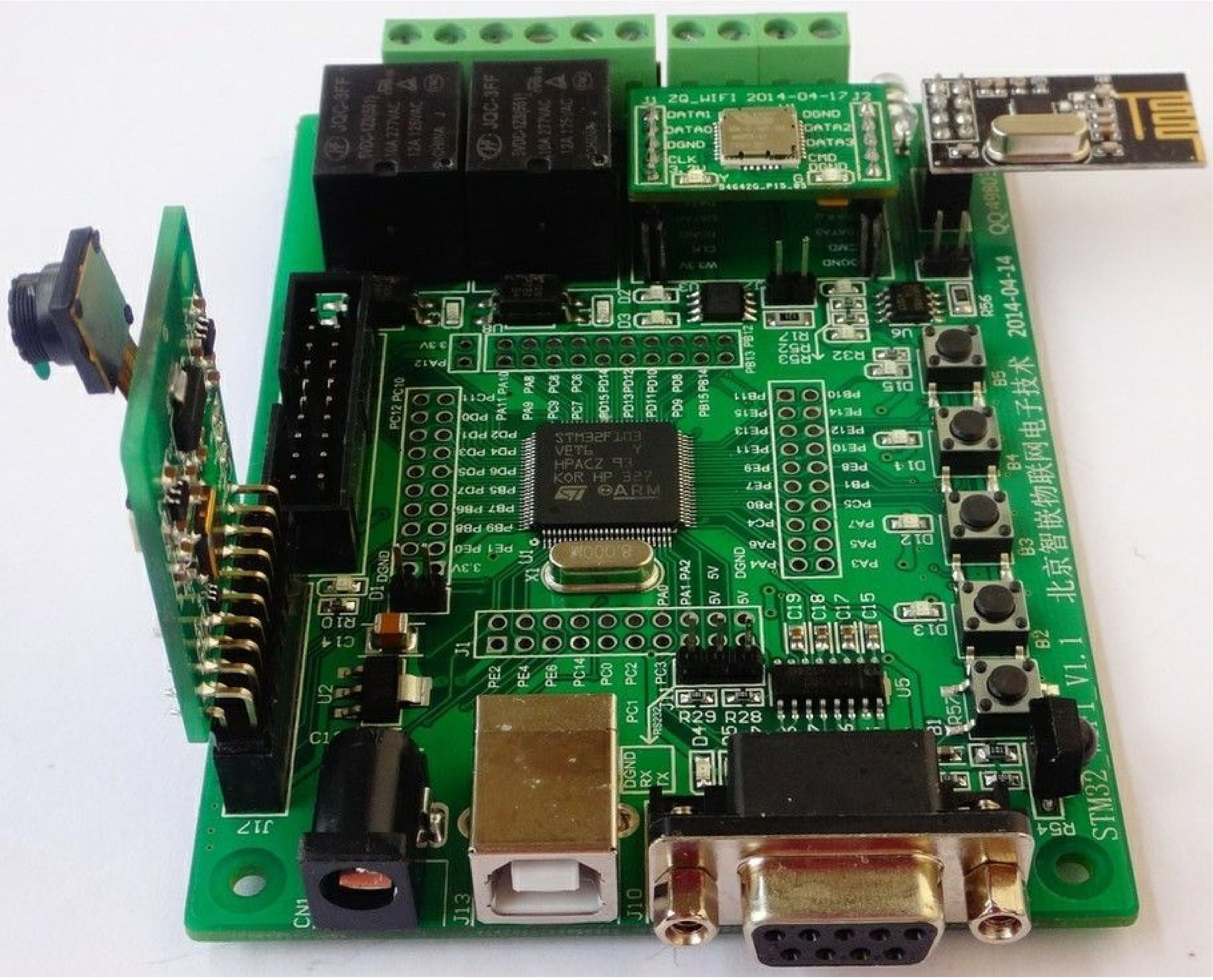
WHEN HUMAN LIFE DEPENDS ON SOFTWARE

INTRODUCTION TO SAFETY-CRITICAL SYSTEMS

MACIEJ GAJDZICA

Solw'IT Let's Solve It



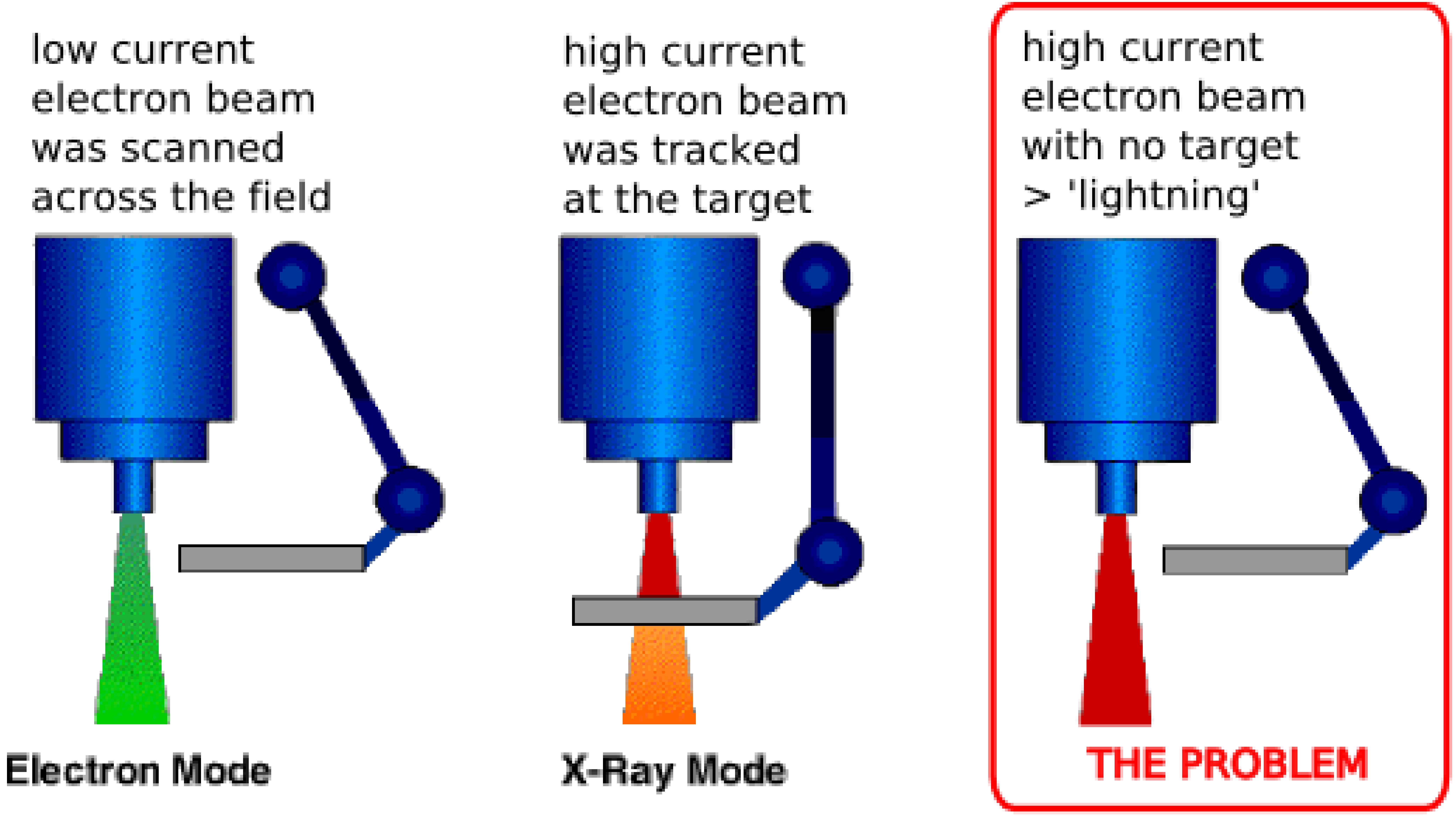
@MaciekGajdzica

ucgosu.pl


HELLO, IT
HAVE YOU TRIED TURNING IT OFF
AND ON AGAIN?

„Will we continue on our undisciplined course, blown by the chaotic winds of business and government, until one of us finally blunders badly enough to wake the sleeping giant of government regulation?”

ROBERT C. MARTIN



tray including the target, a flattening filter, the collimator jaws and an ion chamber was moved OUT for "electron" mode, and IN for "photon" mode.

