Software Safety (draft)

Istvan Majzik

DISCOM TEMPUS - September 1999

1 Terminology

- accident: undesired and unplanned event that results in a speci…ed level of loss
  (unplanned - not as sabotage)
- incident: event that involves no loss, but with the potential of loss in other circumstances
- hazard: state of a set of conditions of the system that together with conditions of the environment will lead inevitably to an accident
  - de…ned in respect of the environment (hazard in computer systems: react to environment)
  - depends on system boundaries (‡ammable vapor can not be separated from ignition)
  - characteristics:
    - endogenous: inherent in the system
    - exogenous: external phenomena (e.g. lightning)
  - hazard level:
    - severity (damage)
    - likelihood
- risk: hazard level combined with

  - the likelihood of leading to an accident and hazard duration (the longer->higher risk)
    (relationship between hazard and accident)
  - Risk analysis: involves analysis of environmental conditions and hazard duration

- safety: freedom from accidents

  - (a relative de…nition: enabling "acceptable" loss - by whom it is judged?)
  - it can only be approached asymptotically

2 Basic concepts

General issues

- safety = building in safety, not adding it to a complete system
  (part of the initial phases - minimal negative impact)

- safety deals with systems as a whole
  (safety is not a component property)
  (interfaces, e¤ects on another component are important)

- larger view of hazards than failures
  (failure <-x-> hazard)
  (hazard <- also in the case of functioning components)

- analysis rather than past experience and standards
  (pace of change not allows to accumulate)
  (prevent before they occur!)

- qualitative rather than quantitative approach
  (early stages: no quantitative information)
  (accuracy of quantitative models is questionable; e.g. accidents are caused by failures, testing is perfect, failures are random and independent, good engineering)

- safety recognizes the importance of tradeo¤s and con‡icts in design
  (safety is a constraint)

- safety is more than system engineering½
  (also political, social, management, cognitive psychological issues)

2.1 Design for safety

- hazard elimination
- hazard reduction: minmize the occurrence (locks)
- hazard control: mitigate the e¤ects if the hazard has occurred
  e.g. passive control (do not require a positive action to prevent hazard if the control breaks, the default action is to prevent gravity switches (railway semaphore)
- damage reduction (isolation, emergency actions)

2.2 Software safety

Software safety: sw will execute without contributing to hazards

- exhibiting behavior (output, timing)
- failing to recognize and handle hazards

Safety-critical software: contribute to the occurrence of hazardous system state

Safety-critical functions: correct/incorrect/lack of operation may contribuet in hazard

Software errors: deal with them by

- correct requirements (safe, all behaviors)
- correct coding (theoretically possible)
- software fault tolerance (not enough)
- apply system safety techniques (analysis, elimination, reduction, ...)

2.3 Accident models

Energy model: uncontrolled and undesired release of energy (chemical, thermal, electrical etc.)

- to reduce: barriers, ‡ow control
- accidents:
  - energy transformation accident: energy is transformed to an other object
  - energy de…ciency action: energy is not available
- consequence: sw can not cause an accident (but together with hw)
- limited scope:
  - - limited to energy processes
  - - loss of mission is not treated

Domino model: emphasizing unsafe acts over unsafe conditions removing a domino will prevent the accident

1. ancestry or social environment
2. fault of a person
3. unsafe act or condition
4. accident

More general model:

1. management structure (organization, objectives, operations)
2. operational errors (supervisor behavior)
3. tactical error (employee behavor, work conditions)
4. accident

Chain-of-events model

- multiple factors (actions, conditions) are treated
- if the chain can be broken, the accident will not happen
- AND, OR relationships between actions -> logic tree
- actors can be involved: parallel horizontal event tracks by the actors
- external in‡uences: perturbations; actors have to adapt; unable to adapt->accident
- correction of the path can prevent accident
- role of change is important (nonroutine operation: TMI, Chernobyl)

System theory models: what went wrong within the system to allow accident

- accident: interaction which violates constraints
  lack of constraints
  - boundary areas (interfaces)
  - overlap zones (in‡uence on the same object)
  - asynchronous evolution of subsystems
- dynamic equilibrium: feedback loops and control

accident: disturbations are not handled correctly
Human task and error models

3 Design process

Managing safety: POLC: plan, organize, lead, control

- responsibility
- authority (right to command)
- accounting (measurement of results)

3.1 The system and software safety process

Integrating function: safety considerations are involved early

3.1.1 Conceptual development task: essential groundwork

- develop system safety program plan
  - identifying software-related hazards: turn to requirements
  - consistentcy of safety constraints with requirements
  - identify safety-critical parts
  - trace safety requirements, develop a tracking system
  - develop test plans
  - assembly safety-related information into documentation
- establish information and documentation …les
- establish hazard auditing and log …le (tracking system)
- review applicable documents (similar systems)
- establish certi…cation and training
- participate (safety engineer) in system concept formation
- de…ne the scope of analyses: objective, basis, hazard types, required standards
- identify hazards and safety requirements
- identify design, analysis and veri…cation requirements
- establish organizational structure (working groups etc.)

