

What does System Safety mean for Software?

Some Aspects on Software in Safety-critical Applications.

Inga-Lill Bratteby-Ribbing

inga-lill.bratteby-ribbing@fmv.se

Defence Materiel Administration (FMV)

Sweden



”Säkerhet” – a generic concept: Security + Safety

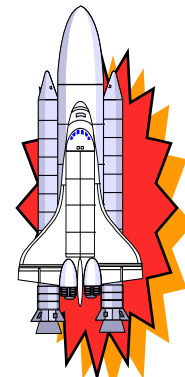
- **Security** – *IT-säkerhet*

The ability of a system to withstand intrusion, uncontrolled damage and/or influence.



- **Safety** – *Systemssäkerhet*

The property of a system to not cause accidents or losses to people, property or environment when used under prescribed conditions.



*Focus: Safety of software-based systems
(Software Safety)*



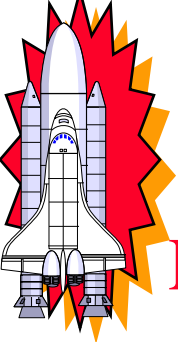


Safety of a software product

The safety properties of a software product can only be evaluated

... with software as an integrated part of the system in its intended environment and usage

Software regarded as safe in a specific system-environment-usage, need not be safe in a different context.



Ariane 5: Flight 501 Accident scenario

Explosion

- ← Destruction triggered by the safety protection system
- ← Improper flight data
- ← Error diagnostics from the flight control systems
- ← Execution of active + identical hot stand-by system stopped
- ← Overflow in alignment function after 37 s in flight mode
- ← No exception handler for trajectory value overflow
- ← Alignment function allowed to execute 50 s after count down

But

- **Stronger engine** for A5
- + Alignment function based on trajectory data for A4
- ⇒ Higher horizontal velocity in the alignment function
- ⇒ Overflow



Software deficiencies

(*"Fel" i programvara*)

(fault → error → failure)
(hazard → unsafe state → accident)

Product, e.g. systematic deficiencies in the

- Specifications
- Design

Processes, e.g. inadequate

- Production processes
- Production tools

People, e.g. erroneous usage by those involved in the

- Development,
- Maintenance and
- Operation of software-based systems

Software deficiencies

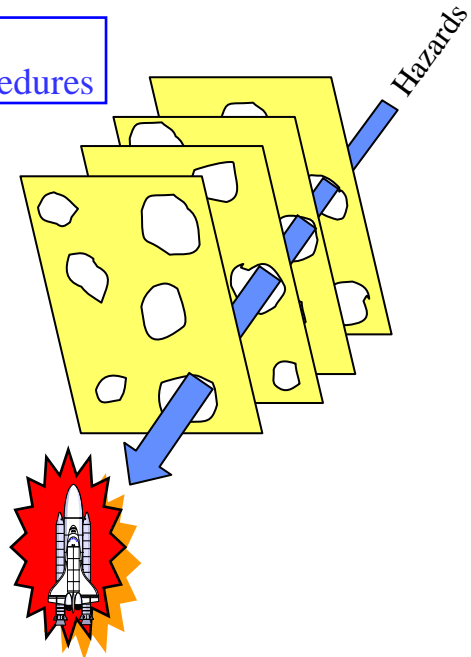
("Fel" i programvara)

(fault → error → failure)
(hazard → unsafe state → accident)

Systematic deficiencies in

- **Specification** of system, operational conditions, interfaces
- **Design**
 - Redundancy ineffective for SW
 - Function allowed in wrong mode
 - Exception handler excluded
- **Production process**
 - No safety analysis
 - Insufficient verification procedures
- **Production tools**

HW-SW i/f



Erroneous usage

- **Reuse** under changed conditions
- Usage domain \neq the specified domain
- Operational procedures \neq instructions and specifications

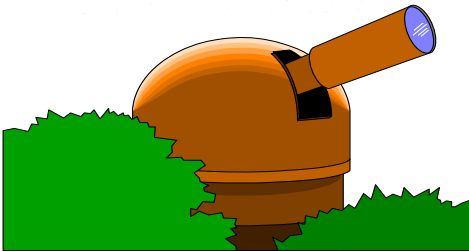
Stronger engine
Resumed countdown simplified