THERAC-25

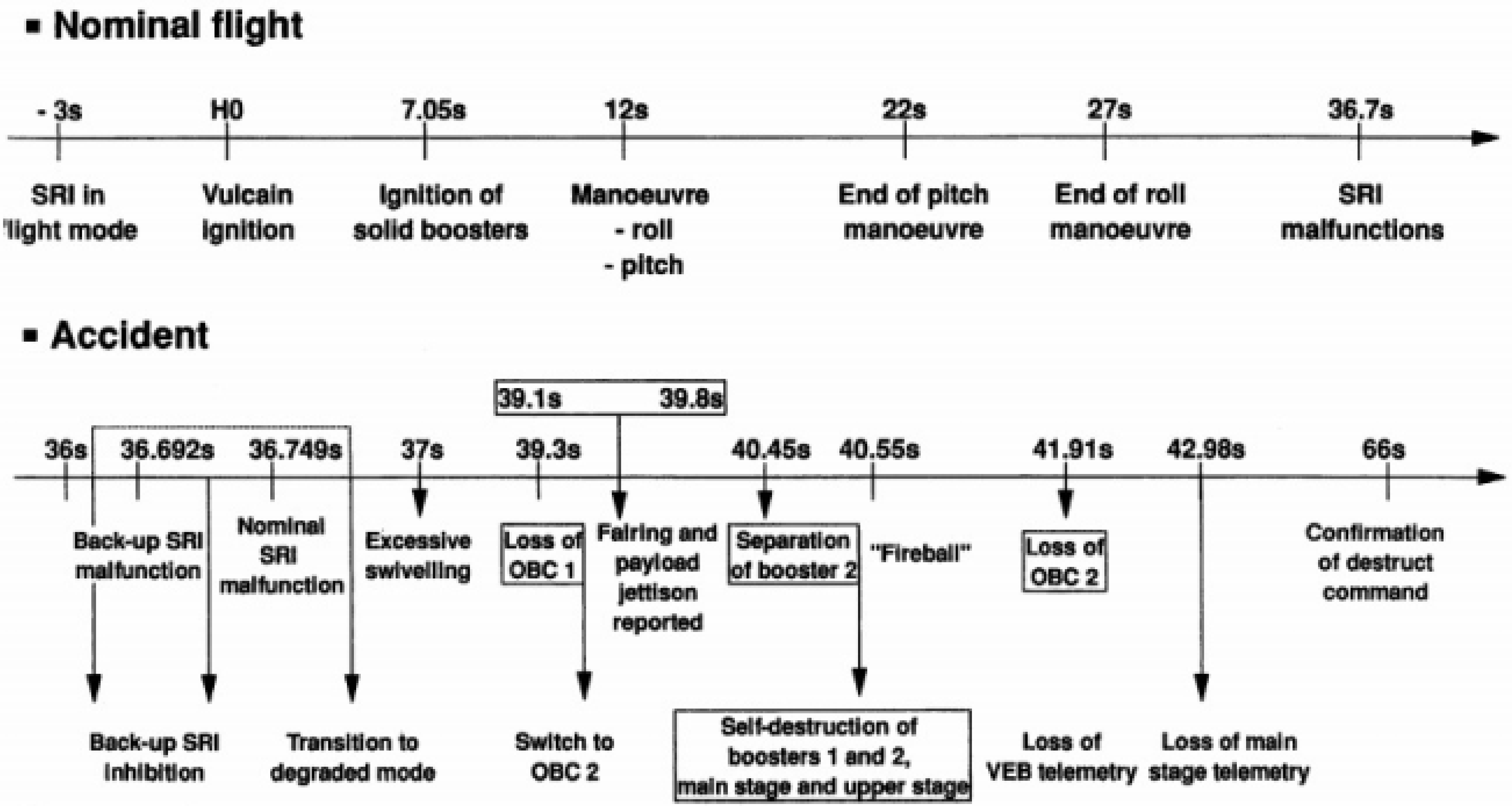
```
if mode/energy specified then
    begin
        calculate table index
        repeat
            fetch parameter
            output parameter
            point to next parameter
        until all parameters set
        call Magnet
        if mode/energy changed then return
    end
if data entry is complete then set Tphase to 3
if data entry is not complete then
    if reset command entered then set Tphase to 0
return
```

```
Magnet:
        Set bending magnet flag
        repeat
            Set next magnet
            call Ptime
            if mode/energy has changed, then exit
        until all magnets are set
        return

Ptime:
        repeat
            if bending magnet flag is set then
                if editing taking place then
                    if mode/energy has changed then exit
        until hysteresis delay has expired
        Clear bending magnet flag
        return
```

THERAC-25

- Whole program implemented by a single person
- No documentation
- Faulty risk assessment
- Insufficient tests
- Ignoring reported issues



Directorate of Launchers

Why visibility matters—the Ariane 5 crash

- Velocity was represented as a 64-bit float
- A conversion into a 16-bit signed integer caused an overflow
- The current velocity of Ariane 5 was too high to be represented as a 16-bit integer
- Error handling was suppressed for performance reasons

```
-- Vertical velocity bias as measured by sensor
L_M_BV_32 :=
    TBD.T_ENTIER_32S ((1.0/C_M_LSB_BV) *
    G_M_INFO_DERIVE(T_ALG.E_BV));
-- Check, if measured vertical velocity bias ban be
-- converted to a 16 bit int. If so, then convert
if L_M_BV_32 > 32767 then
    P_M_DERIVE(T_ALG.E_BV) := 16#7FFF#;
elsif L_M_BV_32 < -32768 then
    P_M_DERIVE(T_ALG.E_BV) := 16#8000#;
else
    P_M_DERIVE(T_ALG.E_BV) :=
              UC_16S_EN_16NS(TDB.T_ENTIER_16S(L_M_BV_32));
end if;
-- Horizontal velocity bias as measured by sensor
-- is converted to a 16 bit  int without checking
P_M_DERIVE(T_ALG.E_BH) :=
    UC_16S_EN_16NS (TDB.T_ENTIER_16S ((1.0/C_M_LSB_BH) *
    G_M_INFO_DERIVE(T_ALG.E_BH)));
```

