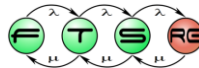


Testing of a safe driver-machine interface for ERTMS train control (SAFEDMI)

Software Verification Techniques course

István Majzik
 Budapest University of Technology and Economics
 Dept. of Measurement and Information Systems



Outline

- What is ERTMS?
- Goals of the project
- The DMI architecture
- Test groups
 - Testing the ERTMS functions
 - Testing robustness
 - Testing the internal safety mechanisms
 - Testing the wireless communications
- Summary of the evaluation techniques

What is ERTMS?

- European Rail Traffic Management System
 - Single Europe-wide standard for **train control and command** systems
- Main components:
 - European Train Control System (ETCS): standard for **in-cab train control**
 - GSM-R: the GSM mobile **communications standard** for railway operations (from/to control centers)
- Equipment used:
 - **On-board equipment**: e.g., **EVC** European Vital Computer for on-board train control
 - **Infrastructure equipment**: e.g., **balise**, an electronic transponder placed between the rails to give the exact location of a train



3



The SAFEDMI project

Safe Driver Machine Interface for ERTMS Automatic Train Control (SAFEDMI, supported by the European Community in FP6)



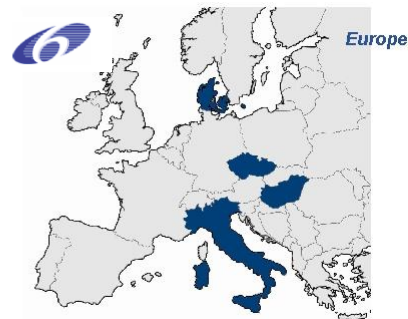
Partners:



Project Co-ordinator



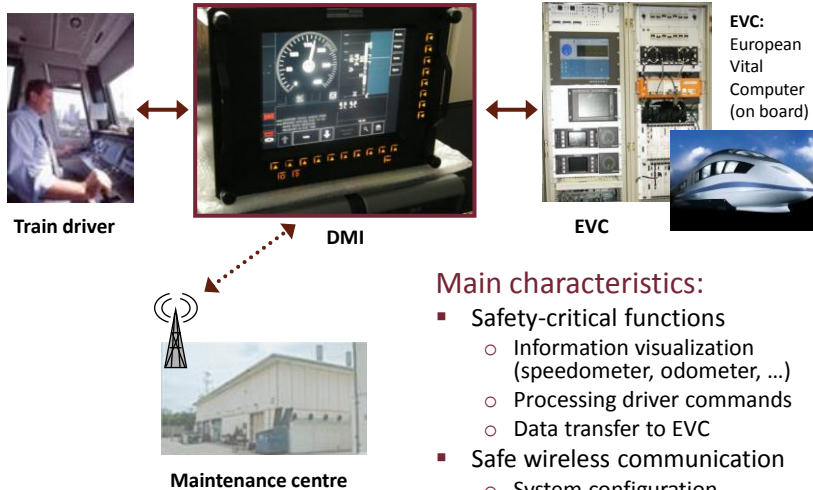
ISTITUTO DI SCIENZA E TECNOLOGIE DELL'INFORMAZIONE "A. FAEDO"



4



Development of a safe DMI



Main characteristics:

- Safety-critical functions
 - Information visualization (speedometer, odometer, ...)
 - Processing driver commands
 - Data transfer to EVC
- Safe wireless communication
 - System configuration
 - Diagnostics
 - Software update



5



Requirements

- Safety:
 - Safety Integrity Level: **SIL 2**
 - Tolerable Hazard Rate: **$10^{-7} \leq \text{THR} < 10^{-6}$**
(hazardous failures per hours)
 - CENELEC standards: EN 50129 and EN 50128
- Reliability:
 - Mean Time To Failure: **MTTF > 5000 hours**
(5000 hours: ~ 7 months)
- Availability:
 - **$A = \text{MTTF} / (\text{MTTF} + \text{MTTR}), A > 0.9952$**
Faulty state: shall be less than 42 hours per year
MTTR < 24 hours if MTTF=5000 hours



