Dependable Systems

Qualitative Dependability Evaluation

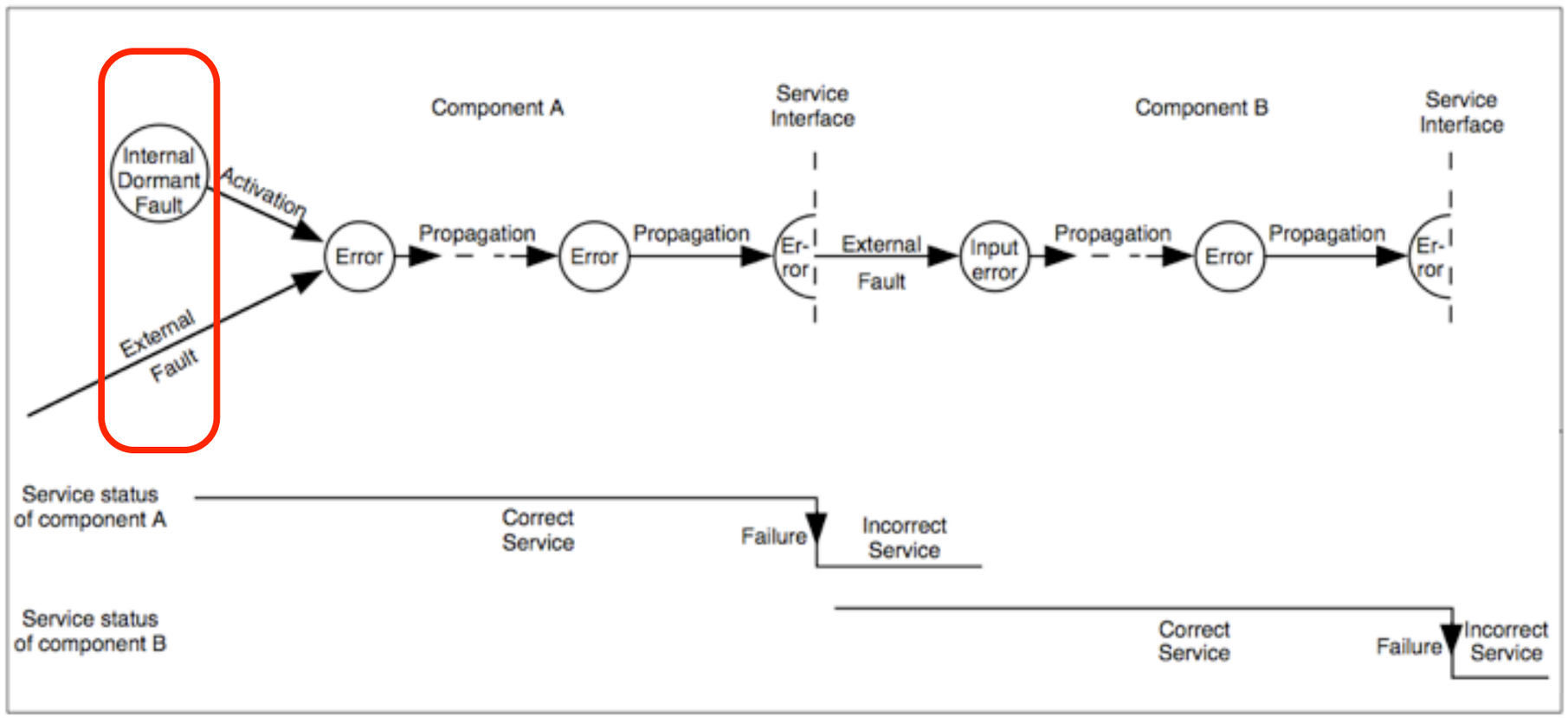
Dr. Peter Tröger

Qualitative Dependability Investigation

- Different approaches that focus on structural (qualitiative) system evaluation
  - Root cause analysis
    - Broad research / industrial topic about error diagnosis approaches
    - Specialized topic in quality methodologies
  - Development process investigation
    - Procedures for ensuring industry quality in production
    - Software development process
- Organizational investigation
  - Non-technical influence factors on system reliability

Root Cause Analysis

- What caused the fault ? - Starting point of dependability chain
  - Peeling back the layers
  - Must be performed systematically as an investigation
  - Establish sequence of events / timeline



(C) Avizienis

Root Cause Analysis

- Class of approaches and algorithms for identifying root causes of problems
  - Iterative process of continuous improvement
  - Can be performed in reactive or pro-active fashion
- Applied in different fields, competing definitions and understandings
  - Accident analysis in safety-critical systems, e.g. aviation industry
  - Quality control in industrial manufacturing
  - Investigation of formally described business processes
  - Hardware failure analysis
  - Risk management for huge hardware / software projects
- Assumes broken process or alterable cause (e.g. no physical faults)

RCA: 5 Whys

- Originally developed by Sakichi Toyoda for car manufacturing process
  - Limit number of dive-in‘s to avoid tracing the chain of causality
- Example: The Washington Monument was disintegrating.
  - Why? Use of harsh chemicals.
  - Why? To clean pigeon poop.
  - Why so many pigeons? They eat spiders and there are a lot of spiders at monument.
  - Why so many spiders? They eat gnats and lots of gnats at monument.
  - Why so many gnats? They are attracted to the light at dusk.
  - Solution: Turn on the lights at a later time.