*Source: http://moscova.inria.fr/~levy/talks/10enslongo/enslongo.pdf

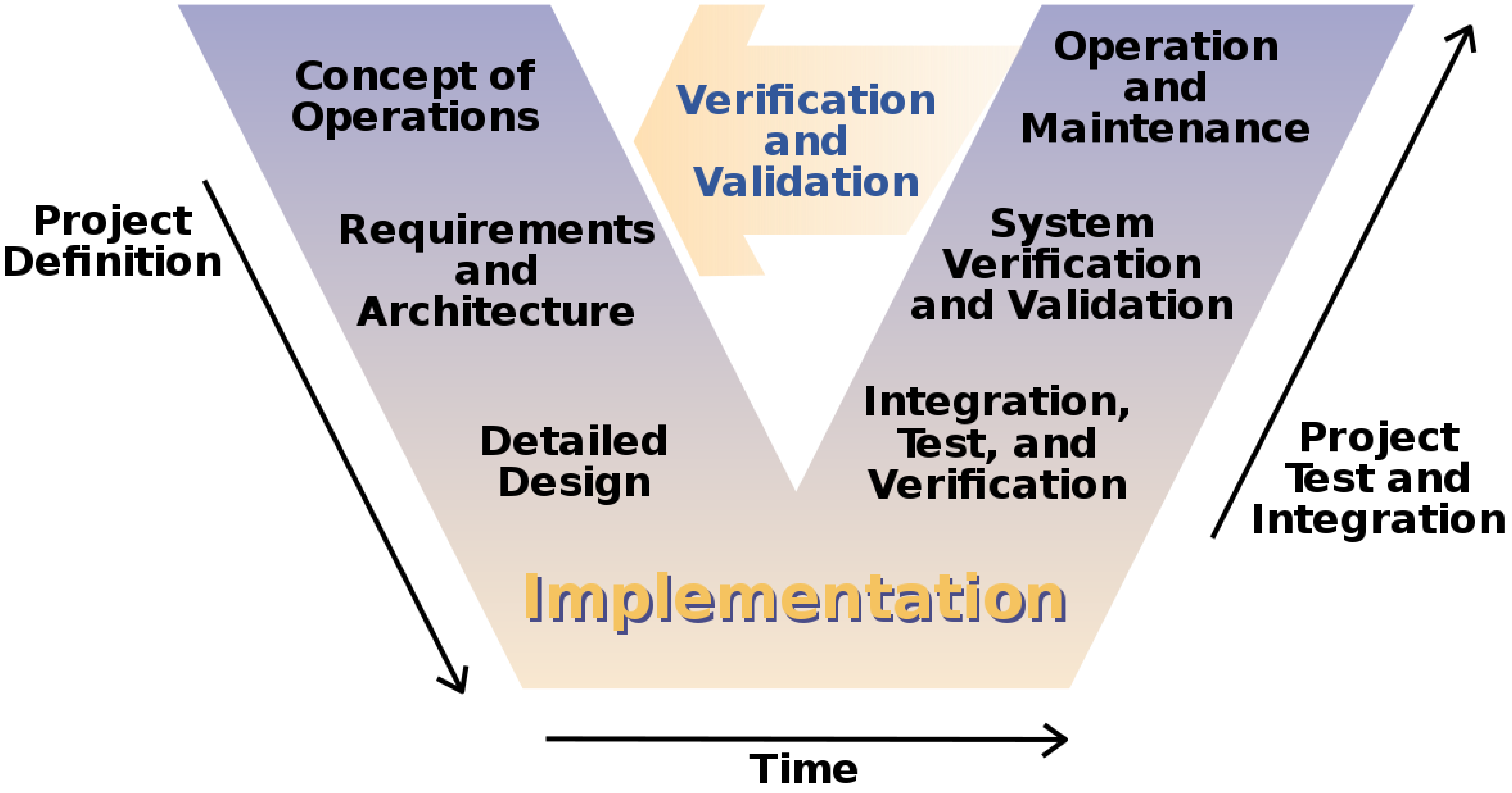
ARIANE 5

- Overflow of int16 variable
- Variable not needed - dead code
- Copy pasted from Ariane 4
- Insufficient simulation tests

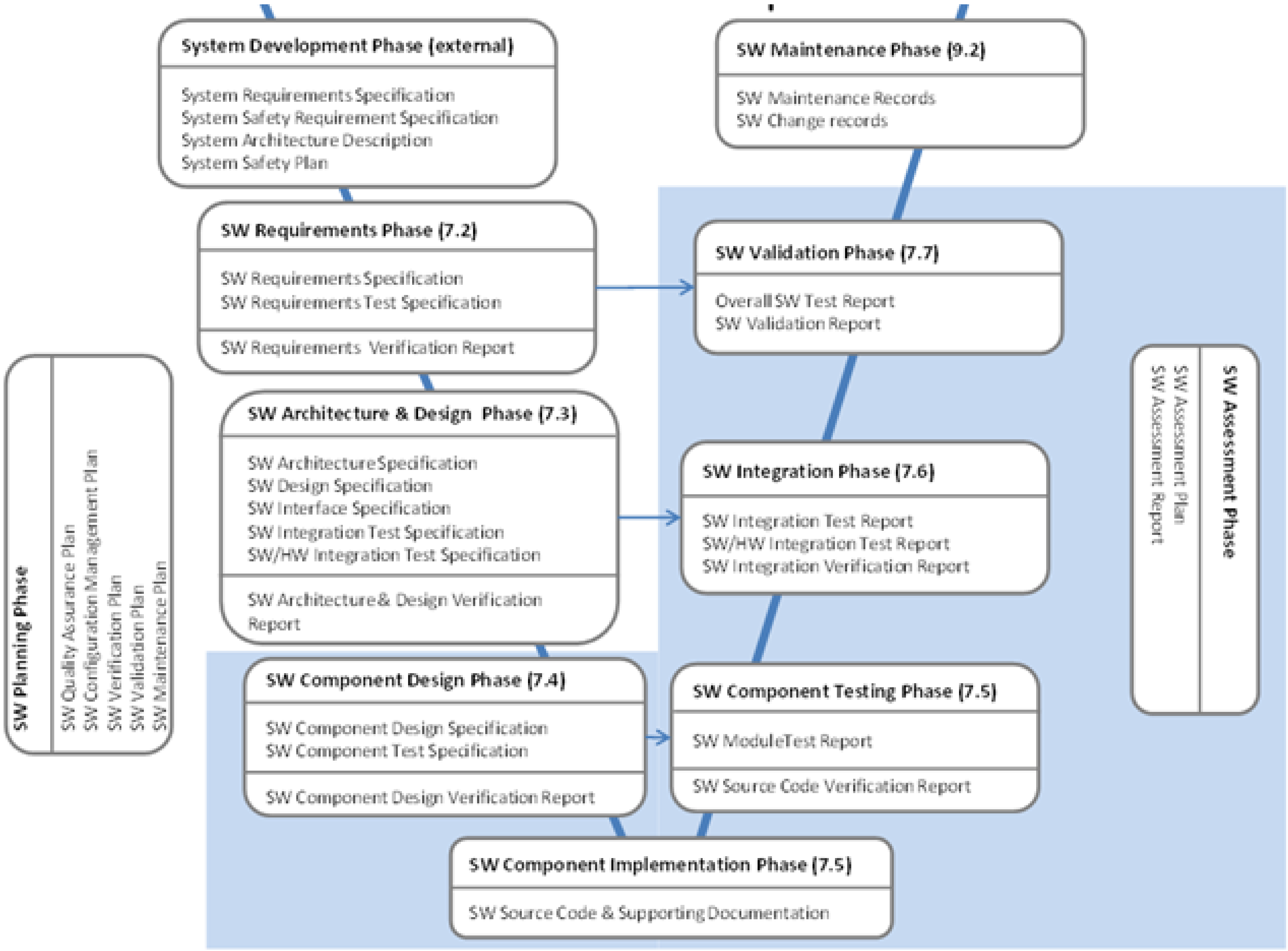
SOFTWARE STANDARDS

- IEC 61508: General standards for industry
- IEC 62304: Medical systems
- ISO 26262: Automotive
- IEC 61513: Nuclear Power Plants
- EN 50128: Railway Transportation
- DOC-178C: Aerospace
- NASA Safety Critical Guidelines

V-MODEL



Concept of Operations
Requirements and Architecture
Detailed Design
Verification and Validation
Operation and Maintenance
System Verification and Validation
Integration, Test, and Verification
Implementation
Project Definition
Project Test and Integration
Time



System Development Phase (external)
System Requirements Specification
System Safety Requirement Specification
System Architecture Description
System Safety Plan
SW Maintenance Phase (9.2)
SW Maintenance Records
SW Change records
SW Requirements Phase (7.2)
SW Requirements Specification
SW Requirements Test Specification
SW Requirements Verification Report
SW Validation Phase (7.7)
Overall SW Test Report
SW Validation Report
SW Planning Phase
SW Quality Assurance Plan
SW Configuration Management Plan
SW Verification Plan
SW Validation Plan
SW Maintenance Plan
SW Architecture & Design Phase (7.3)
SW Architecture Specification
SW Design Specification
SW Interface Specification
SW Integration Test Specification
SW/HW Integration Test Specification
SW Architecture & Design Verification Report
SW Integration Phase (7.6)
SW Integration Test Report
SW/HW Integration Test Report
SW Integration Verification Report
SW Assessment Phase
SW Assessment Plan
SW Assessment Report
SW Component Design Phase (7.4)
SW Component Design Specification
SW Component Test Specification
SW Component Design Verification Report
SW Component Testing Phase (7.5)
SW Module Test Report
SW Source Code Verification Report
SW Component Implementation Phase (7.5)
SW Source Code & Supporting Documentation

„It does not require that any particular lifecycle model is used, but it does require that the plan include certain ACTIVITIES and have certain ATTRIBUTES.”

IEC 62304 (medical standard)

SAFETY INTEGRITY LEVEL

- SIL4: Life of many people in danger
- SIL3: Life of one person in danger
- SIL2: Severe injury possible
- SIL1: Minor injury possible

Safety Integrity Level	Probability of Dangerous Failure per hour
SIL 4	>= 10^{-9} to 10^{-8}
SIL 3	>= 10^{-8} to 10^{-7}
SIL 2	>= 10^{-7} to 10^{-6}
SIL 1	>= 10^{-6} to 10^{-5}

AEROSPACE (DO-178C)

Software Level	Effect of software anomalous behavior
Level A	Multiple loss of life, usually with loss of aircraft
Level B	The aircraft, or crew, is less capable to deal with adverse operating conditions
Level C	The aircraft, or crew, is less able to deal with unfavorable operational conditions
Level D	No significant reduction in the aircraft's level of safety
Level E	*No effect on safety*

DO-178C Software Levels

MEDICAL (62304)

- Class A: No injury or damage to health possible
- Class B: Non-SERIOUS INJURY is possible
- Class C: Death or SERIOUS INJURY is possible

How to control probability of failure during development?