Patriot

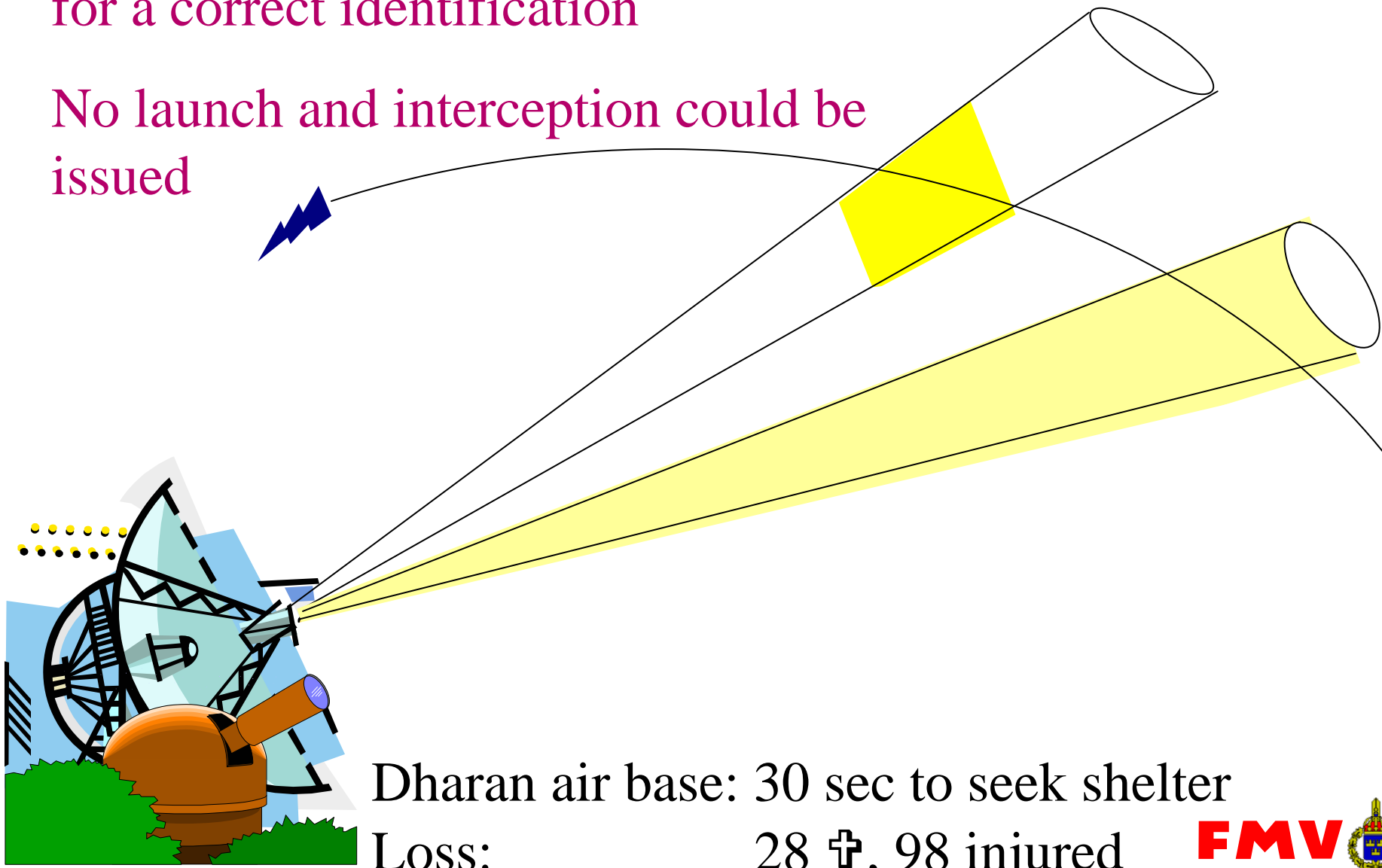
-for defence against attacking aircraft/missiles

Usage in Dharan:

- Protection against missiles with a velocity several times higher than intended
- Continuous operation (100h without restart) far beyond the specified period of time (14h).