5 Whys

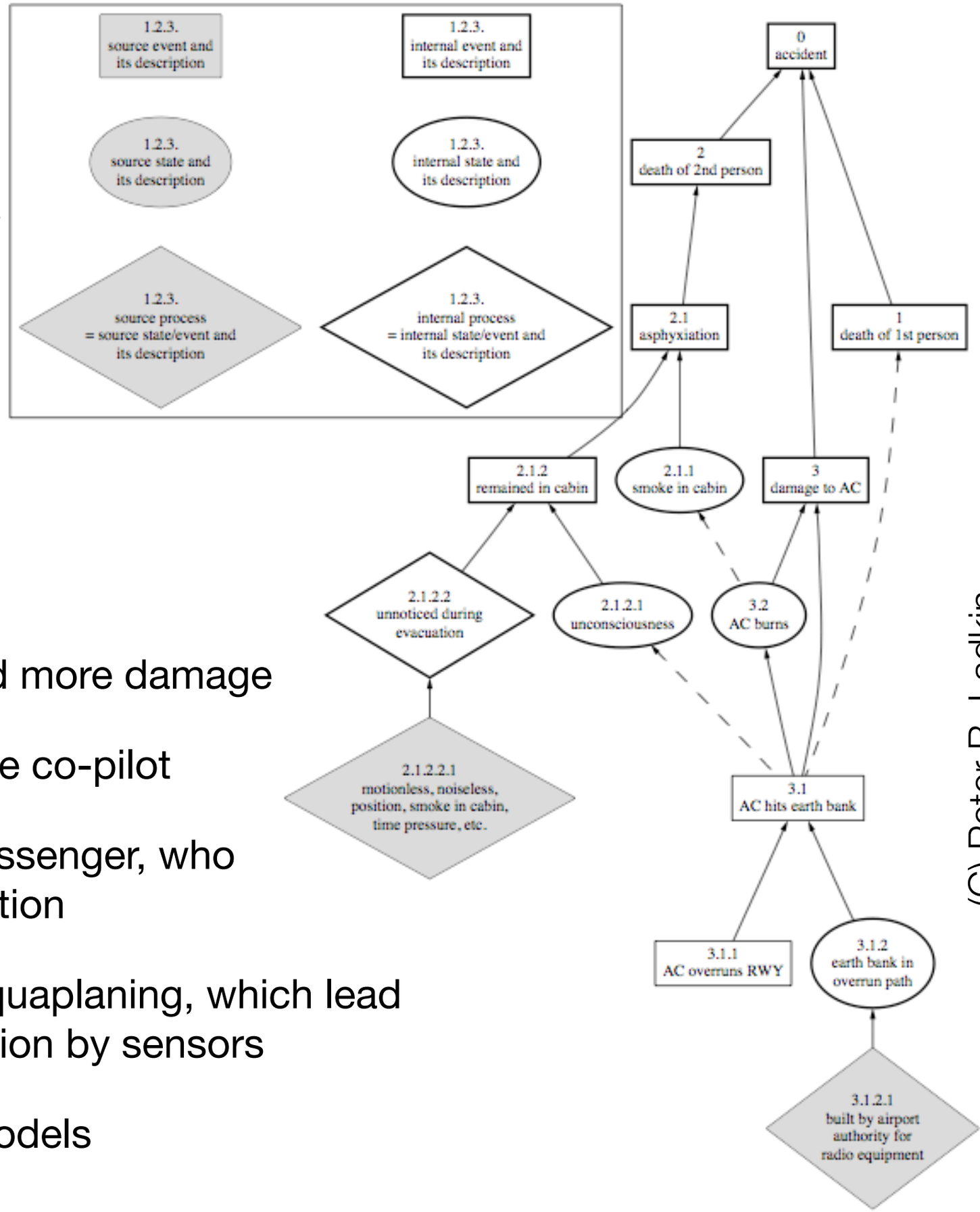
- Limited steps ensure that investigator moves through layers
- Has danger of stopping too early at symptoms
- Results are not reproducible
- Investigators cannot consider reasons behind their own information - need teams
- Recommendation to use observation instead of deduction
  - Deduction does not allow proper ranking of answers
- Depends on completely honest answers and complete problem statement
- Only good for low risk problems

RCA: Why-Because Analysis

- Mainly intended for accident analysis (train, bus, airplane, security, industry)
- Central notion of a causal factor
- Output is directed acyclic **Why-Because graph (WBG)**
  - Showing causal connections between all events and states of behavior
  - Nodes express causal factors, arcs express cause-effect relationship
- Different subgraphs can be tested, results are combinable
  - **Causal sufficiency test** -
    Will an effect always happen if all parent causes happen ?
  - **Counterfactual test** (Dawid Lewis 1975, philosophical logician) -
    If the cause would not have existed, could the effect still have happened ?
    - If „no“ for two effects, then the cause is a **necessary causal factor** (NCF)
    - *Nearest possible world* assumption