Table A.3 – Software Architecture (7.3)

TECHNIQUE/MEASURE	Ref	SIL 0	SIL 1	SIL 2	SIL 3	SIL 4
1. Defensive Programming	D.14	-	HR	HR	HR	HR
2. Fault Detection & Diagnosis	D.26	-	R	R	HR	HR
3. Error Correcting Codes	D.19	-	-	-	-	-
4. Error Detecting Codes	D.19	-	R	R	HR	HR
5. Failure Assertion Programming	D.24	-	R	R	HR	HR
6. Safety Bag Techniques	D.47	-	R	R	R	R
7. Diverse Programming	D.16	-	R	R	HR	HR
8. Recovery Block	D.44	-	R	R	R	R
9. Backward Recovery	D.5	-	NR	NR	NR	NR
10. Forward Recovery	D.30	-	NR	NR	NR	NR
11. Retry Fault Recovery Mechanisms	D.46	-	R	R	R	R
12. Memorising Executed Cases	D.36	-	R	R	HR	HR
13. Artificial Intelligence – Fault Correction	D.1	-	NR	NR	NR	NR
14. Dynamic Reconfiguration of software	D.17	-	NR	NR	NR	NR
15. Software Error Effect Analysis	D.25	-	R	R	HR	HR
16. Graceful Degradation	D.31	-	R	R	HR	HR
17. Information Hiding	D.33	-	-	-	-	-
18. Information Encapsulation	D.33	R	HR	HR	HR	HR
19. Fully Defined Interface	D.38	HR	HR	HR	M	M
20. Formal Methods	D.28	-	R	R	HR	HR
21. Modelling	Table A.17	R	R	R	HR	HR
22. Structured Methodology	D.52	R	HR	HR	HR	HR
23. Modelling supported by computer aided design	Table	R	R	R	HR	HR

TECHNIQUE/MEASURE	Ref	SIL 0	SIL 1	SIL 2	SIL 3	SIL 4
1. Formal Proof	D.29	-	R	R	HR	HR
2. Static Analysis	Table A.19	-	HR	HR	HR	HR
3. Dynamic Analysis and Testing	Table A.13	-	HR	HR	HR	HR
4. Metrics	D.37	-	R	R	R	R
5. Traceability	D.58	R	HR	HR	M	M
6. Software Error Effect Analysis	D.25	-	R	R	HR	HR
7. Test Coverage for code	Table A.21	R	HR	HR	HR	HR
8. Functional/ Black-box Testing	Table A.14	HR	HR	HR	M	M
9. Performance Testing	Table A.18	-	HR	HR	HR	HR
10. Interface Testing	D.34	HR	HR	HR	HR	HR

Requirements:

1) For software Safety Integrity Levels 3 and 4, the approved combination of techniques is 3, 5, 7, 8 and one from 1, 2 or 6.

2) For Software Safety Integrity Level 1 and 2, the approved combinations of techniques is 5 together with one from 2, 3 or 8.

NOTE 1 Techniques/measures 1, 2, 4, 5, 6 and 7 are for verification activities.

NOTE 2 Techniques/measures 3, 8, 9 and 10 are for testing activities.

Table A.12 – Coding Standards

TECHNIQUE/MEASURE	Ref	SIL 0	SIL 1	SIL 2	SIL 3	SIL 4
1. Coding Standard	D.15	HR	HR	HR	M	M
2. Coding Style Guide	D.15	HR	HR	HR	HR	HR
3. No Dynamic Objects	D.15	-	R	R	HR	HR
4. No Dynamic Variables	D.15	-	R	R	HR	HR
5. Limited Use of Pointers	D.15	-	R	R	R	R
6. Limited Use of Recursion	D.15	-	R	R	HR	HR
7. No Unconditional Jumps	D.15	-	HR	HR	HR	HR
8. Limited size and complexity of Functions, Subroutines and Methods	D.38	HR	HR	HR	HR	HR
9. Entry/Exit Point strategy for Functions, Subroutines and Methods	D.38	R	HR	HR	HR	HR
9. Limited number of subroutine parameters	D.38	R	R	R	R	R
10. Limited use of Global Variables	D.38	HR	HR	HR	M	M

Requirement:

1) It is accepted that techniques 3, 4 and 5 may be present as part of a validated compiler or translator.

TECHNIQUE/MEASURE	Ref	SIL 0	SIL 1	SIL 2	SIL 3	SIL 4
1. ADA	D.54	R	HR	HR	HR	HR
2. MODULA-2	D.54	R	HR	HR	HR	HR
3. PASCAL	D.54	R	HR	HR	HR	HR
4. C or C++	D.54 D.35	R	R	R	R	R
5. PL/M	D.54	R	R	R	NR	NR
6. BASIC	D.54	R	NR	NR	NR	NR
7. Assembler	D.54	R	R	R	R	R
8. C#	D.54 D.35	R	R	R	R	R
9. JAVA	D.54 D.35	R	R	R	R	R
10. Statement List	D.54	R	R	R	R	R

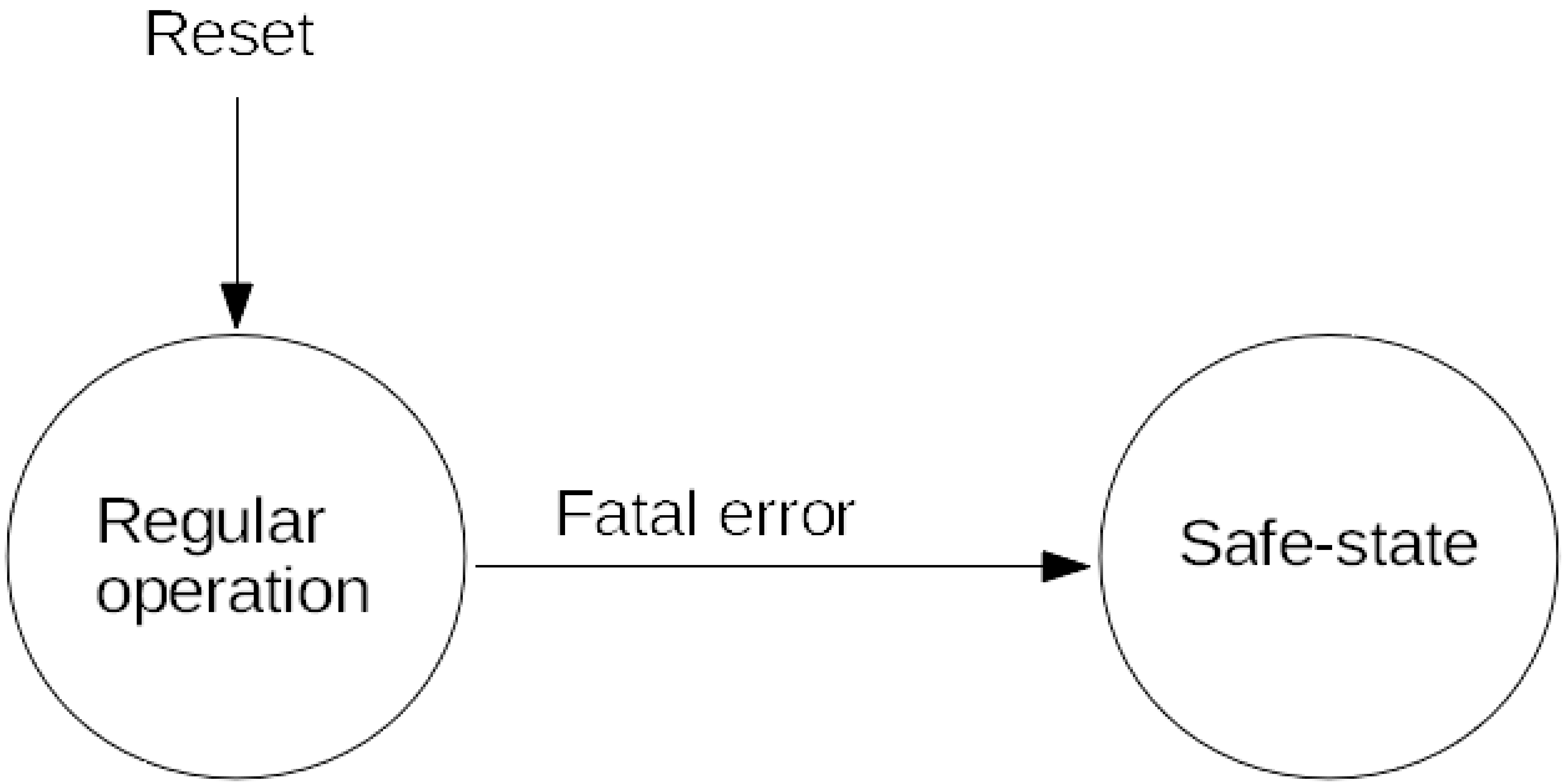
SAFETY AT SYSTEM LEVEL?

