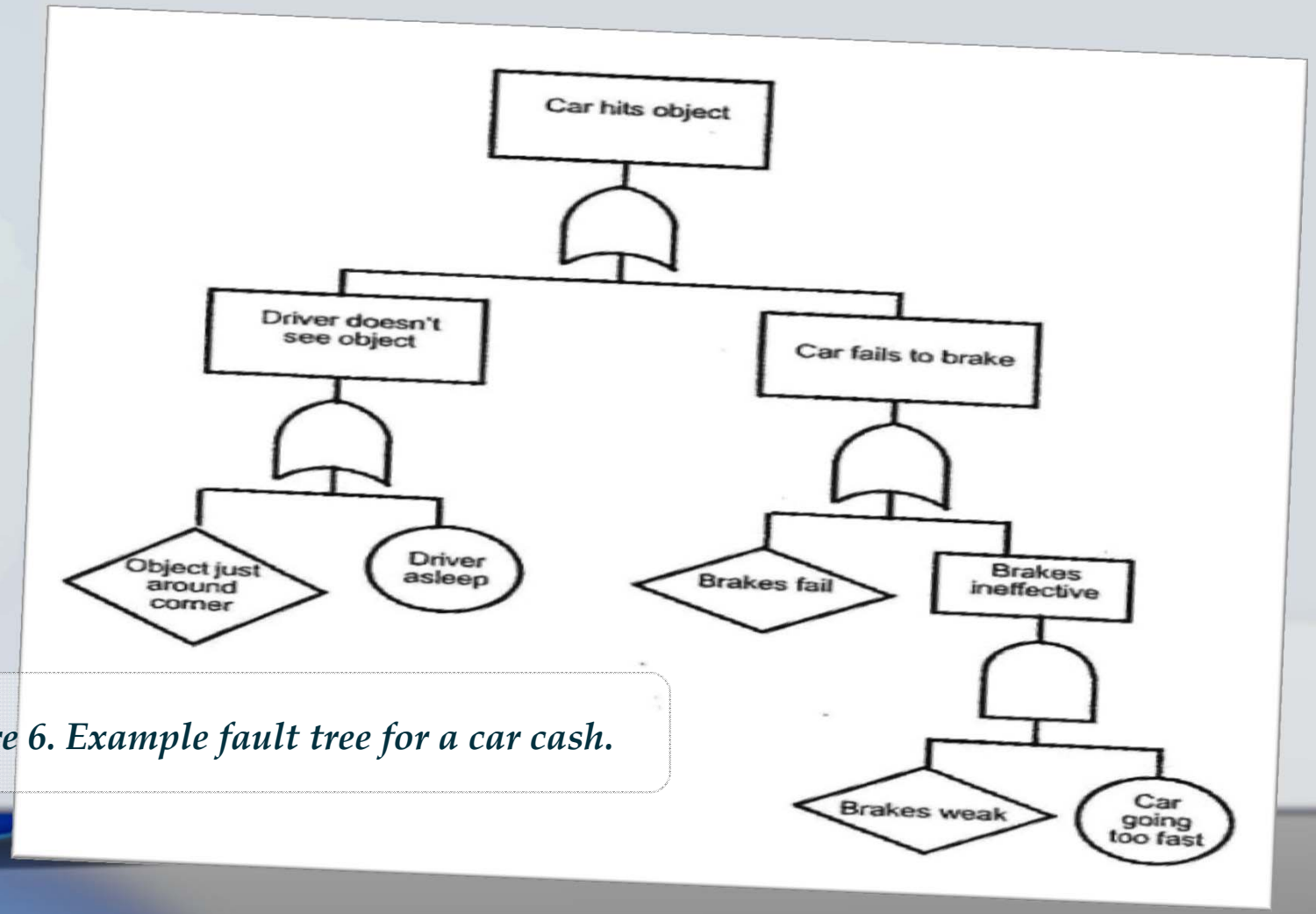


Figure 6. Example fault tree for a car cash.

3.2 Hazard Analysis

+ Deductive techniques : Fault Tree Analysis.