WBG Example

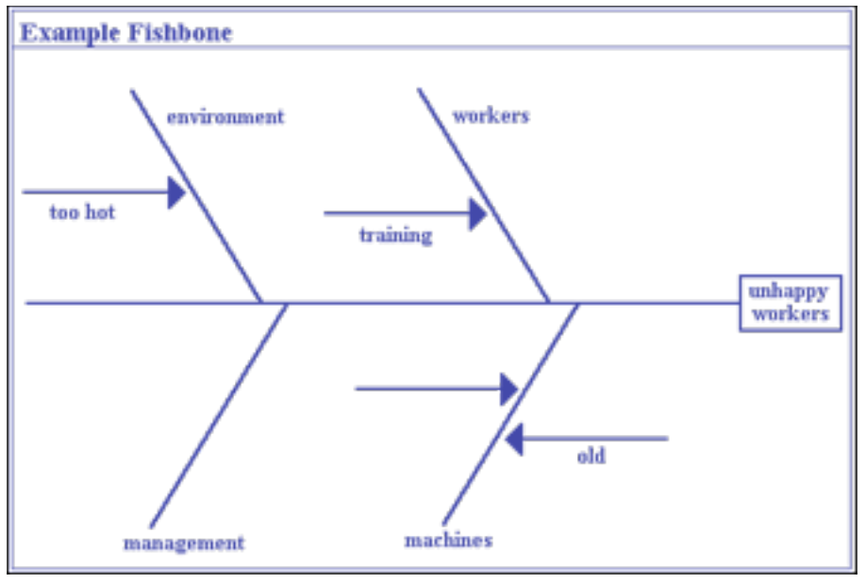
- 1993 Lufthansa A320-200 overrun accident in Warsaw
- Precipitating event is node 3.1
  - AC hits earth bank
  - Caused damage to the aircraft
  - Resulted in a fire which caused more damage
  - Caused the trauma death of the co-pilot
  - Loss of consciousness of a passenger, who was overlooked during evacuation
- But: Overrun was reasoned by aquaplaning, which lead to missing ground contact detection by sensors
  - Change of software in A320 models



(C) Peter B. Ladkin

RCA: Ishikawa / Fishbone Diagram

- Invented by Japanese quality control statistican for quality design and failure prevention, by identifying (categorized) sources for variation
- Analysis tool for systematic cause-and-effect analysis
  - List problem to be studied in the ‚head of the fish‘
  - Label main ‚fish bones‘
    - 4 M‘s: Methods (process), machines (technology), materials (raw, consumables), manpower (physical and brain work)
    - 4 P‘s: Place, procedure, people, policies
    - 4 S‘s: Surroundings, suppliers, systems, skills
  - Identify factors per category that may affecting the problem / issue
  - Repeat with sub-factors