- You cannot finish implementation first and then introduce safety.
- You cannot implement safe modules and expect resulting system to be safe
- You must care about safety from the start

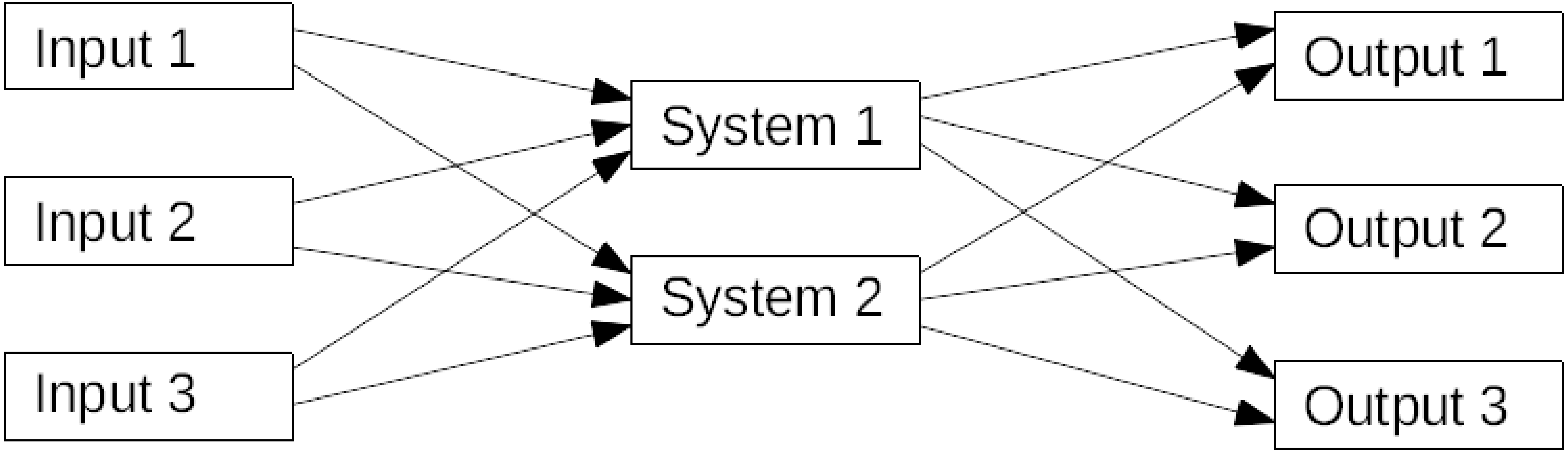
RISK ANALYSIS

RISK ASSESSMENT MATRIX				
SEVERITY / PROBABILITY	Catastrophic (1)	Critical (2)	Marginal (3)	Negligible (4)
Frequent (A)	High	High	Serious	Medium
Probable (B)	High	High	Serious	Medium
Occasional (C)	High	Serious	Medium	Low
Remote (D)	Serious	Medium	Medium	Low
Improbable (E)	Medium	Medium	Medium	Low
Eliminated (F)	Eliminated			