3.1.2 System design task: design phase

- update analyses (update previous analysis in new design phase)
- participate in system tradeo¤ studies (design decisions)
- ensure incorporation of safety requirements
- ensure identi…ed hazards being eliminated
- identify safety critical components
- trace system hazards into components/subsystems -> software
- review test and evaluation procedures, training
- evaluate design changes
- document safety information

3.1.3 System production and deployment tasks

- update hazard analyses
- perform system level safety evaluation
- perform safety inspections
- incorporate safety related info in documentation
- review change proposals
- perform a …nal evaluation

3.1.4 System operation tasks

- update procedures (new hazard modes)
- maintain information feedback system (logs, reports)
- conduct safety audits (periodically + triggered by needs)
- review changes and maintenance

3.2 Example of a system safety project: Zurich underground rail station

Environment: electric rail system: platform+tracks, ramp, tunnel, shopping mall, stairs, escalators, elevators, o¢ce building

Process:

- safety personnel + involving external experts (also an insurance company)
- more information in design space -> more detailed analysis
- complex analysis (maximum depth) at de…ned stages

1. De…nition of scope: safety personnel

- project documentation -> information to be used

2. Hazard identi…cation: system engineers

- project documentation -> HAZARD CATALOG

hazard	cause	level	e¤ect	category
...	...	...	...	...

3. Evaluate hazard levels: system engineers

- hazard catalog -> RISK MATRIX
- 6 levels: probability of occurrence: frequent, moderate, occasional, remote, unlikely, impossible
- 4 e¤ects: catastrophic, critical, marginal, negligible
- risk matrix:

hazard levels	hazard e¤ects		
	catastrophic	critical	...
frequent			
moderate			
...			

4. Review hazard levels: interdisciplinary group
   - risk matrix -> revised risk matrix
   - interdisciplinary knowledge is involved (transport, psychology)
5. Determine protection level: management
   - revised risk matrix -> protection level
   - protection level: line in risk matrix (priorities, cost limitations) risk reduction: above the line
   - types of risk reduction:
     - (re)design required
     - hazard must be controlled
     - hazard control desirable if cost effective
6. Revise hazards and risk matrix: experts (specialists)
   - hazards in protection level -> corrected hazard catalog and risk matrix
7. Recommend risk reduction measures: experts (specialists)
   - expert knowledge -> catalog of corrective actions (RISK REDUCTION CATALOG)

risk profile	hazard	corrective action	by/date
...	...	...	...

   - corrective actions above department authority:
     - sent to upper management level with cause, efect, action, cost
     - decision -> sent back to involved departments
     - action taken -> crossed off in the list; open items are visible
8. Quality assurance check of risk reduction measures: responsible experts
   - catalog of corrective actions -> verified catalog

9. Review of progress: management + safety personnel

    - verified catalog -> fact sheet
        - fact sheet for non-experts to document progress
        - remaining unreduced risk: further, deeper analysis

4 Hazard analysis

4.1 Basics

Central role, continuous effort

Phases of design:

- in development: identify potential hazards
- in operation: improve safety
- in licensing: demonstrate safety
  evaluate the effects of hazards that cannot be avoided

Types:

- Preliminary hazard analysis: early phase
    - identify critical system functions
- System hazard analysis: mature design
- Subsystem hazard analysis: subsystem design phase
    - studies of possible hazards
    - identifying hazards
    - determine causes, effects
    - find ways how to avoid/eliminate/control
    - planned modifications
- Operating and support hazard analysis: system use and maintenance

– human-machine interfaces

Qualitative analyses (quantitative: e¤ect of incorrect measures)

² General features:

– continual and iterative

² Steps:

– de…nition of objectives, scope, system, boundaries
– identi…cation of hazards: magnitude, risk
– collection of data (historical record, standards)
– ranking of hazards
– identi…cation of causal factors
– identi…cation of preventive measures: design criteria
– veri…cation of implementation
– quanti…cation of unresolved hazards and risks
– feedback and operational experience

Hazard level:
MIL-STD-822b

² I: catastrophic (death, system loss)

² II: critical (injury, major system damage)

² III: marginal (minor injury)

² IV: negligible

NASA:

² 1: loss of life or vehicle

² 2: loss of mission

² 3: all others

Design criteria (used to derive requirements)

- train starts with open door: must not be capable of start with open doors
- door opens while train moves: doors must remain closed
- ...