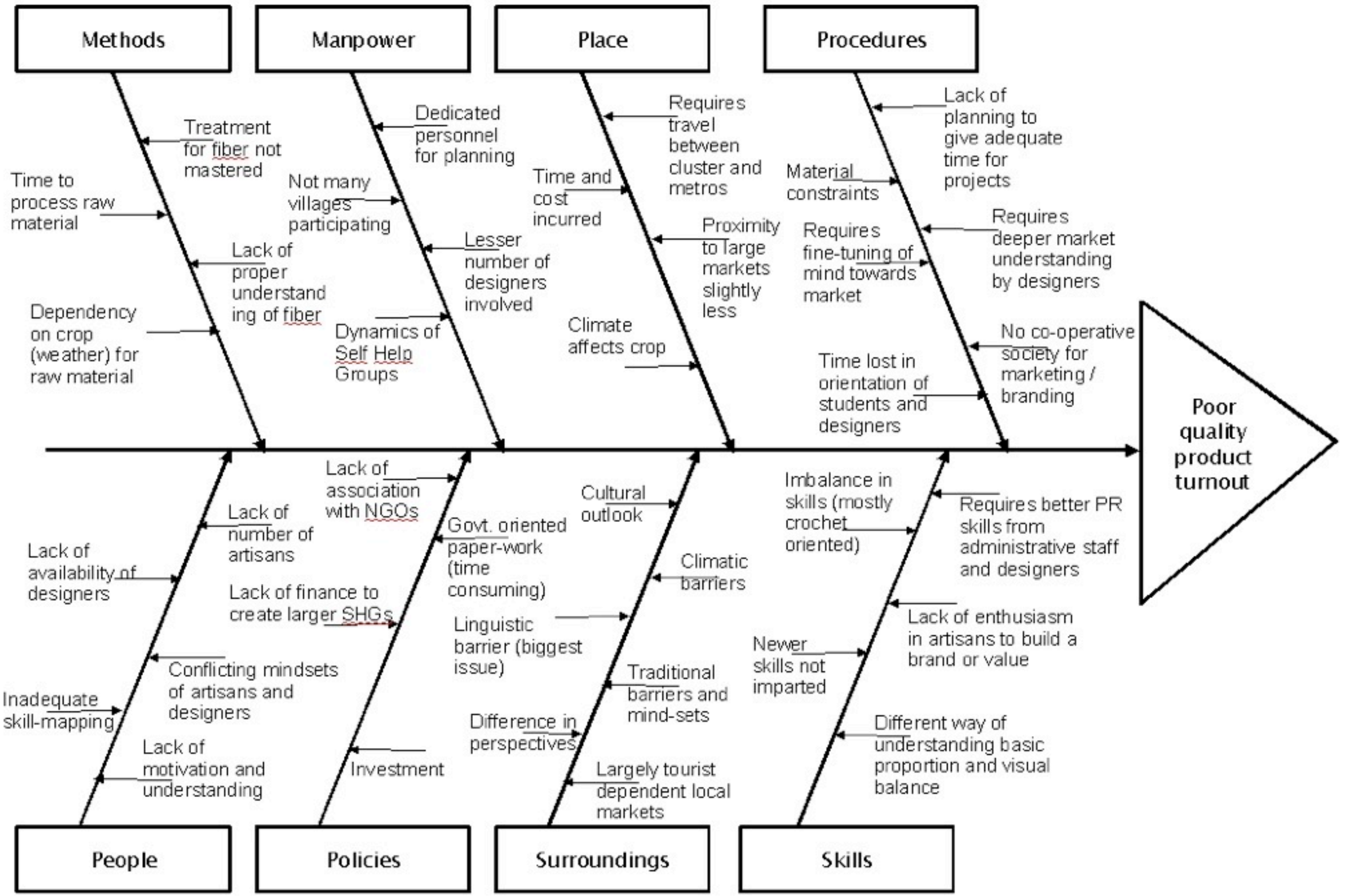


Fig: Application of Fish-bone diagram to craft based workshops in Indian design

FMEA

- **Failure Mode and Effects Analysis**
- Engineering quality method for early concept phase - identify and tackle weak points
- Introduced in the late 1940s for military usage (MIL-P-1629)
  - Later also used for aerospace program, automotive industry, semiconductor processing, software development, healthcare, ...
- Main goal is to identify and prevent critical failures
  - Outcome are actions to reduce the severity of such failures
- Used in many formal quality improvement programs
- Not a root cause analysis
  - Does not look on past events, but on a process - „how“ vs. „why“

FMEA

- Main assumption: System is vulnerable to certain failure modes (error states)
  - Examples: Electrical short-circuiting, corrosion, deformation
  - Identify relevant **failure mode** candidates based on past experience
  - **Effect analysis** for (specific) failure modes - what happens to the functionality visible to the end user ?
    - Examples: Degraded performance, noise, injury
  - Top-Down FMEA: Start with one function failure and find according failure modes
    - Typical for certification procedure, were the undesired functional problems are specified by the requirement documents
  - Bottom-Up FMEA: Find all the effects caused by all failure modes
- Failures are prioritized according to their consequences, how frequently they occur, and how easily they can be detected

FMEA Steps

- Identify FMEA scope in cross-functional team (design, quality, testing, support, **...**)
- Identify **functions** in the investigation scope (verb + noun)
- Per function, identify all ways a failure could happen - **potential failure modes**
- Identify **consequences** / **effects** per failure mode - on the system itself, related systems, product, service, customers or regulations
- Perform **severity rating (S)** per effect - From insignificant to catastrophic, add only highest ranked effects to the further analysis
- Identify **root causes** per failure mode, rank by **probability of occurence (O)**
- List tests / procedures (**process controls**) that are in place to keep the causes away
- Determine **detection rating (D)** per control - from certain detection to no detection resp. no control possible
- **Risk priority number (RPN)** = S x O x D, **Criticality (C)** = S x O

Example: Bank FMEA

Function	Potential Failure Mode	Potential Effects(s) of Failure	S	Potential Cause(s) of Failure	O	Current Process Controls	D	RPN	CRIT	Recommended Action(s)	Responsibility and Target Completion Date	Action Results					
												Action Taken	S	O	D	RPN	CRIT
Dispense amount of cash requested by customer	Does not dispense cash	Customer very dissatisfied Incorrect entry to demand deposit system Discrepancy in cash balancing	8	Out of cash	5	Internal low-cash alert	5	200	**40**								
				Machine jams	3	Internal jam alert	10	240	24								
				Power failure during transaction	2	None	10	160	16								
	Dispenses too much cash	Bank loses money Discrepancy in cash balancing	6	Bills stuck together	2	Loading procedure (riffle ends of stack)	7	84	12								
				Denominations in wrong trays	3	Two-person visual verification	4	72	18								
	Takes too long to dispense cash	Customer somewhat annoyed	3	Heavy computer network traffic	7	None	10	210	21								
				Power interruption during transaction	2	None	10	60	6								