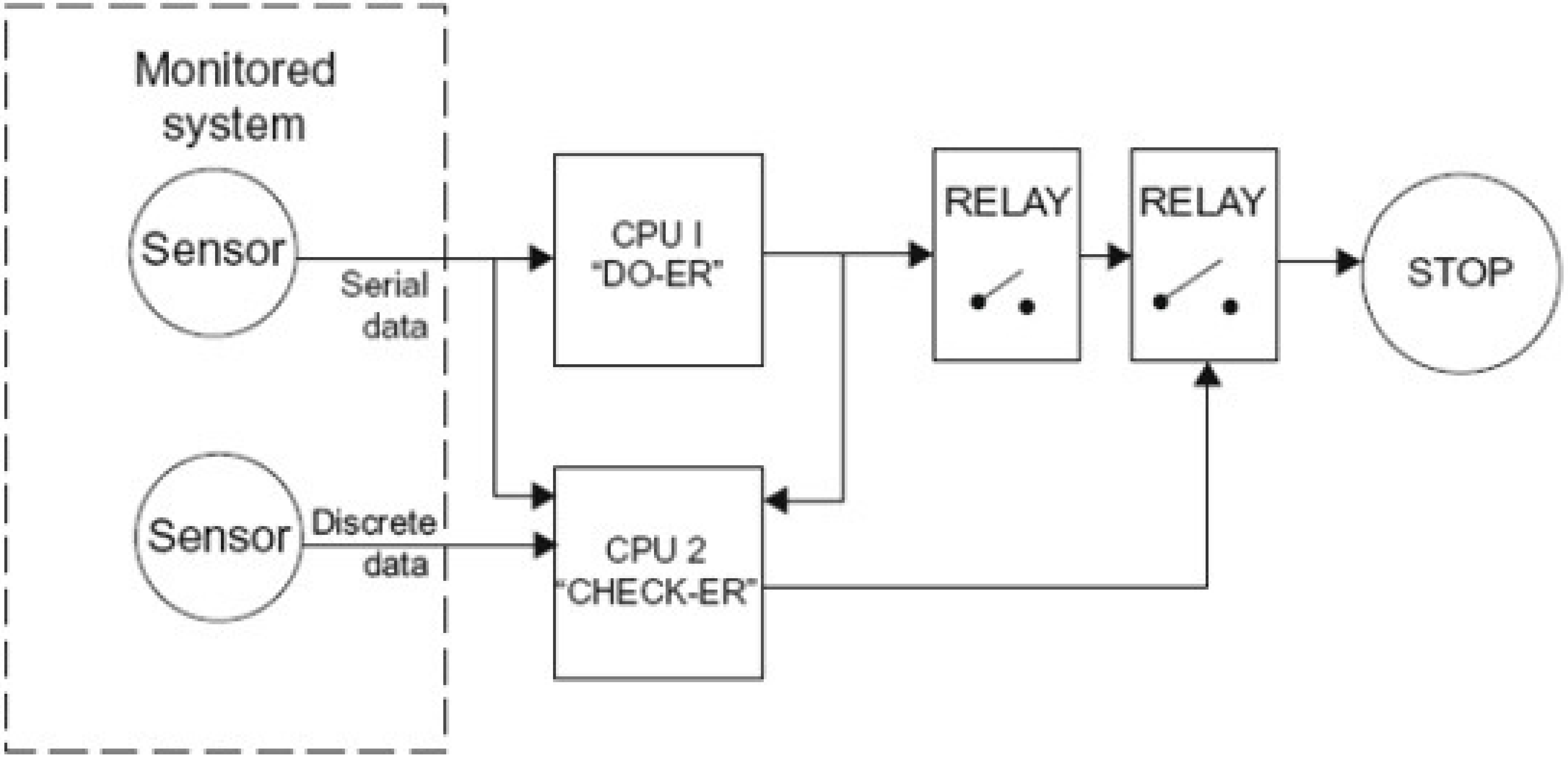
SAFE STATE



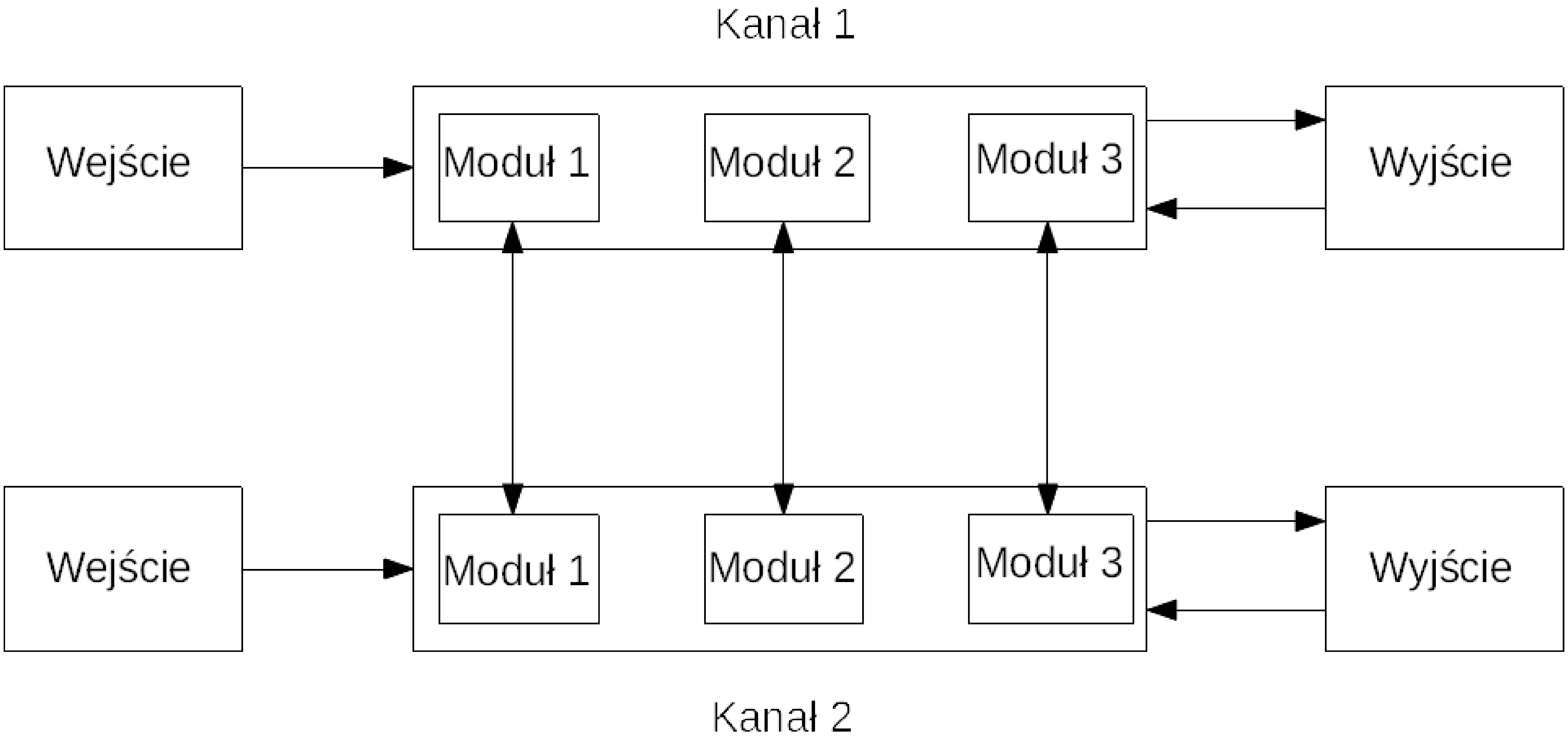
REDUNDANCY


Input 1
Input 2
Input 3
System 1
System 2
Output 1
Output 2
Output 3

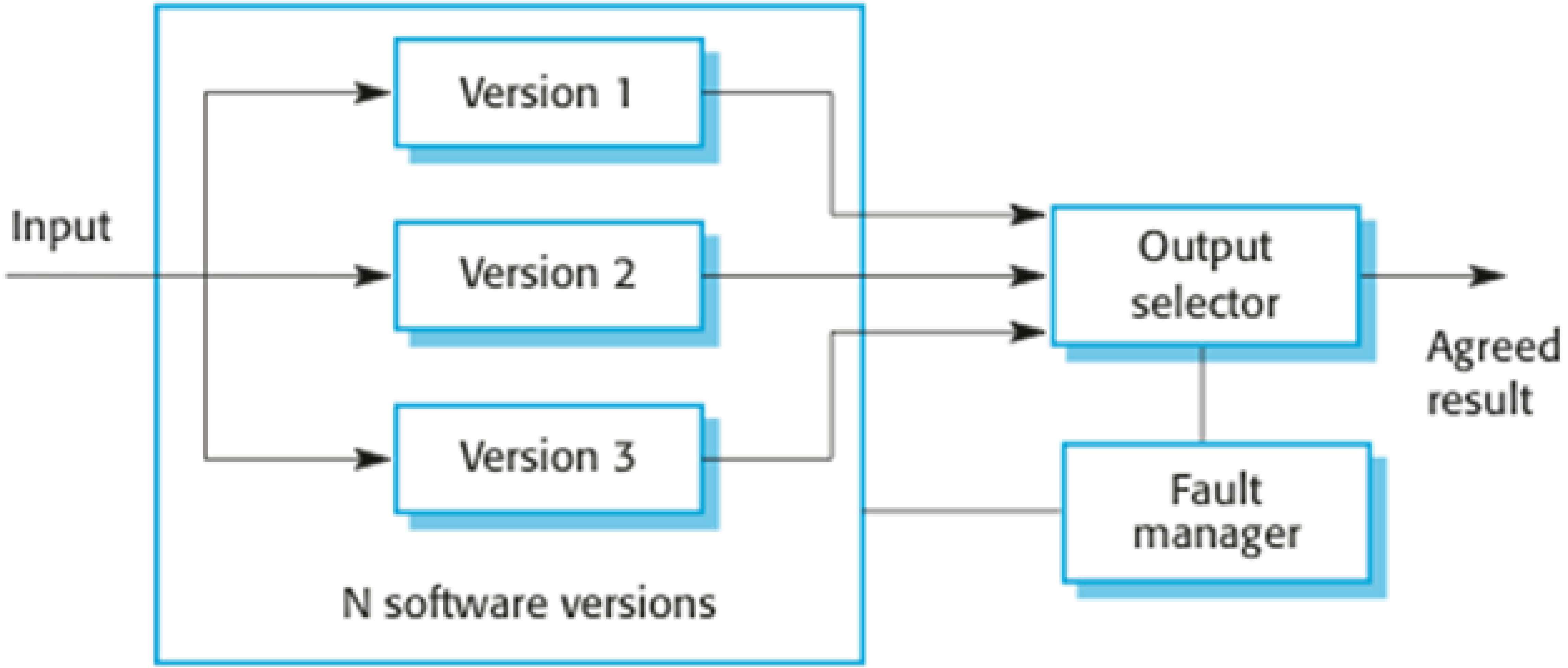
SUPERVISOR CPU



INDEPENDENT CHANNELS

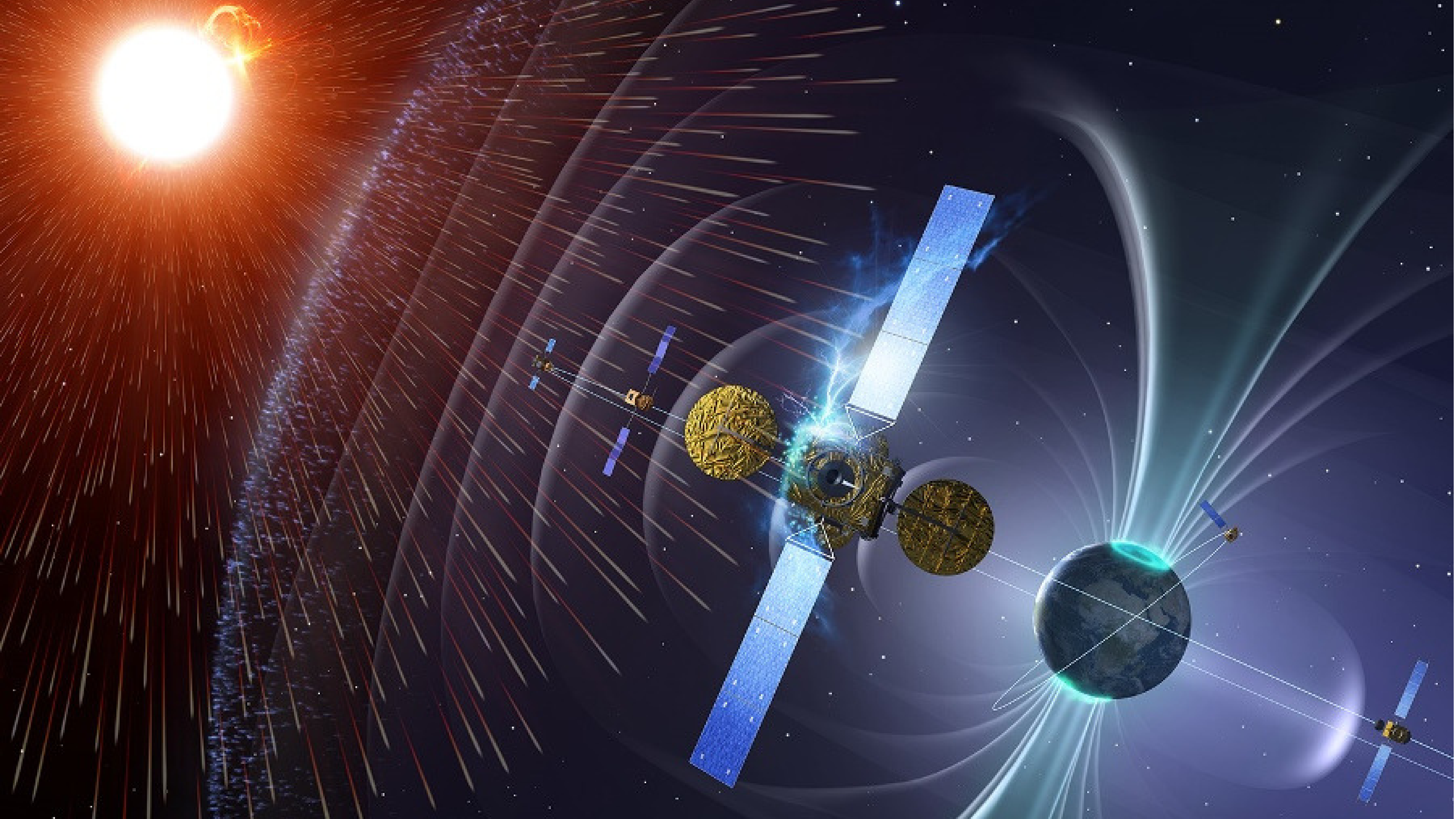


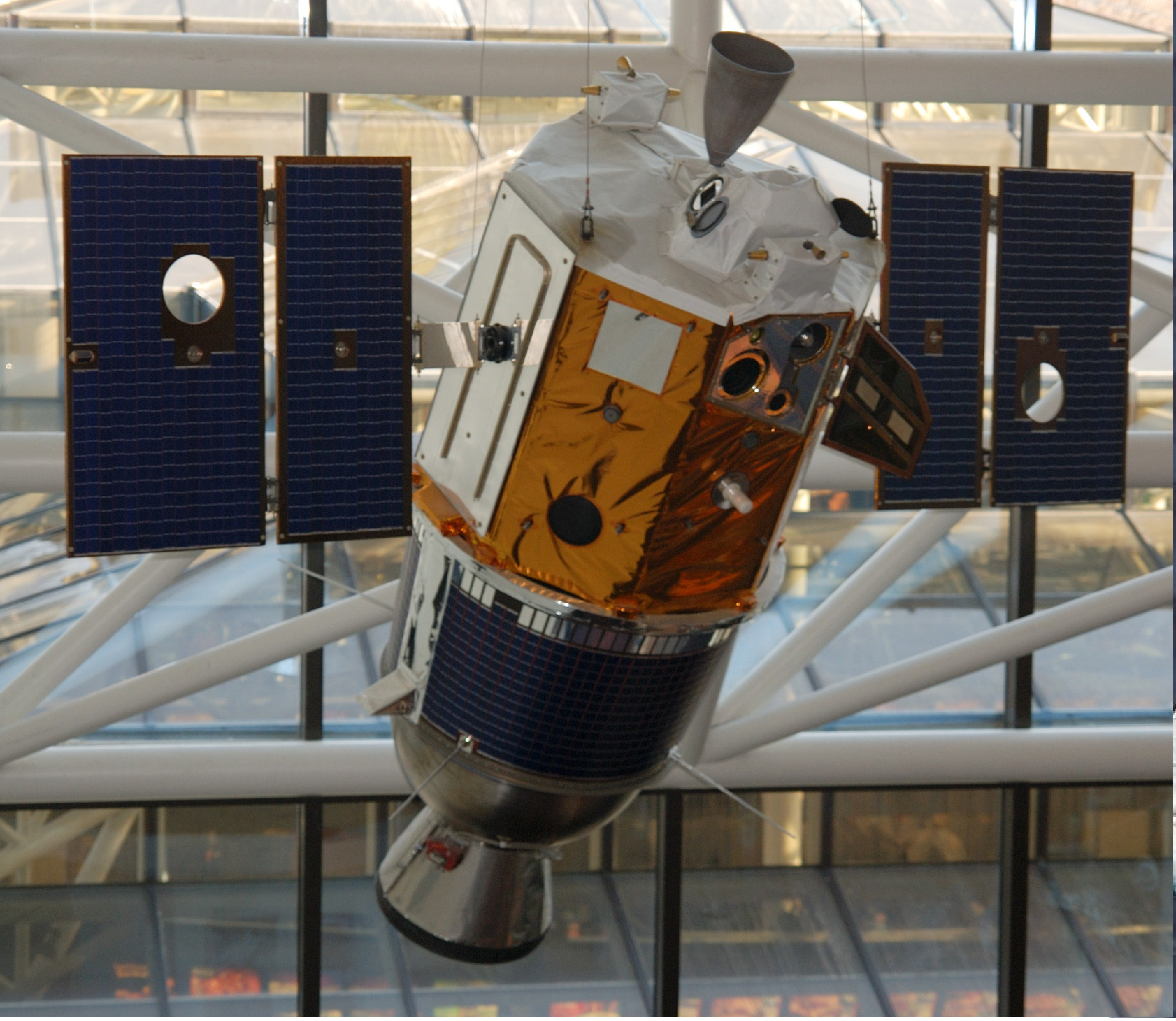
VOTING SYSTEM



DIVERSE PROGRAMMING

- Channels implemented by independent teams