General types of analysis:

- forward (inductive) search
  - initiating event is traced forward in time/causality
  - look at the effects
  - problem: state space
- backward (deductive) searches
  - final event is traced back
  - accident investigations
- bottom-up search: subsystems are put together
  - problem: combinations of subsystems
- top-down search: higher level abstractions are refined (subsystems, components)

Problems: unrealistic assumptions

- (good engineering, testing, etc.)
- (discreapancy between documentation and system)
- (changing conditions)

4.2 Models and techniques

4.2.1 Checklist

- to check earlier experiences: they guide thinking
- dynamial update is needed
- phases:
  - hazard analysis: not to overlook known hazards
  - design: conformance to existing codes, standards
  - operational: periodic audits
- problem: grows large and difficult to use
  - false confidence about safety (incomplete checlist)

4.2.2 Hazard indices

- measure fire, explosion, chemical hazards (in processes)
- Dow Index: 1964
- plant = units, measured on the basis of tables/equiations (fireable material etc.)
- problem: mainly for process industry, or in early stages (minimum data)
- only hazard level, no causes/elimination/reduction

4.2.3 Fault tree analysis

- aerospace, avionics, electronics industry
- analyzing causes of hazards (not to identify them)
- Booelan logic methods are used
- top-down method:

- top level: foreseen, identified hazard
- intermediate level: events necessary and sufficient to cause event shown at the upper level
- pseudoevents: combination of the sets of primary events
- primary events: no further development is possible (resolution limit)

- analysis:
  - reducing pseudoevents
  - simplifying Boolean expressions
  - show combinations sufficient to hazard
  - frequency (prob.) of the hazard based on probabilities of primary events

- basic steps:

1. system definition
2. fault tree construction
3. qualitative analysis
4. quantitattive analysis

1. System definition

- determining top event (hazard) -> for all significant top events
  initial conditions
  existing events, impermissible events
- using: functional / flow diagrams, design representation

2. Fault tree construction

   - elements: top event + causal events + logical relations
   - graphical representation: symbol set, readability (underlying: Boolean algebra, truth table)
     - AND gate: causes of the event above
     - OR gate: re-expressions of the event above
     - NOT (inhibit) gate: used to express "both" property
   - automatic techniques: mainly for hardware (DF-like)

3. Qualitative analysis

   - reduce the tree to an equivalent form
   - cut sets: relationships primary and top events
   - minimal cut set: cannot be reduced further
   - tree: OR gate + minimal cut sets (including the same event is possible)
   - identify weakness: by ranking of primary events (importance: structure, occurrences in tree)

4. Quantitative analysis

   - tree: sum of the probabilities of (disjunct) minimal cut sets
   - cut set: product of prob. of primary events
   - problem: events in multiple cut sets
   - prob. density functions -> Monte-Carlo simulation

   Properties

   - fault tree for software:
     - after the implementation, with manual assistance

- only qualitative analysis

² phase in life cycle:

- after implementation, proving safeness
- early phases: problem of incomplete speci…cation

² advantages:

- helps the understanding of the system
- identifying scenarios leading to hazards
- minimal cut trees: potential weak points
  - ¤ small number of events, single-point failures
  - ¤ components in multiple cut sets: important e¤ects
  - ¤ independence of events: common cause failures
    common in‡uencing factors, to be reduced
    fault propagation (domino)

² limitations of qualitative analysis:

- constructed after the implementation (detailed design)
- cause and e¤ect relationship and little more
- simpli…ed model, without
  - ¤ time- and rate-dependent events
  - ¤ partial failure
  - ¤ dynamic behavior
- ordering and delay is not covered (fault tree is a snapshot)
  -> DELAY node is required -> loss of simplicity
- sequence of events is not handled
- multiple phases of system operation requires separate trees

² limitations of quantitative analysis:

- common-mode failures
- input data is unavailable, unrealistic

4.2.4 Event tree analysis: decision tree formalism

(fault tree proved to be hopelessly complicated, nuclear station 1970)