RPN prioritizes differently from criticality

Assign persons during the analysis

(C) asq.org

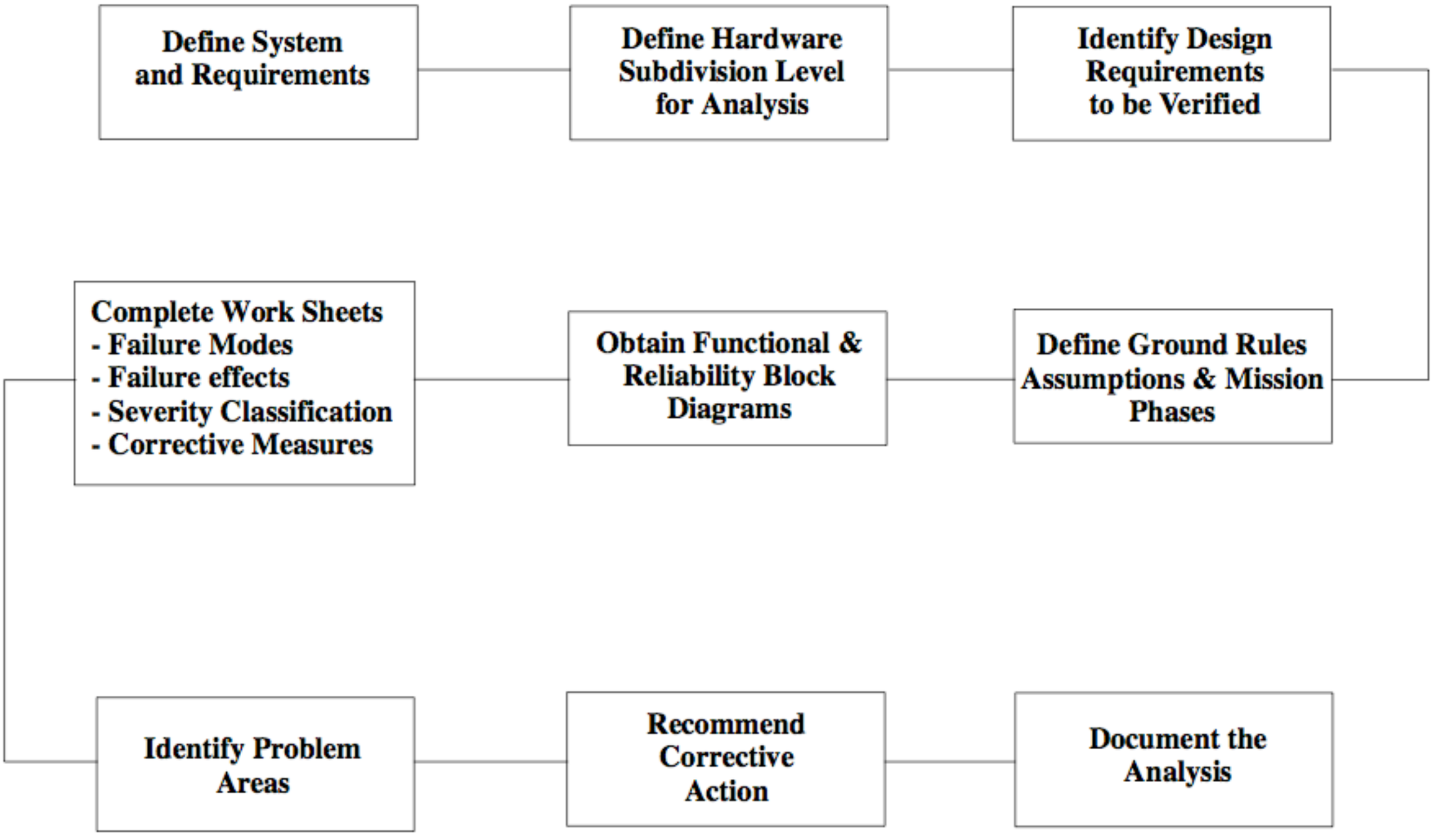
Example: NASA Spacecraft FMEA Severity Categories

- Category 1, **Catastrophic** - Failure modes that could result in serious injury or loss of life, or damage to the launch vehicle.
- Category 1R, **Catastrophic** - Failure modes of identical or equivalent redundant hardware items that, if all failed, could result in Category 1 effects.
- Category 2, **Critical** - Failure modes that could result in loss of one or more mission objectives as defined by the GSFC project office.
- Category 2R, **Critical** - Failure modes of identical or equivalent redundant hardware items that could result in Category 2 effects if all failed.
- Category 3, **Significant** - Failure modes that could cause degradation to mission objectives.
- Category 4, **Minor** - Failure modes that could result in insignificant or no loss to mission objectives.

Example: NASA Spacecraft FMEA Ground Rules

- Only one failure mode exists at a time
- All inputs (including software commands) to the item being analyzed are present and at nominal values
- All consumables are present in sufficient quantities
- Nominal power is available
- All mission phases are considered in the analysis; mission phases that prove inapplicable may be omitted
- Connector failure modes are limited to: connector disconnect
- Special emphasis will be directed towards identification of single failures that could cause loss of two or more redundant paths

Example: NASA Spacecraft FMEA Flow Diagram



FMEA

- Variations
  - DFMEA (D: Design)
    - Focus on failure modes reasoned by design deficiencies
    - Focus on parts that can be prototyped before high volume production
  - PFMEA (P: Process)
    - Analyze manufacturing and assembling processes
    - Influence design of machinery, selection of tooling and component parts
- Do not mix up design failures and causes („incorrect material specified“) with process failures and causes („incorrect material used“)
- FMEA typically makes use of fault tree modeling