- Teams don't exchange information
- Teams share documentation
- Reducing risk of the same software errors
- Possible ways of diversification: hardware, programming languages, techniques

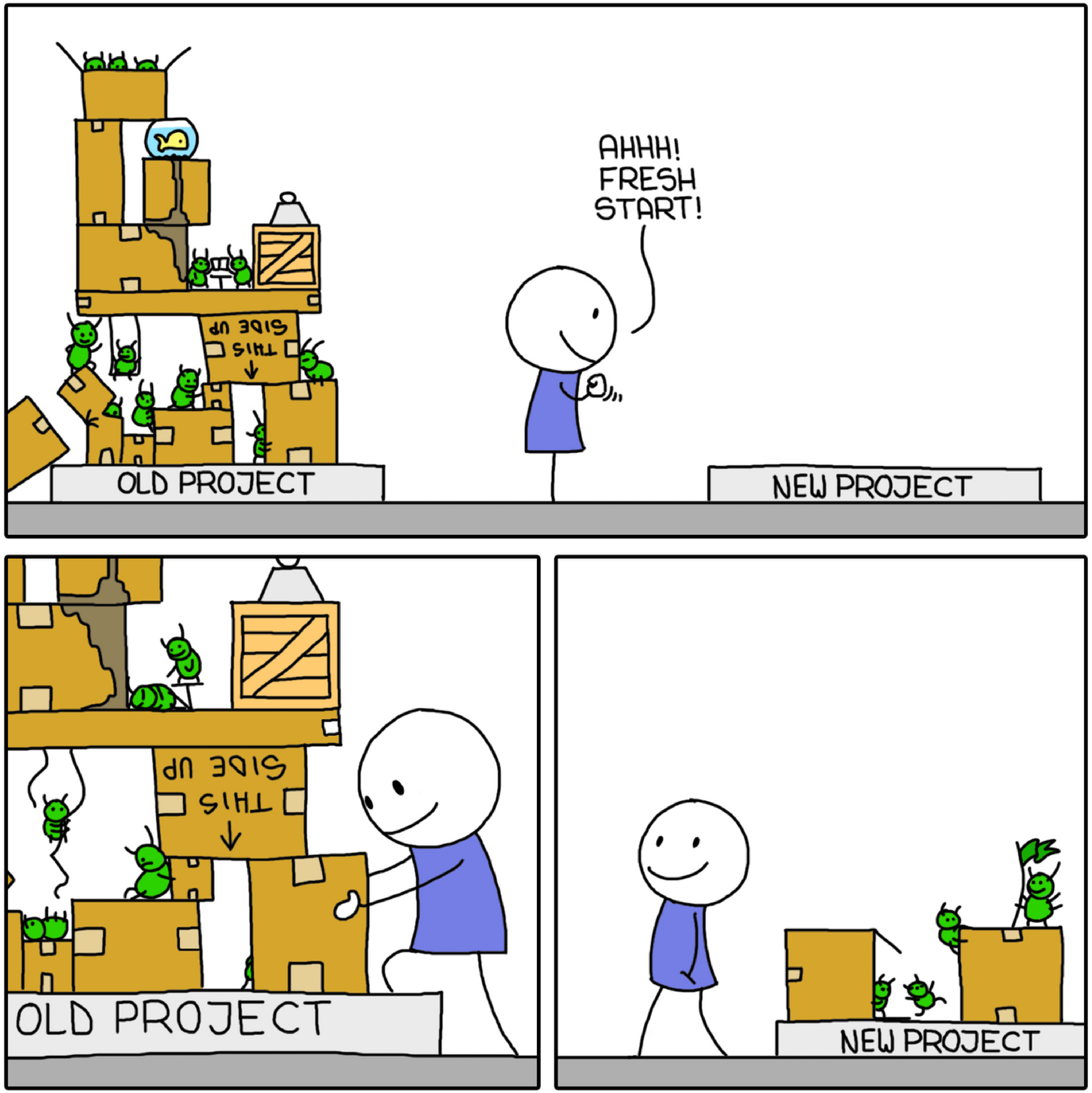




SANITY CHECKS

- RAM tests
- Non-volatile Memory tests
- CPU Tests - registers and instructions
- Power supply tests
- Sensor tests
- Clock system tests