6



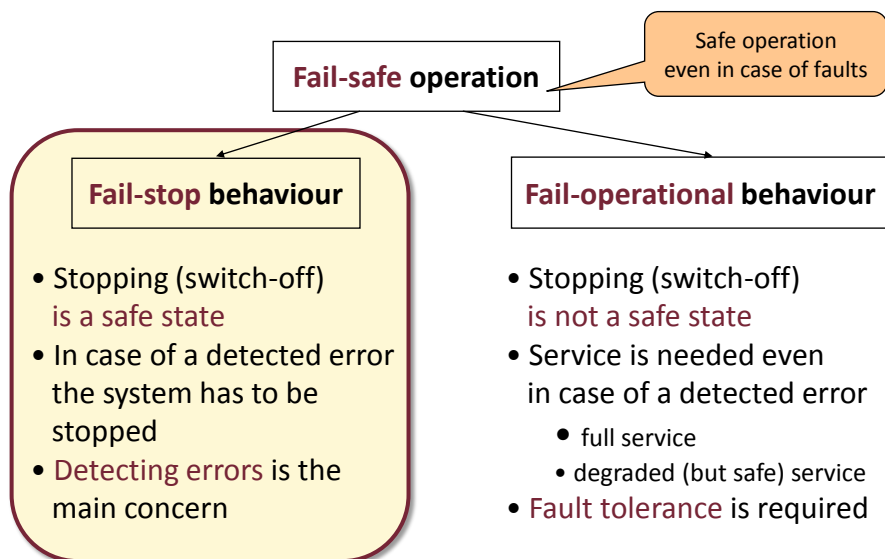
Design of the DMI



7



Operational concerns

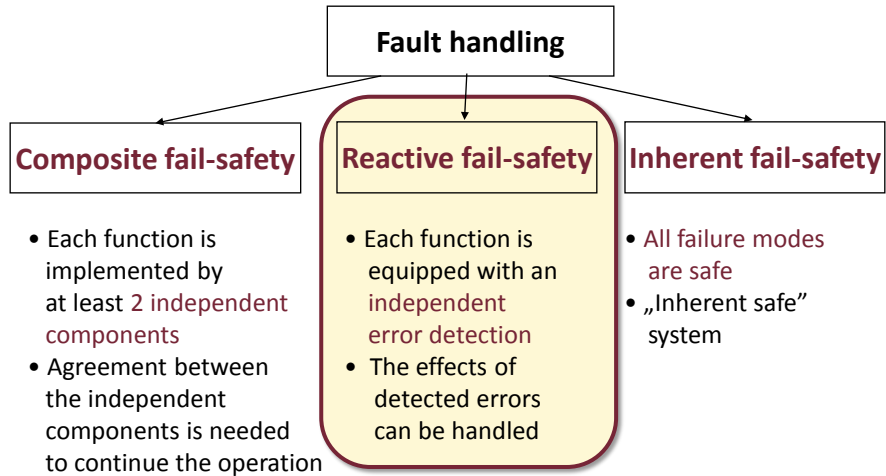


8



Fail-safety concerns

Safety in case of single random hardware faults

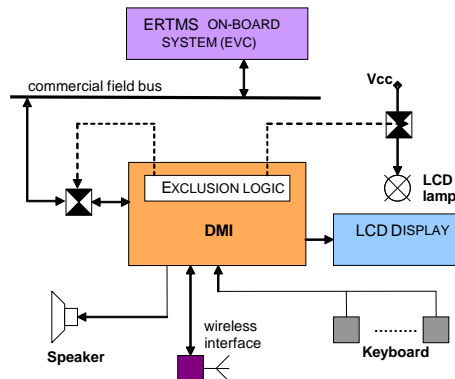


9



The SAFEDMI hardware concept

- Single electronic structure based on **reactive fail-safety**
- Generic (off-the-shelf) hardware components are used
- Most of the safety mechanisms are **based on software implemented error detection and error handling**