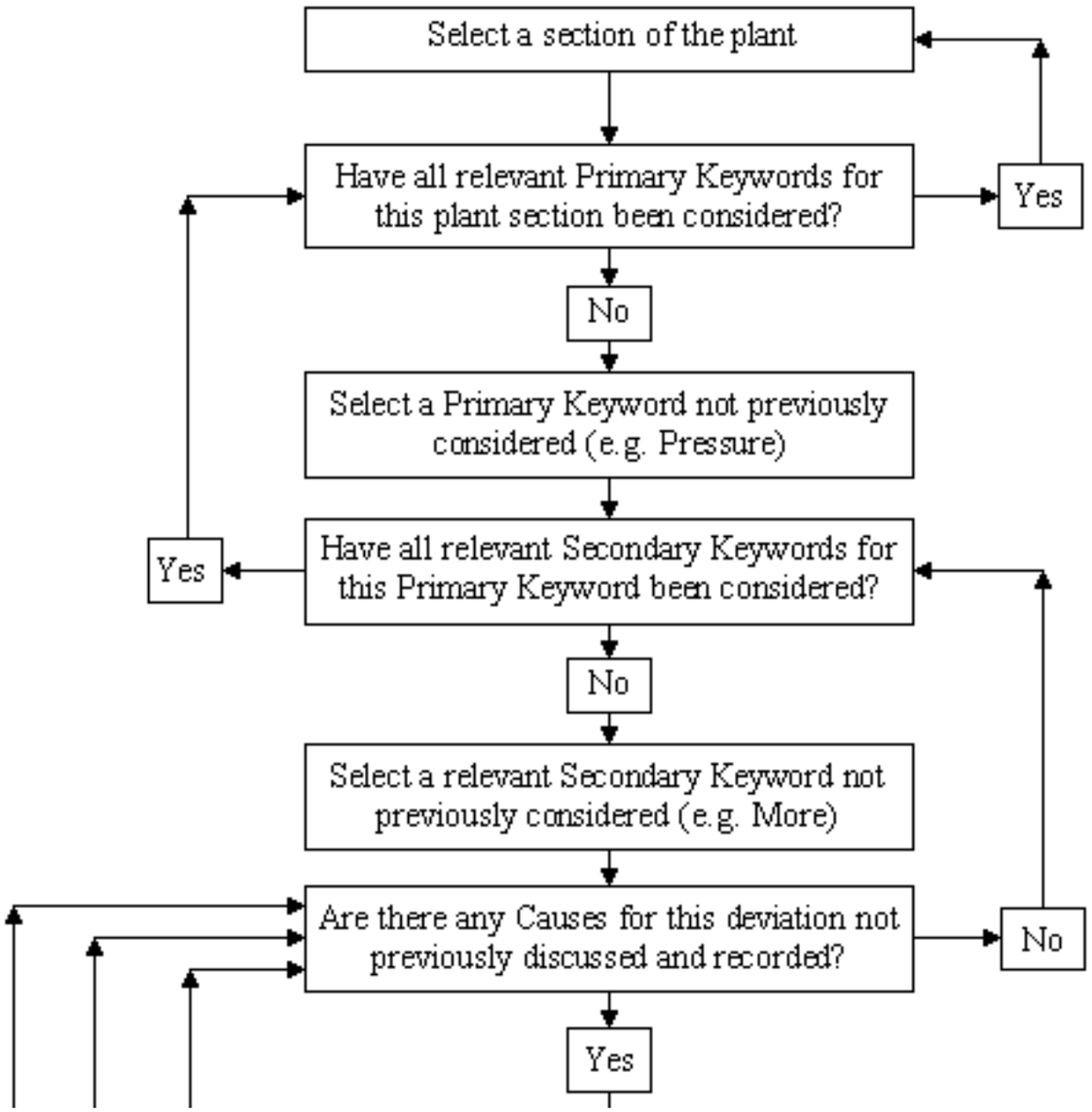
Hazard & Operability Studies (HAZOPS)

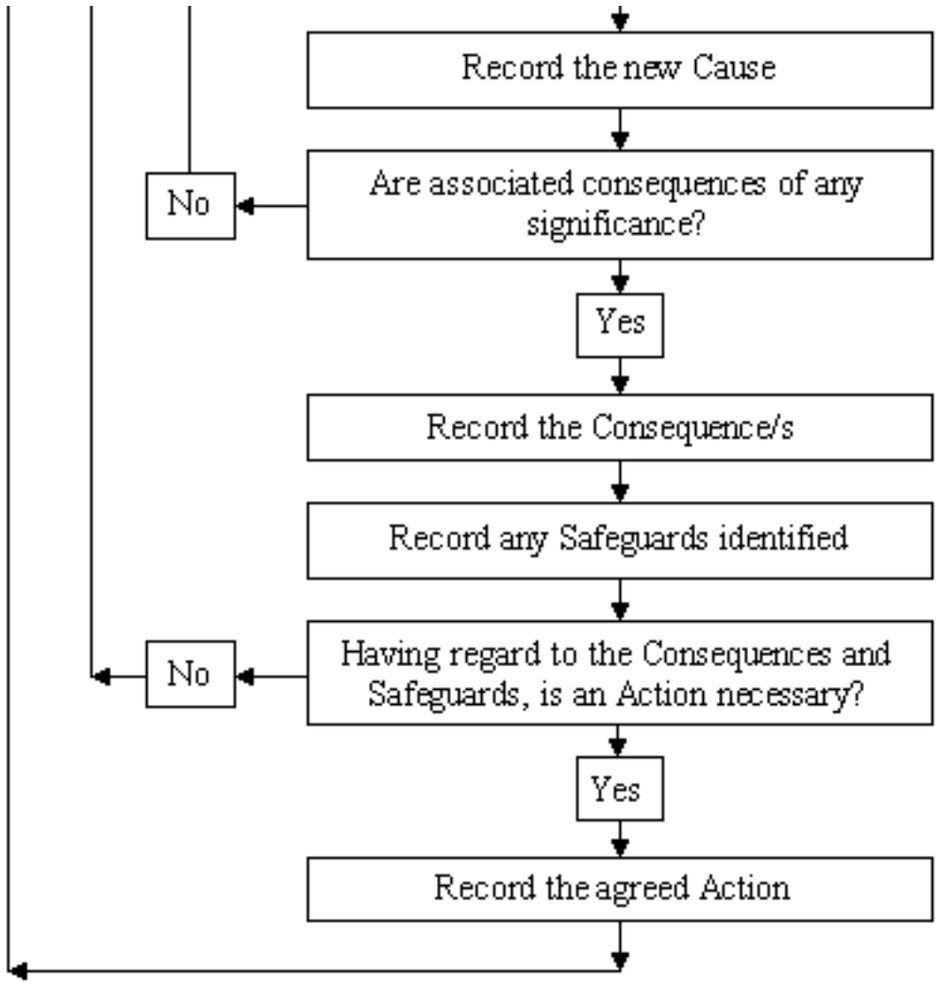
- Process for identification of potential hazard & operability problems caused by **deviations from the design intend**
  - Difference between deviation (failure) and its cause (fault)
  - Conduct intended functionality in the safest and most effective manner
  - Initially developed to investigate chemical production processes, meanwhile for petroleum, food, and water industries
  - Extended for complex (software) systems
- Qualitative technique
  - Take full description of process and systematically question every part of it
  - Assess possible deviations and their consequences
  - Based on guide-words and multi-disciplinary meetings