CODE REUSE

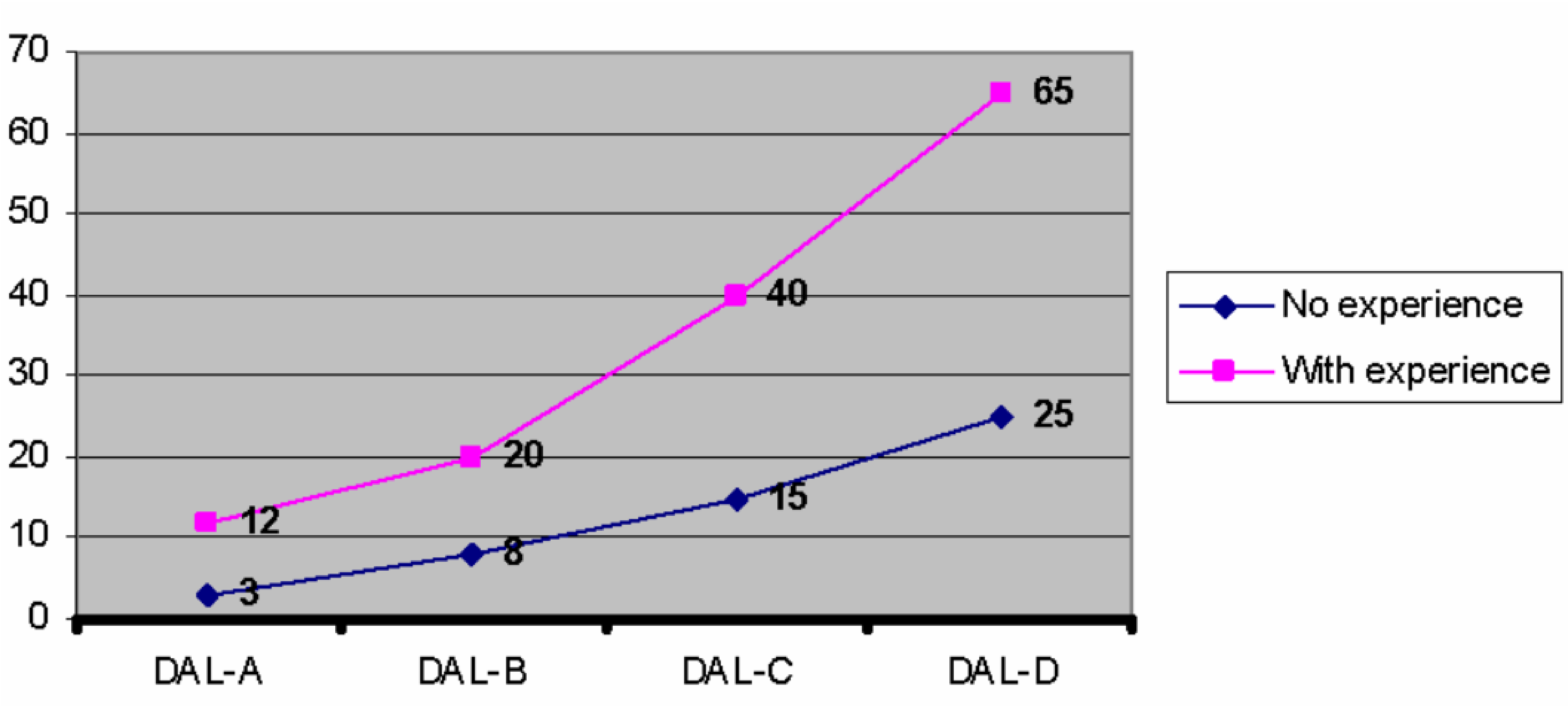


THIS SIDE UP
AHHH! FRESH START!
OLD PROJECT
NEW PROJECT
THIS SIDE UP
OLD PROJECT
NEW PROJECT

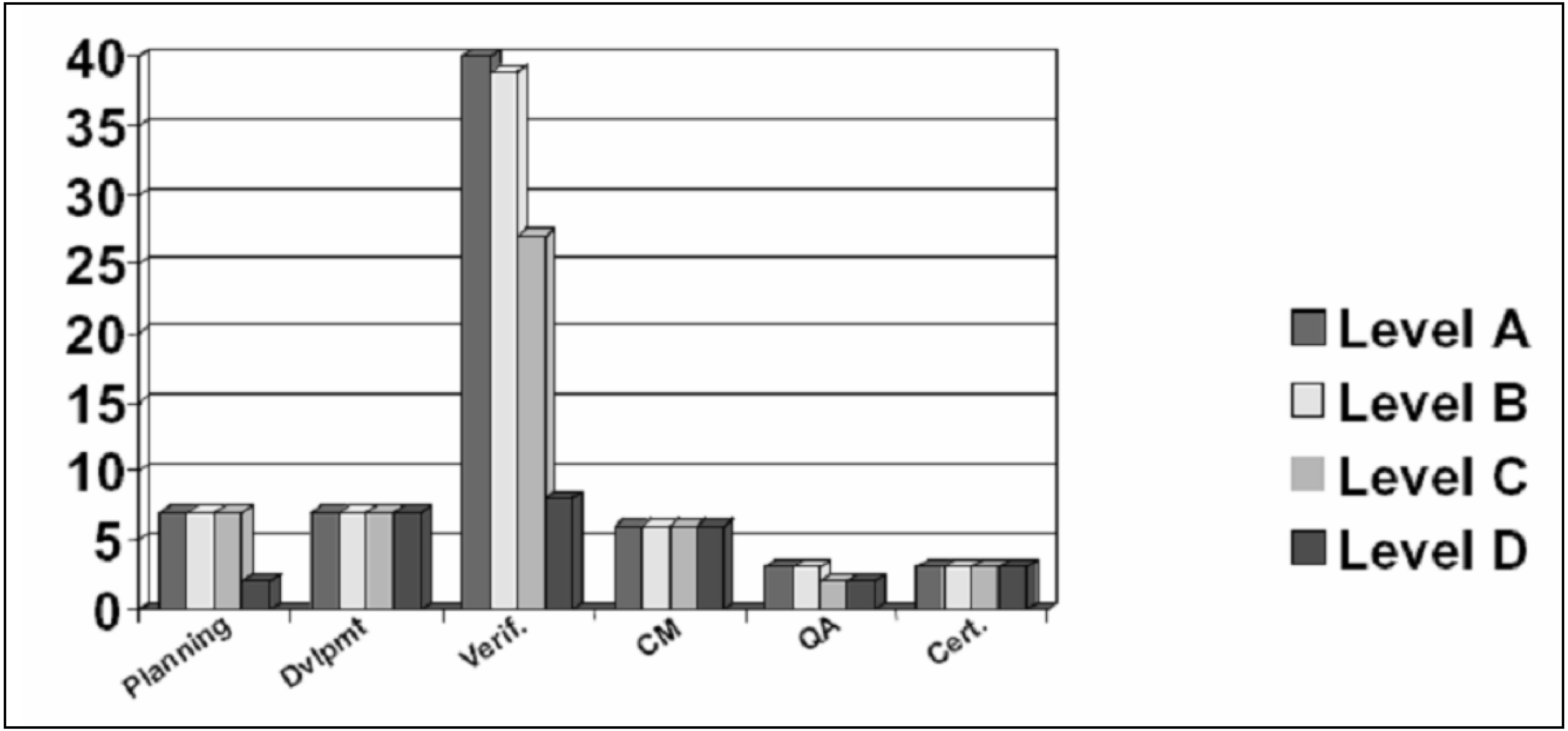
SOUP - SOFTWARE OF UNKNOWN PROVENANCE

- Insufficient documentation
- No risk assesment
- Unknown development and testing procedures
- Dedicated commercial libraries turn out to be cheaper

LINES OF CODE PER HOUR



DISTRIBUTION OF ACTIVITIES IN SAFETY-CRITICAL SYSTEM



SHOULD WE BE AFRAID OF REGULATIONS?

Read more:

https://ucgosu.pl/safety-critical-eng/

@MaciekGajdzica

THANK YOU FOR YOUR ATTENTION!