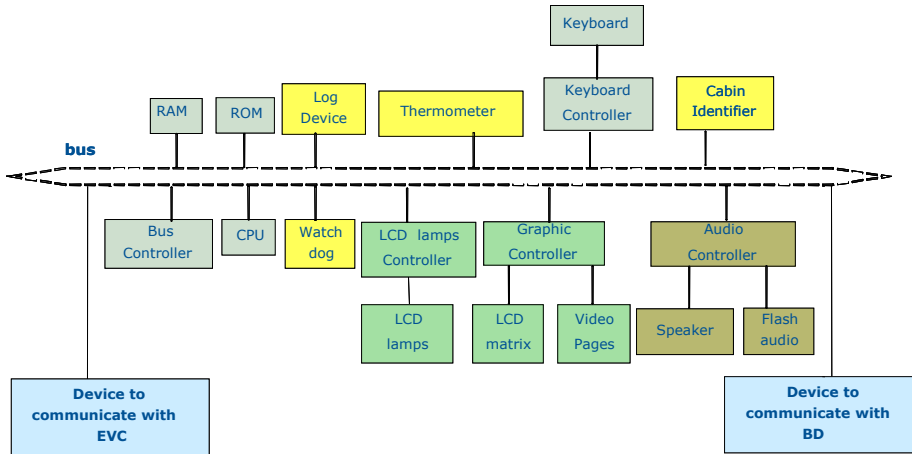


10



The SAFEDMI hardware architecture

Commercial hardware components:

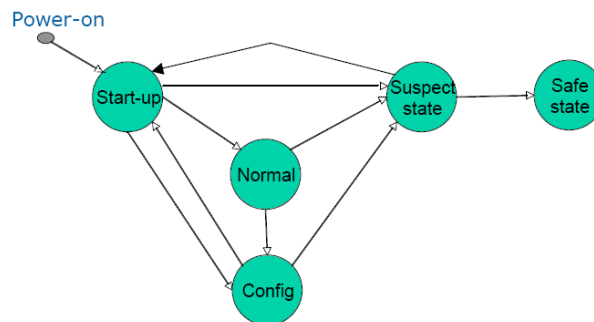


11



The SAFEDMI fault handling

- Operational modes:
 - Startup, Normal, Configuration and Safe (stopped) modes
 - **Suspect state** to implement controlled restart/stop after error: counting occurrences of errors in a given time period; forcing to Safe state (stop) in a given limit is exceeded



12



Error detection in Startup mode

Detection of permanent hardware faults by thorough self-testing

- Memory testing:
 - March algorithms (for stuck-at and coupling faults): writing and reading back regular 1 and 0 patterns stepwise
- CPU testing:
 - External watchdog circuit: Basic functionality (starting, heartbeat)
 - Self-test of functions: Core functionality → complex functionality (instruction decoding, register decoding, internal buses, arithmetic and logic unit)
- Integrity of software (in EEPROM):
 - Error detection codes
- Device testing (speaker, keyboard etc.):
 - Operator assistance is needed



13



Error detection in Normal/Config mode

- Hardware devices:
 - Scheduled low-overhead memory, video page and CPU tests
 - Acceptance checks for I/O
- Communication and configuration functions:
 - Assertions for data acceptance / credibility checks of internal data
 - Error detection and correction codes for messages
- Operation mode control and driver input processing:
 - Control flow monitoring (based on the program control flow graph)
 - Time-out checking for operations
 - Acknowledgement procedure: the driver shall confirm risky operations
- Visualization of train data (bitmap computations):
 - Duplicated computation and comparison of the results
 - Visual comparison by the driver (periodic change of bitmaps)



14



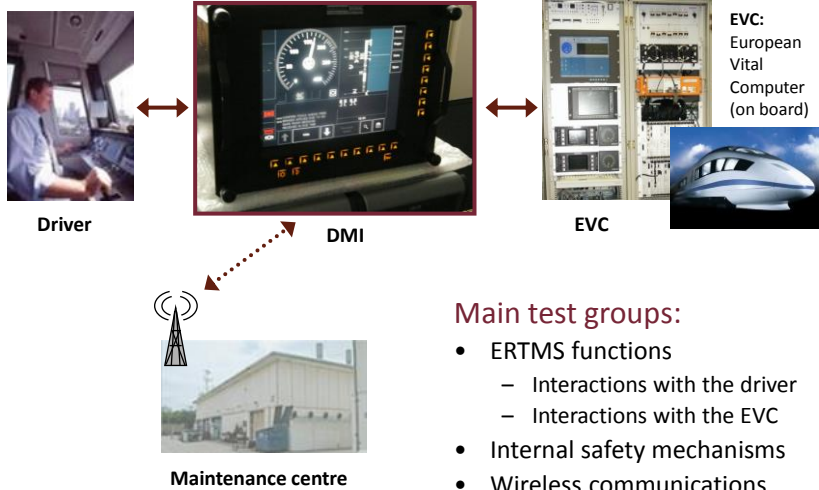
Testing the DMI



15



Testing goals



Main test groups:

- ERTMS functions
 - Interactions with the driver
 - Interactions with the EVC
- Internal safety mechanisms
- Wireless communications






16



Testing the ERTMS functions

- Sequences of test inputs: DMI inputs + workload
- Test output: DMI display + Diagnostic device

Step	Action	Expected Event
1.	Driver: give traction to the train	SAFEDMI: the current train speed increases.
2.	None	SAFEDMI: <ul style="list-style-type: none"> • The text message "Entry in Full Supervision Mode" is shown and a sound is produced. • the FS mode icon  is shown in area B7; • in area A2 the distance to target is shown;
3.	Driver: give traction to the train until the current train speed overcomes the permitted speed.	SAFEDMI: <ul style="list-style-type: none"> - In area A1 the warning to avoid brake intervention is displayed and sound is produced; - In area E1 the icon  (Brake applied) is shown; • In area C9 the icon  (Service brake intervention or emergency brake intervention) is shown.



17



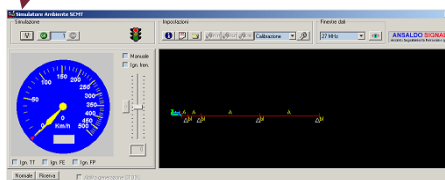
Test environment



Track simulator



ERTMS OnBoard equipment



Simulating the workload:

- signals from balises on a given route
- control messages from the railway regulation control center

Plus: Diagnostic device



18



Output of the diagnostic device

Offset	Format	Remark	Value
[00]	novramArea.xLogger.LDEC	novramArea.xLogger.ulNumReg	100

Offset	Format	Remark	Value
[00]	novramArea.xLogger.LDEC	novramArea.xLogger.ulNextIdx	100

Offset	Format	Remark	Value
[00]	novramArea.xLogger.LCNV	novramArea.xLogger.xData[0].ulTimestamp	2
[04]	novramArea.xLogger.LBSC	Evento	EVENT_FLT_INJECT_START
[08]	novramArea.xLogger.LHEX	novramArea.xLogger.xData[0].ulParam1	00000002
[0C]	novramArea.xLogger.LDEC	novramArea.xLogger.xData[0].ulParam2	1000



19



Robustness testing



Driver



DMI



EVC

- Focus: Exceptional and extreme inputs, overload
- Testing behaviour on the driver interface:
 - Handling buttons: pressing more buttons simultaneously, ...
 - Input fields: empty, full, invalid characters, ...
- Testing behaviour on the EVC interface:
 - Invalid messages: empty, garbage, invalid fields, flooding, ...



20



Testing the internal mechanisms

- **Operational modes** and the corresponding functions
 - Activation of operational modes, configuration, disconnection from the environment
 - Coverage of the **state machine of the operational modes**
 - Coverage of the **state machine of error counting**
- **Performance:** Testing deadlines in case of maximum workload (specified on the EVC interface)
- **Handling of buttons:** Blocked buttons, safety acknowledgements, ordering of events
- **Handling temperature sensors:** Startup and operational temperature conditions (tested in climate test chamber)