- forward analysis to …nd e¤ects of an event, determine all sequences
  - initial state: failure of a component
  - next states: other system components
  - ordering: chronological, from left to right
  - decision: success/failure of other components
  - path probability: product of event/state probabilities
- reduction: eliminate illogical/meaningless events
- timing issues: phased-mission analysis
- example: failure + protection system components in nuclear station
- phase in life cycle: after the design is completed
- advantages:
  - fault tree: snapshop of the system state; event scenarios combinations of component failures leading to hazard
  - event tree: sequences of events; notion of continuity, ordering
  - useful:
    - ¤ - analysing protection systems
    - ¤ - identifying top events (for FTA)
    - ¤ - displaying accident scenarios
- limitations:
  - complexity
  - separate tree for each initiating event
  - multiple events - a problem
  - ordering of events is critical

4.2.5 Cause - consequence analysis: both time dependency and causal relationship

- procedure:
  1. selection of a critical event
  2. backward search for factors that cause
  3. propagation of e¤ects of the critical event
- attached to a consequence chart
  - cause charts: alternative prior event sequences and conditions
  - fault trees: for events and conditions
- table of symbols:
  - events and conditions
  - gates between events, vertices between conditions
  - decision boxes
- automatic construction is possible
- advantages:
  - shows sequence of events (sequential control)
  - combinations of events (additional event trees)
- disadvantages:
  - separate diagrams for each critical event

4.2.6 Hazards and operability analysis: for chemical industry

- accidents are caused by deviations from the design / operating cond.
- procedure:
  - identify all possible deviations
  - identify hazards associated with the deviations (consequences)
  - identify causes of deviations
  - systematic search: defined by a flowchart
- guidewords: applied to any variables of interest (flow, temperature, time)
  - NO, NONE: result is not achieved (e.g. no flow)
  - MORE: more result than should be (e.g. more flow)
  - LESS: less result than should be (e.g. less flow)
  - AS WELL AS: additional activity, more components
  - PART OF: only some of the design intentions are achieved (e.g. mix)
  - REVERSE: OPPOSITE OF WHAT WAS INTENDED
  - OTHER THAN: something different happens
- phase in life cycle: after the design documentation is available
  - hazards are controlled by additional devices (detector, emergency valve etc.)
- advantages:
  - simplicity, easy to use
- disadvantages:
  - labour intensive, experts of the process are needed

4.2.7 Interface analyses

² structured walkthrough, to examine the propagation of faults

² types:

   – no/degraded/erratic/excessive/unprogrammed output
   – undesired side e¤ects

4.2.8 Failure modes and e¤ects analysis: developed for reliability analysis

² procedure:

   – list all components with failure modes and probabilities
   – identify the e¤ects on other components/system
   – forward search
   – system failure modes are calculated with probability

² input: failure probabilities (based on statistical data)

² output: tabular form

component	failure mode	failure prob.	% failures	e¤ects prob
...	...	...	...	...

² phase in life cycle: hardware items are identi…ed

² advantages:

   – identi…es redundancy, fail-safe design, single point of failure
   – spare part requirements

² disadvantages:

   – all failure modes have to be known
   – e¤ects of multiple failures?

4.2.9 Failure modes, e¤ects and criticality analysis

- FMEA extended with failure criticality data (rankings 1..10 etc.)
- description and preventive/corrective actions are also described

4.2.10 State machine hazard analysis

- state machine: states + transitions + conditions + actions
- safety analysis: determine if the model contains hazardous states
  - theoretical: initial state -> forward to states (computation tree)
  - practical: search backward to determine how to avoid hazardous state
- for hw and also for sw;
- safety and fault-tolerance analysis
- phase in life cycle: at any stage where a state-machine model is available
- advantages:
  - automated analysis
  - close to the view of engineers
- disadvantages:
  - logic and algebraic models and languages:
    - hard to understand and use
    - (external experts can not be involved)
    - mathematical proofs are not understood by reviewers
  - state space explosion in real systems
    hierarchical view is required (statechart)

4.2.11 Human error analysis

- Task = series of actions
- Qualitative techniques: examine for each action:
  - criticality
  - mental and physical demands
  - possible failures (forget, wrong ordering)
  - performance deviations (too slow, too fast)
  - equipment availability

Task	Danger	E¤ects	Causes	Avoidance
...	...	...	...	...