- The precision of code was insufficient for a correct identification
- No launch and interception could be issued



Dharan air base: 30 sec to seek shelter

Loss: 28 †, 98 injured



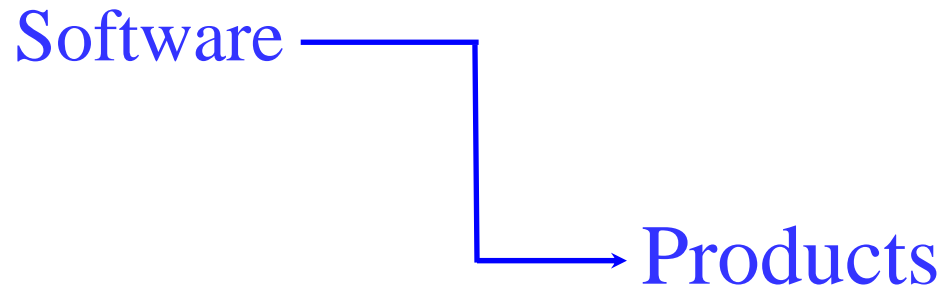
Safety problems at software reuse

A software component with a sufficiently safe behaviour in its intended environment and usage may at reuse **in a different situation** or **in a modified system** fail and lead to an accident.

For **COTS products** with limited access to internal details the possibilities to analyse the effect of changes and the resulting behaviour will be even more difficult.

How to achieve Safety in Sw-based systems

Software Safety issues with respect to the

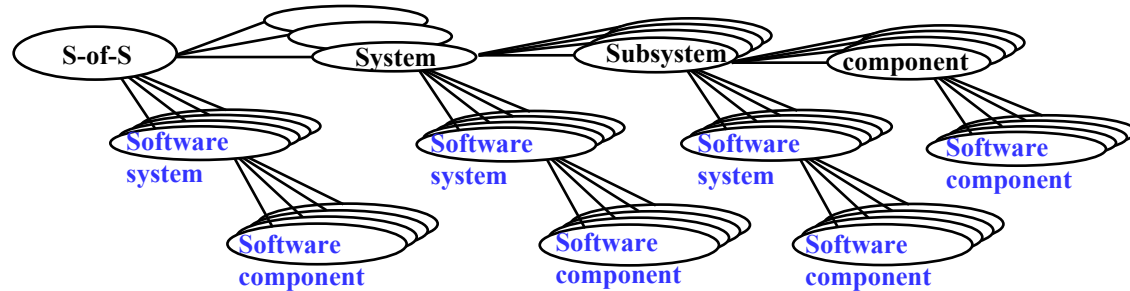


A well-defined Architecture

A safety-oriented view as part of the overall system architecture

System-of-systems

Hierarchy of architectures



Every architecture

with multiple views: Logical, Process, Development, Physical, Scenarios

for description of: Underlying structure (*parts, relationship*)

Uniform behaviour for incl. parts (*principles for interaction, diagnostics, time*)

...also from a

Safety viewpoint:

Critical parts/functionalities identified

Strategies for how to handle safety threats defined

Safety features added



Independence to parts of lower criticality

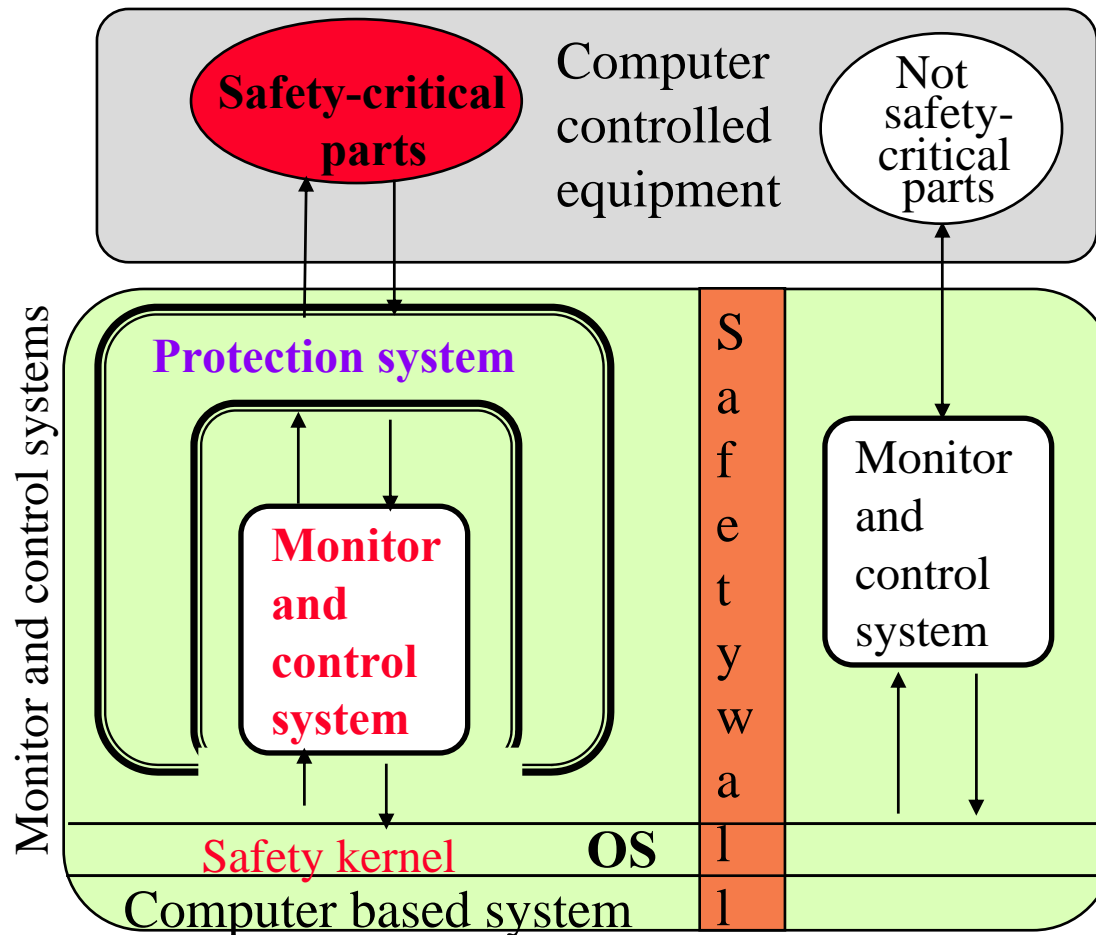
Critical parts isolated from parts of lower/none criticality