HAZOPS

- First identify system entities and their attributes (e.g. state diagrams)
- Use of keywords to focus the attention
  - **Primary keywords** - Focus attention on a particular aspect of the design intent / process condition / investigated parameter
    - Examples for plants: flow, pressure, react, corrode, temperature, level, mix, absorb, erode, ... isolate, vent, inspect, start-up, shutdown, purge, maintain, ...
    - Examples for Java Language Definition analysis: Type default value, type value range, name scope, class modifier, class name, field modifier, field type, method modifier, method name, formal parameter, ...
  - **Secondary keywords** - When combined with a primary keyword, suggest possible deviations of the system from the design intent
    - Standardized list, not all combinations are appropriate

HAZOPS Procedure





(C) www.lihoutech.com

Plant Example - Secondary Keywords

Word	Meaning
No	The design intent does not occur (e.g. Flow/No), or the operational aspect is not achievable (Isolate/No)
Less	A quantitative decrease in the design intent occurs (e.g. Pressure/Less)
More	A quantitative increase in the design intent occurs (e.g. Temperature/More)
Reverse	The opposite of the design intent occurs (e.g. Flow/Reverse)
Also	The design intent is completely fulfilled, but in addition some other related activity occurs (e.g. Flow/Also indicating contamination in a product stream)
Other	The activity occurs, but not in the way intended (e.g. Flow/Other could indicate a leak or product flowing where it should not)
Fluctuation	The design intention is achieved only part of the time (e.g. an air-lock in a pipeline might result in Flow/Fluctuation)
Early	Usually used when studying sequential operations, this would indicate that a step is started at the wrong time or done out of sequence
Late	

Plant Example - Combinations [Wikipedia]

Parameter / Guide Word	More	Less	None	Reverse	As well as	Part of	Other than
Flow	high flow	low flow	no flow	reverse flow	deviating concentration	contamination	deviating material
Pressure	high pressure	low pressure	vacuum		delta-p		explosion
Temperature	high temperature	low temperature					
Level	high level	low level	no level		different level		
Time	too long / too late	too short / too soon	sequence step skipped	backwards	missing actions	extra actions	wrong time
Agitation	fast mixing	slow mixing	no mixing				
Reaction	fast reaction / runaway	slow reaction	no reaction				unwanted reaction
Start-up / Shut-down	too fast	too slow			actions missed		wrong recipe
Draining / Venting	too long	too short	none		deviating pressure	wrong timing	
Inertising	high pressure	low pressure	none			contamination	wrong material
Utility failure (instrument air, power)			failure				

HAZOPS for Java Language Specification [Kim, Clark, McDermid] - Secondary Keywords

Word	Meaning
No	No part of the intention is achieved. No use of syntactic components
More	A quantitative increase, the data value is too high (within or out of bounds)
Less	A quantitative decrease, the data value is too low (within or out of bounds)
As Well As	Specific design intent is achieved but with additional results
Part Of	Only some of the intention is achieved, incomplete
Reverse	Reverse flow - flow of information in wrong direction, iteration count modified in wrong direction, logical negation of condition
Other Than	A result other than the original intention is achieved, complete but incorrect
Narrowing	Scope or accessibility is narrower than intended.
Widening	Scope or accessibility is enlarged
Equivalent	The same design intent is achieved in a different way (without any side effect)

HAZOPS Documentation

- Apply in a systematic way all relevant keyword combinations to the design
- Per combination, record
  - Deviation (keyword combination)
  - Cause (potential causes for the deviation)
  - Consequence (from both the deviation and the cause)
    - Should not take credit for protective systems or instruments in the design, since not all operational conditions are clarified at this point
  - Safeguards (which prevents the cause or safeguards the system against it)
    - Can be hardware, software, or procedures
  - Actions (which either remove the cause or mitigate the consequences)