- Quantitative techniques:
  - assign probability ot human errors
  - factors that are e¤ective:
    - psychological stress
    - quality of controls and displays (human enginnering)
    - quality of training
    - quality of (written) instructions
    - coupling of human actions (dependencies)
    - personnel redundancy (inspectors)
  - probability by data collection in documented environments
  - safety analysis by event tree (path probabilities)
  - emergency: greater probabilities!
    - (best: repetitive actions, long response time)
    - (worst: emergency, short time, complex tasks)

5 Risk reduction techniques

5.1 Basics

Safety analysis data have to be used in the design process
In early phases of development

- to be efficient
- poorly designed additional modules may increase risk
- additional efforts like operators may fail or will be tricky

5.1.1 Software special:

- new hazards
  - safety dependent on sw errors
  - sw errors are difficult to tolerate, they are unpredictable
  - hw is much more simple: it may fail into a well-known state (short/open)
- new possibilities to be more powerful
  - e.g. analyzing trends

5.1.2 Design process: 2 basic approaches

- standards and experiences
  - for hw it is well-defined: how to use a valve, electrical standards etc.
  - no standard for software
    reliability, maintainability standards may even increase risk
    no generic software hazards
- guided by hazard analysis
  - identify sw-related safety requirements and constraints

– identify parts of sw which controls safety-critical operations
– elaborate behavior in erroneous states
– formal technique: data-‡ow based analysis
  ¤ -> identi…cation of critical nodes
  ¤ -> formal safety constraints
  ¤ -> design to be certi…able + veri…able
– documentation: record of safety-related decisions + assumptions -> to be taken into account in sw update

5.1.3 Risk reduction procedures: In precedence:

1. 1. Hazard elimination: Eliminating the hazardous state or the negative consequences
   ² substitution
   ² simpli…cation
   ² decoupling
   ² elimination of speci…c human errors
   ² elimination of hazardous materials or conditions
2. 2. Hazard reduction
   ² design for controlability
   ² barriers: lockout, lockin, interlock
   ² failure minimization: safety factors and margins, redundancy
3. 3. Hazard control: If a hazard occurs, reducing the likelihood leading to an accident
   ² reducing exposure
   ² isolation and containment
   ² protection systems and failt-safe design
4. 4. Damage reduction
   ² Accidents: often outside the system boundary
   ² warnings, emergency actions

5.2 Hazard elimination

5.2.1 Substitution: materials, equipments

new risks may arise, but they should be minimal

- chemical processes: flammable heat transfer to water
  hydraulic instead of pneumatic (avoid rupture and shock wave)
- missile propulsion: hybrid systems instead of gas
- gas cooled reactors (cooled also by convection if the cooling fails)
- simple mechanical locks instead of computer systems
  (e.g. automatically open the circuit if the door is open)

5.2.2 Simplification

- minimizing the number of parts, modes, interfaces
  - -> fewer opportunities to fail
  - e.g. chemical industry: fewer leakage points
  - accidents <- tight coupling, interactive complexity
  - simple interfaces -> testability
- sw: easy to use complex interfaces and systems
  - -> special care has to be taken
  - simple control structures needed
    (Honeywell autopilot: no interupts, procedures and back branches; one loop which is executed at fixed rate factors to be determined at design time
  - avoiding nondeterminism is crucial
    - time perodicity in RT systems
    - predict algorithm behavior
    - test software (avoid "transient" faults)
    - operator: rely on consistency
    - -> static scheduling (polling)

    - ¤ -> exclusive modes
    - ¤ -> state transition depends only on the current state
  - requirements:
    - ¤ testability (deterministic, no interrupts, single tasking)
    - ¤ readability (sequence of events processed)
    - ¤ interactions limited
    - ¤ worts-case timing done by code analysis
    - ¤ minimum features
  - avoiding the e¤ect of hw failures
    - ¤ state encoding: redundant
    - ¤ message encoding: only the necessary functions ("0 missiles" =/= "I am alive")

- ² reducing the unknown events caused by unproven technology:
  - space: "‡ight-proven" hw
  - new design only if requirements are not met by old ones
- ² problems:
  - adding hazard control <-> system simplicity
  - ‡exibility <-> leakage points
  - reliability (redundancy) <-> complexity increase

5.2.3 Decoupling: e¢cient but often not safe

- ² failure modes:
  - tightly coupled system: interdependent
  - failure -> rapidly a¤ect others
  - hard to isolate erroneous parts
- ² hazards: unplanned interactions -> domino e¤ect
- ² examples of decoupling:

– - …rebreaks
– - over/underpasses

² computers: increase coupling

– control multiple systems (coupling agent)

² software:

– modularization: crucial how to split up
safety critical functions into a module
– information hiding:
non-critical system does not a¤ect critical one
– safety kernel: enough to ensure safety
on a …rewall: (virtual) computer for safety-related functions

5.2.4 Elimination of speci…c human errors

² reduce the opportunities for errors

– incorrect assembly is impossible (interfaces, connectors)
– color coding

² clear status indications -> next chapter

² software: the question of programming language

– - pointers,
– - complex control structures,
– - implicit/default actions
– - overloading

5.2.5 Reduction of hazardous material or conditions

- reduction:
  - in chemical industry:
  - software: no unused code <-> COTS
- change conditions:
  - lower temperature, pressure etc.