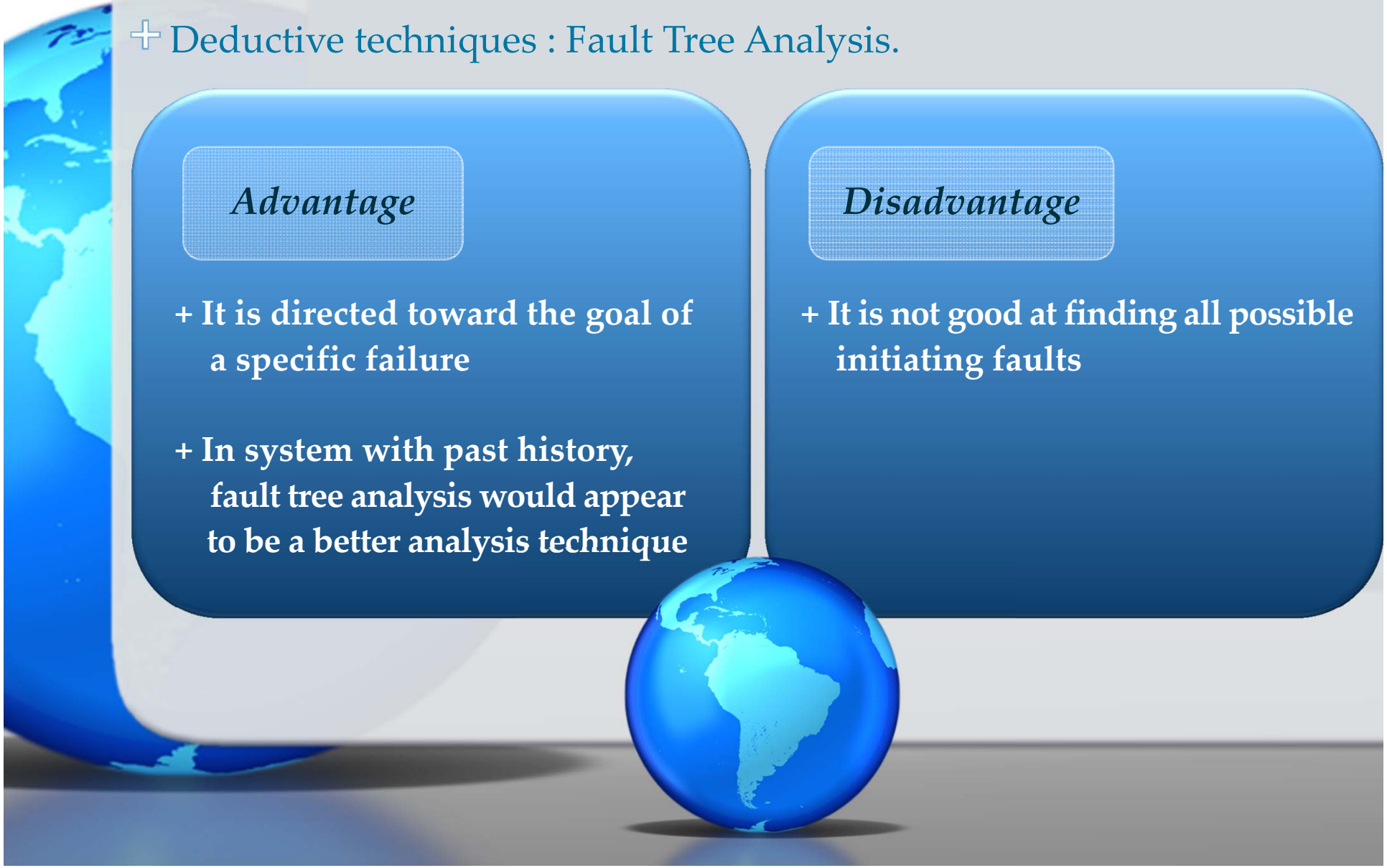
Advantage

+ It is directed toward the goal of a specific failure

+ In system with past history, fault tree analysis would appear to be a better analysis technique

Disadvantage

+ It is not good at finding all possible initiating faults



3.2 Hazard Analysis

+ Inductive techniques : Event Tree Analysis.

+ To analyze effects caused by initialing events

+ Its approach is to consider an initialing event and its possible consequences, then for each of these consequential events in turn, the potential consequences are considered, thus drawing the tree

+ The initialing events for event tree analysis may be both desirable and undesirable

> The choice of initiating events is the range of events is the range of events that may occur in the system

+ It is forward-looking and considers potential future problems

3.2 Hazard Analysis

+ Inductive techniques : Event Tree Analysis.

Advantage

+ In wholly new system, it may play a valuable role since the consequences of failures may be analyzed to determine if a mishap might occur

Disadvantage

+ It deals with a large part due to the difficulty of considering all of the possible of consequences of an event.
+ It may become large and unmanageable rapidly without discovering a possible mishap
+ Much analysis time may be wasted by considering an event tree from a given event

3.2 Hazard Analysis

- Inductive techniques : Failure Modes and Effects Analysis.

+ It is known as FMEA

+ FMEA consists of constructing a table based on the components of the system and the possible failure modes

+ It is not an additional technique

+ Its approach is to create a table with failure, occurrence, severity, probability of detection, risk priority number, and corrective action

3.2 Hazard Analysis

+ Inductive techniques : Failure Modes and Effects Analysis.

Table 1. Example failure modes and effects analysis table

Component	Failure mode	Effect of failure	Cause of failure	Occ-urrence	Severity	Probability of detection	Risk priority number	Corrective action
Tie bar bracket	Bracket fractures	Stabilizing function of tie bar removed. All engine motion transferred to mountings	Inadequate specification of hole-to-edge distance	1	7	10	70	Test suitability of specification
	Bracket corrodes	As above	Inadequate specification for preparation of bracket	1	5	10	50	Test suitability of specification
	Fixing bolts loosen	As above	Bolt torque inadequately specified	5	5	8	200	Test for loosening
			Bolt material or thread type inadequate	1	5	10	50	Test suitability of specification

+ A closely related approach is a FMECA that provides a more formal process for performing the criticality analysis

Hazard analysis Techniques

- First approach.

+ Create a list of all hazards and for those with a sufficiently high risk

+ Perform fault tree analysis

Then, continue to apply hazard analysis techniques at each of development

Hazard analysis Techniques

- Second approach.

+ Perform a FMEA, potentially using fault tree and event tree analysis

+ Employ the best development techniques for components with unacceptably high criticality factor

Section Header

4. Summary

1. Introduction of safety critical
2. Software safety
3. hazard identification and analysis


The End
B class 3T
_Thanks for your attentions!