Several partitioning techniques:

- **Physical**, e.g.
 - separate processors
- **Logical**, e.g.
 - encapsulation, different address spaces, separate tasks
- **Temporal**, e.g.
 - interrupt free execution

Safety-oriented Architecture

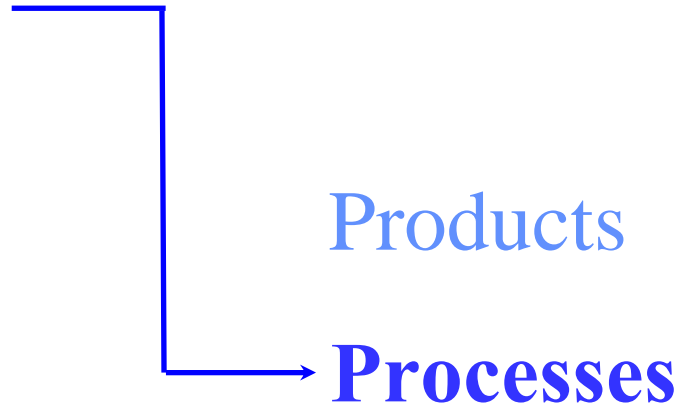
... an example on support for safety partitioning



How to achieve Safety in Sw-based systems

Software Safety issues with respect to the

Software



Products

Processes

The Software safety processes

are based on...

- **Software engineering** — *the framework for software development*
 - Personnel qualifications, Processes , Product
- **Reliability strategies** — *bottom-up technique*
 - **Fulfil required functionality**
 - Identify faults and possible failures *-Behaviour as specified*
 - Eliminate these *-Zero defect programming*
 - Provide correct functionality despite remaining faults *-V&V, Defensive programming*
-Fault tolerance
- **Safety methodology** — *top-down technique*
 - **Prevent undesirable behaviour:** *-Safety threats to people/property/environment*
 - Identify hazards an possible accidents *-Safety analysis (HAZOP, FTA, ...)*
 - Identify safety-critical parts and additional safety requirements
 - Select solutions reducing risk and criticality *-Diversity , Safety protection*
 - Select production processes on a par with criticality *-e.g. test coverage w.r.t source/ object code*