5.3 Hazard reduction: safeguards

- passive:
  - maintain safety by their presence (shields, barriers)
  - fail into safe states (e.g. weight-operated sensors, relays which are open)
- active: require some actions to provide protection (control systems)
  - monitoring (detecting a condition)
  - measuring state variables
  - diagnosis

5.3.1 Design for controllability: make the system easier to control

- incremental control: critical actions not in a single step
  - feedback from the plant
  - corrective actions
- intermediate states: not only run/shutdown
  - multiple levels of functionality
  - "emergency mode": only critical functions
- decision aids: assist in controlling the plant

- alarm analysis: e.g. in nuclear plant
- disturbance measures: measured data -> cause-consequence analysis -> correction
- action sequencing: e.g. valve sequences

² monitoring: detecting a problem

- checking conditions of potential problem
- validating assumptions used during the design
- Detecting:
  - ¤ condition exists
  - ¤ device is ready/busy
  - ¤ input/output is satisfactory
  - ¤ limits are exceeded
- Ideal monitors:
  - ¤ detect problem fast, at low level (-> time for correction)
  - ¤ independent (limited: info + system assumptions)
  - ¤ as litte complexity as possible
  - ¤ easy to maintain, check, calibrate
  - ¤ self-checking

² monitoring computer systems:

- Levels of checking:
  - ¤ hardware level checks: memory access, control ‡ow, signals, checksums, coding
  - ¤ code level: assertions
  - ¤ audit level: data consistency, independent monitoring
  - ¤ system level: supervisory checks
- Checks are better at lower levels:
  - ¤ less delay -> no erroneous side-e¤ects
  - ¤ ability to isolate/diagnose
  - ¤ ability to …x (rather than backward recovery)
- Structure: without additional risk
  - ¤ safety kernel

5.3.2 Barriers

- Types:
  - lockout: make access to a dangerous process/state dsi¢cult
  - lockin: make di¢cult to leave a safe state
  - interlock: enforce a sequence of events/actions
- Lockout: prevents a dangerous event / entering dangerous state
  - physical barriers:
    - avoid elecromagnetic interference
    - (aircraft radio system, electromagnetic particles)
  - authority limiting
    - prevent dangerous actions (e.g. correcting user inputs in autopilots)
    - -> do not prohibit necessary actions!
  - sw: access to safety-critical code/variables
    - security techniques
access rights (for users)
access control list (for resources)
capabilities (ticket to enter)
    - reference monitor: controlling all access
    - multiple con…rmations
    - restricted communication
    - security kernel (low-level)
- Lockin: maintain a condition
  - keep humans in an enclosure (seat belts, doors)
  - contain harmful/potentially harmful products
  - maintain controlled environment (space suits)
  - constrain a particular event (safety valves)
  - SW: tolerate erroneous inputs

- Interlock: enforcing correct sequence of events
  - inhibit: event does not occur inadvertently (sequence check)
  - inhibit: event does not occur if condition C (deadman switch)
  - sequencer: event A occur before event B (traffic signals)
  - interlock fails -> function should safely stop
  - danger: maintenance removal of interlocks
  - SW: often the hw interlocks have to be kept;
    - sw only monitors interlocks;
    - keeps safe sequences
  - SW mechanisms:
    - prg. language synchronization features: error prone (hw, sw)
    - baton: passed to a function; checked before execution: pre-requisite tasks have to modify it
    - come-from check: process receives data from valid source
- Example: Nuclear detonation system
  - isolation: separating critical elements
  - incompatibility: unique signals
    - signal pattern to start
    - different channels (energy, information)
  - inoperability: keeping in inoperable state (without ignition)

5.3.3 Failure minimization:

- reducing failure rate -> reducing risk
  - safety margins
  - redundancy
  - error recovery
- Safety margins:

  – in a design many uncertainties: failure rates, conditions
  – safety factors: designing a component to withstand higher stress nominal (expected) strength / nominal stress > 1
  – problem: probability density functions (may overlap) probability(stress) functions
    -> safety margin has to be de…ned

² Redundancy:

  – many forms: replica, design diversity
  – often con‡ict between safety and reliability
    ¤ e.g. redundancy: more power consumption
    ¤ increased complexity -> new faults (redundancy management)
    ¤ e¤ective againts random failures
  – well-designed redundancy is required
    ¤ no common mode failures
    ¤ reduced dependencies (also during test and maintenance)
    ¤ speci…cation has to be elaborated more precisely
  – reasonableness checks: di¢cult to write

² Recovery:

  – forward and backward recovery have to be used together (time + environment state)
  – avoiding domino e¤ect: complex algorithms which are error prone
  – forward recovery is proposed, if the error can be identi…ed and …xed

5.4 Hazard control

² Limiting exposure

  – normal (default) state is safe

  - starting in a safe state
  - error -> automatical shutdown to safe state
  - trigger is required to go to unsafe state

- Isolation:
  - barriers and shields
  - plants located in isolated area (no population)
  - transport of dangerous material

- Protection systems:
  - detectors (gas, …re, water etc.) -> moving to safe state
  - panic button (training is required)
  - watchdog timers: separate power etc.
  - passive devices are safer
  - protection system: should signal that it works
    it can also cause damage (emergency destruct)
  - fallback states:
    - partial shutdown
    - hold (no new function, maintain safe state)
    - emergency shutdown
      normal: cut power form all circuits
      production: after the current task is completed
      protection: keep only necessary functions
    - restart
  - subsystems:
    - sensor to detect hazardous condition
    - challenge subsystem to test the sensor
    - monitor to watch the interruption of the challange-response sequence

5.5 Damage reduction:

- emergency procedures: prepared, trained, practiced
- point of no return: turn to emergency actions instead of continue to save the system
- warning: too frequent -> insensitive people
- techniques: escape route + limiting damage

6 Software safety analysis

6.1 Basics

- - accidents in which sw involved: due to requirement ‡aws
  - incompleteness
  - wrong assumptions
  - unhandled conditions
  - (coding errors a¤ect reliability, not safety; + unintended functions)
- -> general criteria required: checklist for requirement completeness and safety
  - top-down analysis is possible
  - bottom-up analysis is not practical (too much states)
- components in requirements:
  - 1. Basic function or objective, safety criteria included
  - 2. Constraints on operating conditions
limit the set of possible designs
e.g. physical constraints, performance, process characteristics
  - 3. Prioritized quality goals (to help design decisions)
- completeness: the most important property of speci…cations

  - distinguish from any undesired behavior
  - "lack of ambiguity"
  - ambiguous: subject to more than one implementation

- software model:
  - controller + sensors + actuators + plant
  - state machine model (describing behavior, black box)
  - model of the plant inthe sw:
    - must be synchron wih real plant
    - must completely describe the real plant
    - complete trigger specification is required

6.2 Human-computer interface criteria

- alert queue:
  - events, ordering (time or priority), notification mechanism,
  - review and disposal, deletion
- transactions: multiple events/actions in one
- displaying data:
  - cause events identified
  - refreshing: time, new events, operator required
  - disappearing

6.3 State completeness

- the system and sw must start in a safe state
  - interlocks initialized
- internal model of the plant must be updated after startup

  - (plant changes when the sw not running)
  - (manual actions have to be taken into account)

- system and local variables (incl. clocks) must be initialized upon startup
  - (complete startup or after o¤-line phase)
  - (detecting loss of information: message numbers, timestamps)

- to be speci…ed: handling inputs before startup / after shutdown
  - (some hw can retain inputs)

- the maximum time the computer waits for the …rst input is speci…ed
  - no input -> alarm for operator;
  - the internal model of the planrt cannot be synchronized

- paths from fail-safe states must be speci…ed, the time
  - spent in reduced-function state must be minimized
  - (non-normal processing modes are limited)

- there must be a response for inputs in any state including
  - indeterminate states
  - (also for "unexpected" inputs)
  - (e.g. aborting twice, opening sth twice etc.)
  - (unexpected input indicates a malfunction)

6.4 Input or output variable completeness

- (regarding sensors and actuators)
- all information from the sensors must be used in the speci…cation
  - unused input -> omission in speci…cation; what to do with it?
- legal output values which are never produced should be checked
  - (e.g. spec. only opens a valve, without closing it)