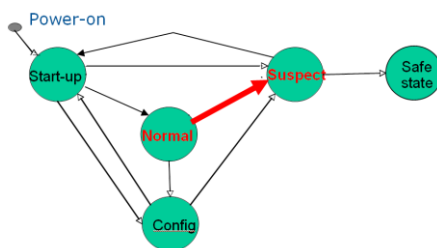
21



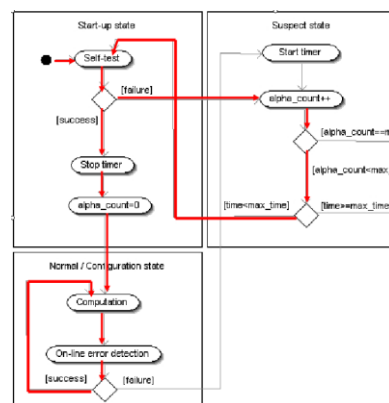
Systematic testing

- **Testing the operational modes:**

- Covering each state and each state transition



State machine of the operational modes



State machine of error counting



22



Testing the internal safety functions

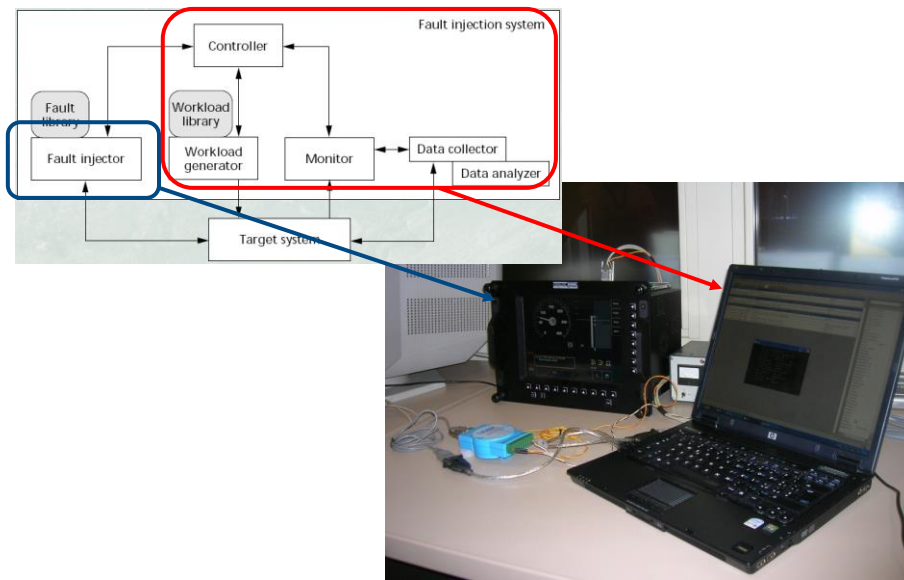
- **Targeted fault injection:** Testing the implementation of the software based error detection and error handling mechanisms
 - Test goals:
 - The injected errors are **detected** by the implemented mechanisms
 - The proper error handling is **triggered**
 - Tested error detection mechanisms:
 - Control flow checking, data acceptance checking, duplicated execution and comparison, time-out checking
- **Random fault injection:** Evaluation of **error detection coverage**
 - Collecting data for coverage statistics
- Checking hardware self-tests in specific configurations
 - Hardware checks (RAM, ROM, video page)
 - I/O device checks (cabin, LCD, temperature)



23



Software based fault injection



24



Collecting diagnostic data

The screenshot shows the Fault Injector application window. The main area displays a list of memory addresses and their corresponding values for various DMI parameters. The parameters are grouped by address:

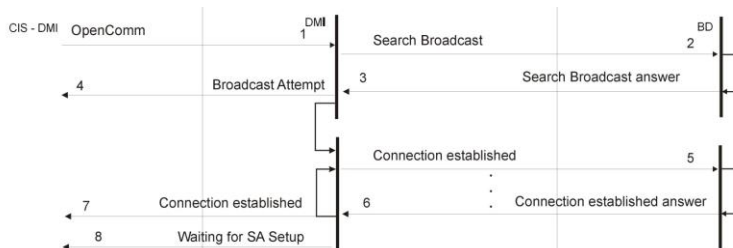
- DMI:mLocalTime** (Group Addr: 010EA63C): Offset, Format, Remark, Value.
- DMI:mLastTime** (Group Addr: 010EA638): Offset, Format, Remark, Value.
- DMI:ubFaultStep** (Group Addr: 010EA640): Offset, Format, Remark, Value.
- DMI:ulTotalFaultCounter** (Group Addr: 010EA644): Offset, Format, Remark, Value.
- DMI:ulFaultCounter[0]** (Group Addr: 010EA648): Offset, Format, Remark, Value.
- DMI:faulParam[0]** (Group Addr: 010FB8E4): Offset, Format, Remark, Value.
- DMI:ubFltSequenceId** (Group Addr: 010EA6F6): Offset, Format, Remark, Value.
- DMI:ulFaultPeriod** (Group Addr: 010EA6F8): Offset, Format, Remark, Value.

The status bar at the bottom shows the file path: C:\SafeDmi\diag\FaultInjector.dat.

25

Testing the wireless communication

- **Scenario based testing:** Communication scenarios
- **Normal operation:**
 - Protocol testing: Establishing connection, message processing, closing the connection
- **Operation in case of transmission errors:**
 - Error detection mechanisms (EDC, ECC)
 - Closing the connection in case of too frequent errors



26