Verifications w.r.t reliability

Static

Dynamic

Require-
ments

Design

Formal
represent-
ation

Semi-
formal
represent-
ation

Source-
code

Object-
code

Exe-
cutable
model

Reviews

Formal
proof

Model
checking

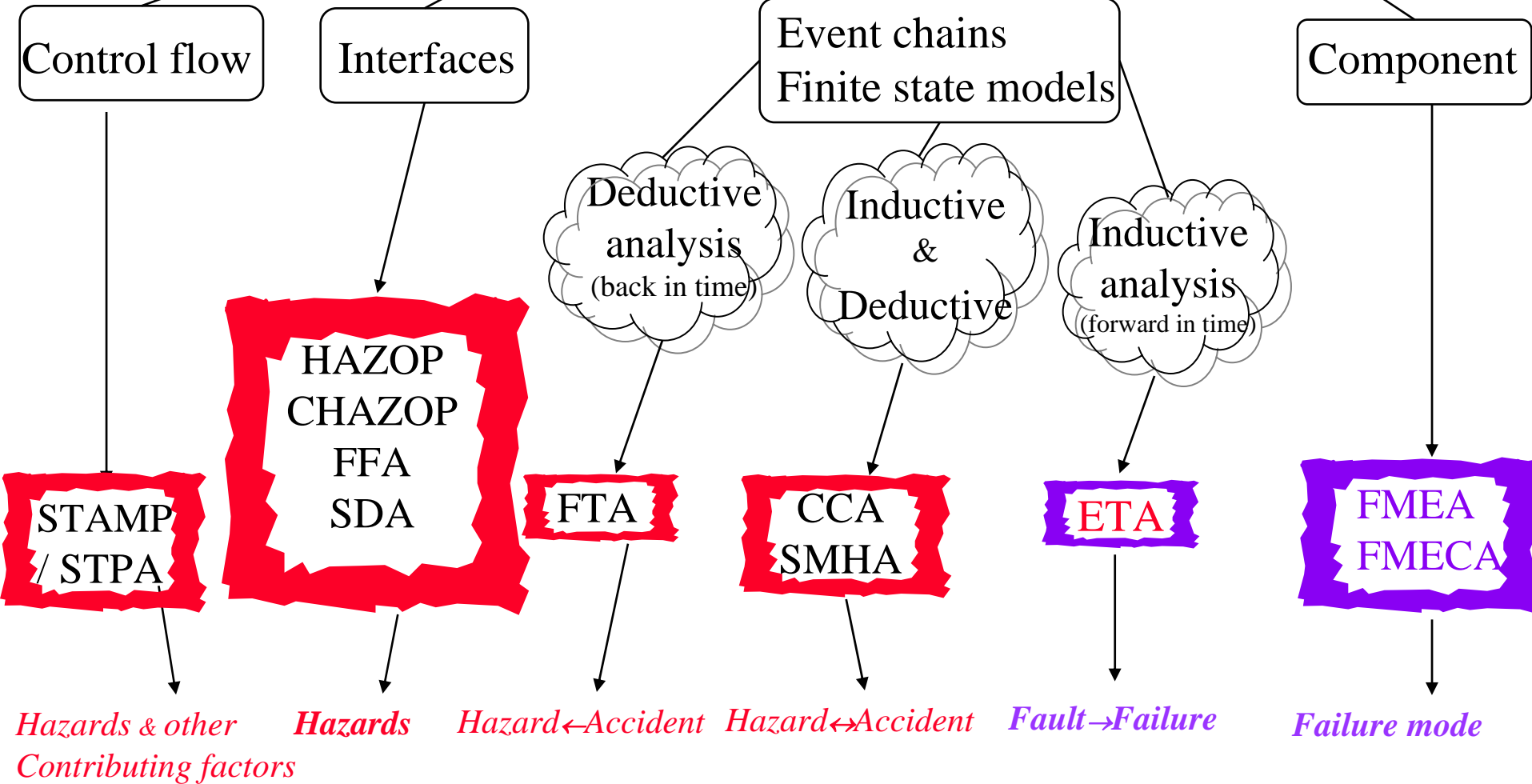
Behaviour
analysis

Static
analysis

Object
code
analysis

Testing

Safety analyses



Analysis results ⇒ *Additional system-specific safety requirements*
⇒ *Design changes and restrictions*

How to achieve Safety in Sw-based systems

Software Safety issues with respect to the ...

....