6.5 Trigger event completeness

- robust system: correct answer to unexpected inputs
- unexpected inputs/behavior checked by environment constraints
- logging unexpected inputs is important
- events that trigger state chenges must satisfy:
  - every state has a transition for every possible input
  - all conditions (input patterns) have to be taken into account
  - every state has a de…ned time-out if no input occurs
- behavior of the state machine must be deterministic
  - (one transition for each input pattern; disjoint conditions)
  - (predictable behavior is required)
- all incoming values sholud be checked;
  - response speci…ed for out-of-range values
  - (indicator of malfunctions / out of synchrony)
- all inputs must be bounded in time;
  - behavor speci…ed if the limits are violated / unexpected inputs arrive
  - ("exactly at" is not a good speci…cation style)
- a trigger involving the unexistence of an input must be bounded in time
  - (given by clocks or using other events)
- minimum and maximum load assumptions must be speci…ed for interrupts
  - whose arrival rate is not limited

- minimum-arrival rate checks should be included
  - (the sw must query the empty communication channels)
- response to overload conditions must be specified
  - alarm
  - trying to reduce load (controlling the plant)
  - lock out interrupts (masking)
  - reduced accuracy output generation
  - reduced functionality (process selected interrupts only)
- performance degradation sholuld be graceful, operators must be informed
  - (predictably and not abrupt degradation)
- if reconfiguration is used, hysteresis delay must be included
  - (to avoid ping-pong)

6.6 Output specification completeness

Safety-critical outputs are checked for reasonableness.

6.6.1 Capacity:

- the absorption rate of the output environment must be higher than the input/computing rate
  - (to avoid output saturation)
- action should be specified if the output rate is exceeded
- human operators should not be overloaded
  - (actions and responses should not be mixed)
- automatic update and deletion of human interface must be specified

- (events negated or updated by other events, becoming irrelevant)

- specify what to do when the event is displayed and when removed

  - (e.g. removing events only after operator commit)

6.6.2 Data age:

- all inputs used by output events must be limited in the time they can be used

  - (data age; validity time of messages)

- incomplete transaction should be cancelled after a time-out

  - (operator should be informed)
  - (incomplete transaction: higher risk case)

- revocation (undo) of actions require:

  - speci…cation of conditions and times when it could be done
  - operator warnings

6.6.3 Latency:

- latency factor is speci…ed if the output is triggered by an interval of time without a speci…ed input

- action to be speci…ed: what to do if an input arrives late, while the "late output" is generated

- latency factor: data display for operator changes just prior to a new command from the operator

  - (ask the operator: the change was noted or not)
  - (the operator has opportunity to observe the change)

- hysteresis must be speci…ed for human interface data,

  - (to allow time for interpretation)
  - speci…ed: what to do if data changes in hysteresis period

6.7 Output to trigger event relationships

- basic feedback loops has to be involved with checks on the inputs
  - (to detect the effect of any output of the sw)
  - (not only limits, but also trends are important)
  - (expected behavior of the plant is checked)
- for every output detected by an input there must be specification
- for normal response
- for abnormal (missing, late, early etc.) response
- too early inputs must be detected and responded as abnormal
  - (considering output latency)
- stability requirements must be specified when the plant
  - seems to be unstable

6.8 Specification of transitions between states

- all specified states must be reachable
  - (otherwise no function or missing state transition)
- states should not inhibit the production of later required outputs
  - (otherwise reachability problems may inhibit the output)
- output commands should be reversible
  - (cancel or reverse some actuator commands)
- states reversing the commands should be reachable
  - (reachability analysis)

- preemption requirements should be specified
  - normal processing in parallel
  - refusing the new action
  - preemption of the partially completed transaction
- soft and hard failure modes should be eliminated from all hazardous outputs
  - soft failure mode: an input is required to go from a
    - given state with A to all others with B;
    - missing of this input is a ~
  - hard failure mode: an input is required to go from all
    - states with A to all others with B;
    - missing of this input is a ~
- multiple paths should be provided for state changes that maintain or enhance safety
  - (a single failure should not prevent taking actions)
- mutiple inputs should be required for paths from safe to
  - hazardous state

6.9 Constraint analysis

- transitions must satisfy software safety requirements
  - failing to perform a required function
  - unintended function, wrong answer
  - function at the wrong time, wrong order
  - failing to recognize a hazardous condition (no correction)
  - producing wrong response to hazardous condition
- reachable hazardous states should be eliminated,

– or at least reduced in time and frequency

- general safety policy:
  - no paths to catastrophic states
  - always path(s) from hazardous to safe state
  - paths from hazardous state to minimum risk state

6.10 Checking the speci…cation:

- automated reachability analysis
- constrained speci…cation language
  - (e.g. time bounds of inputs have to be speci…ed)