HAZOPS for Java Language Specification [Kim, Clark, McDermid] - Deviants Example

FieldDeclaration:
FieldModifiers $_{opt}$ ***Type VariableDeclarators ;***

Attribute: Field modifiers	Guideword: OTHER_THAN
Causes	• The modifier `static` is specified where the field should be an instance variable. • The modifier `static` is not specified where the field should be a class (static) variable.
Consequences	A field becomes a class variable instead of an instance variable (or vice versa), causing different results or compile errors.
Attribute: Field modifiers	Guideword: MORE
Causes	The number of the specified modifiers increases.
Consequences	The behaviour of a field is changed/restricted or compile-time error occurs.
Attribute: Field modifiers	Guideword: LESS
Causes	The number of the specified modifiers decreases. (e.g. from 2 modifiers to 1 or no modifier)
Consequences	The behaviour of a field is changed or compile-time error occurs.
Attribute: Accessibility	Guideword: WIDENING
Causes	An access modifier is changed from `protected` to `public`, or from `private` to `public`.
Consequences	The fields that were not accessible become accessible. For example when a field is changed from `private` to default access, any entities within the same package can access the field either by ***SimpleName*** or ***QualifiedName***.
Attribute: Type compatibility of a field type	Guideword: AS_WELL_AS
Causes	• Class type T is declared instead of class type S, provided that S is a subclass of T. • Interface type K is used instead of interface type J, provided that J is a subinterface of K.
Consequences	Widening reference conversion regards a reference as having some other type.
Attribute: Type compatibility of a field type	Guideword: PART_OF
Causes	Class type S instead of class type T is declared, provided that S is a subclass of T.
Consequences	`ClassCastException` may arise if the actual reference value is not a legitimate value of the declared type at run time.

Process Evaluation

- Software reliability could be derived from level of trust into the development process
- Relevant for software supplier evaluation in public bidding
- Most famous approach is the **Capability Maturity Model (CMM)** by CMU Software Engineering Insitute (SEI), extended to **CMMI**
  - Five levels of process maturity, each level defines process areas to master
    - Level 2 (Repeatable) - requirement management, project planning and tracking, subcontract management, quality assurance, configuration management
    - Level 3 (Defined) - organization process, peer reviews, training program, intergroup coordination, product engineering
    - Level 4 (Managed) - qualitative management, quantitative process management
    - Level 5 (Optimizing) - process change management, technology change management, defect prevention

Reliability Models for IT Infrastructures

- System reliability in a commercial environment is determined by many factors:
  - Software and hardware reliability
  - Training of maintenance personnel
  - „Business processes“ how maintenance is handled
    - The way, the IT department is organized
- Impact of management organization on reliability is an emerging research field
- Standards for IT organization, based on best practices
  - Describe which processes have to be established in an IT department
  - Provide reference models for organization of IT department

CoBiT

- Structures the tasks of IT in processes and control objectives
- Defines what objectives to consider, not how !
- Objectives are categorized in domains
  - „Plan and organize“ domain - Define a strategic IT plan, manage IT investment
  - „Acquire and implement“ domain - Acquire and maintain application software, manage changes
  - „Deliver and support“ domain - Manage organization performance and capacity
  - „Monitor and evaluate“ domain - Monitor IT processes, establish governance
- Processes achieve maturity level, derived from CMM
- Assess the impact of processes and organization on system dependability
  - What are critical departments ? How can process reliability be monitored ? ...

ITIL

- Information Technology Infrastructure Library (ITIL), latest version v3
  - Started as set of recommendations by the UK Government
  - Stronger focus on technology aspects
  - Concepts and guides for IT management, IT development, Operations
- Volumes: ITIL Service Strategy, ITIL Service Design, ITIL Service Transition, ITIL Service Operation, ITIL Continual Service Improvement
- Broad tool support: IBM Tivoli Availability Process Manager, Fujitsu Interstage Business Process Manager, BMC Remedy, HP-Mercury Enterprise Solutions
- High costs for certification and training
- Methodology sometimes over-respected at the expense of pragmatism

Tools for Reliability Modeling

- Typical objectives are the evaluation of fault coverage, reliability, availability, life cycle, MTTF / MTTR or service costs for a system
- Analytical dependability evaluation
  - Analytical modelling
    - Combinatorical
    - Markov
    - Petri Nets
- Simulation (with fault injection)
- Experiments (with fault injection)

Analytical / Simulation Tools

- Availability, Reliability, Maintainability (ARM) Modelling - *http://www.certisa.com*
  - Fault tree analysis, RBDs, cause-consequence charts, failure and repair distributions, spare and repair strategies, MIL-217, Telcordia, ...
- SHARPE (Kishor Trivedi) - *http://people.ee.duke.edu/~kst/*
  - Fault trees, markov chain, RBDs, model simulation, ...
- Isograph Ltd - *http://isograph-software.com*
  - *FaultTree+* - Market leader in fault tree analysis
  - AvSim+ - Monte Carlo simulation and Weibull analysis for fault trees and RBDs
  - Failure Reporting Analysis and Corrective Action System (FRACAS), Hazop+ for hazard and operability study, combined workbench products
- OpenSesame - http://www.lrr.in.tum.de/~walterm/opensesame/
  - Extended RBDs,specification of inter-component dependencies (e.g. propagation)

Analytical / Simulation Tools

- Möbius - *http://www.mobius.illinois.edu/*
  - Multiple modeling languages
  - Distributed discrete-event simulation or numerical solving
  - Free for academic usage, still maintained
- Reliasoft - *http://www.reliasoft.de*
  - BlockSim - block diagrams, fault trees, analytical and simulation solutions
  - Products for FMEA, FRACAS, Weibull analysis, failure rate database, ...
- EDF KB3 Workbench - *http://research.edf.com/*
  - Fault trees, TBDs, markov graphs, petri nets, several calculation tools
- BQR Care - *http://www.bqr.com/*
  - FMEA, fault trees, RBD, markov, MIL 217, ...