Products

Processes

→ **People** involved:

Client / End-user

Supplier (safety engineer, developer)

Acquirer



Defence: SW Safety Programme

Guidelines:

- H ProgSäk 2001 (Swedish edition)
- H ProgSäkE 2005 (English edition)



Handbook for Software
in Safety-Critical
Applications

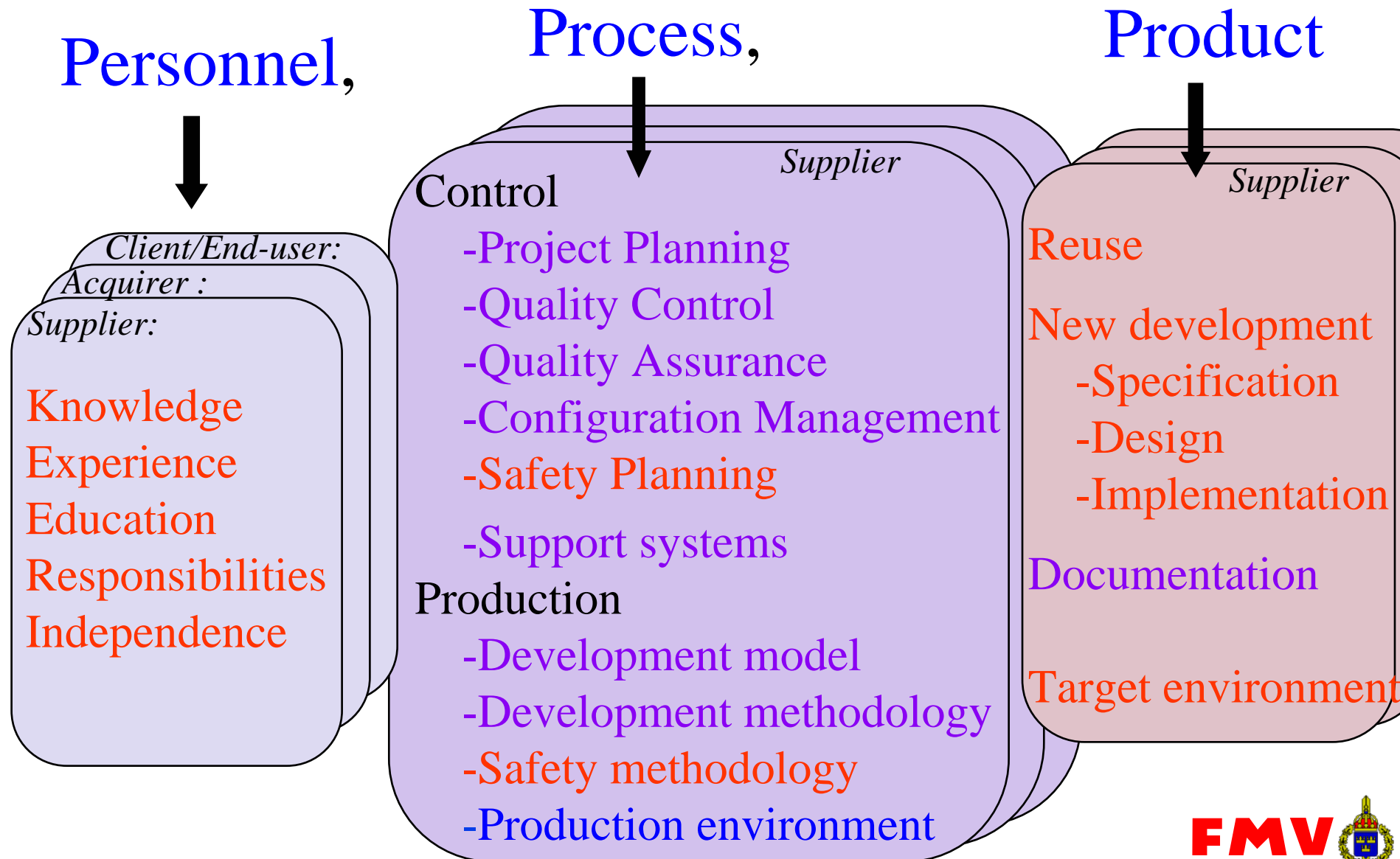
H ProgSäk E



2005



H ProgSäk E: Requirements w.r.t. role



Defence: SW Safety Programme

Guidelines:

- H ProgSäk 2001 (Swedish edition)
- H ProgSäk E 2005 (English edition)

Templates:

- Evaluation of the SW Safety maturity
- Comparison with other Sw Safety standard

Knowledge transfer:

- Seminars , workshops, conferences (next course: Jan 28)
- Network (e.g. SESAM: Study Group in Software Safety)

Web-info:

- www.fmv.se under Publikationer:Handböcker:H ProgSäk 2001
- [//sesam.smart-lab.se](http://sesam.smart-lab.se) under Arbetsgrupper: Programvarusäkerhet

Questions ?

