Requirements analysis

Marco Di Natale

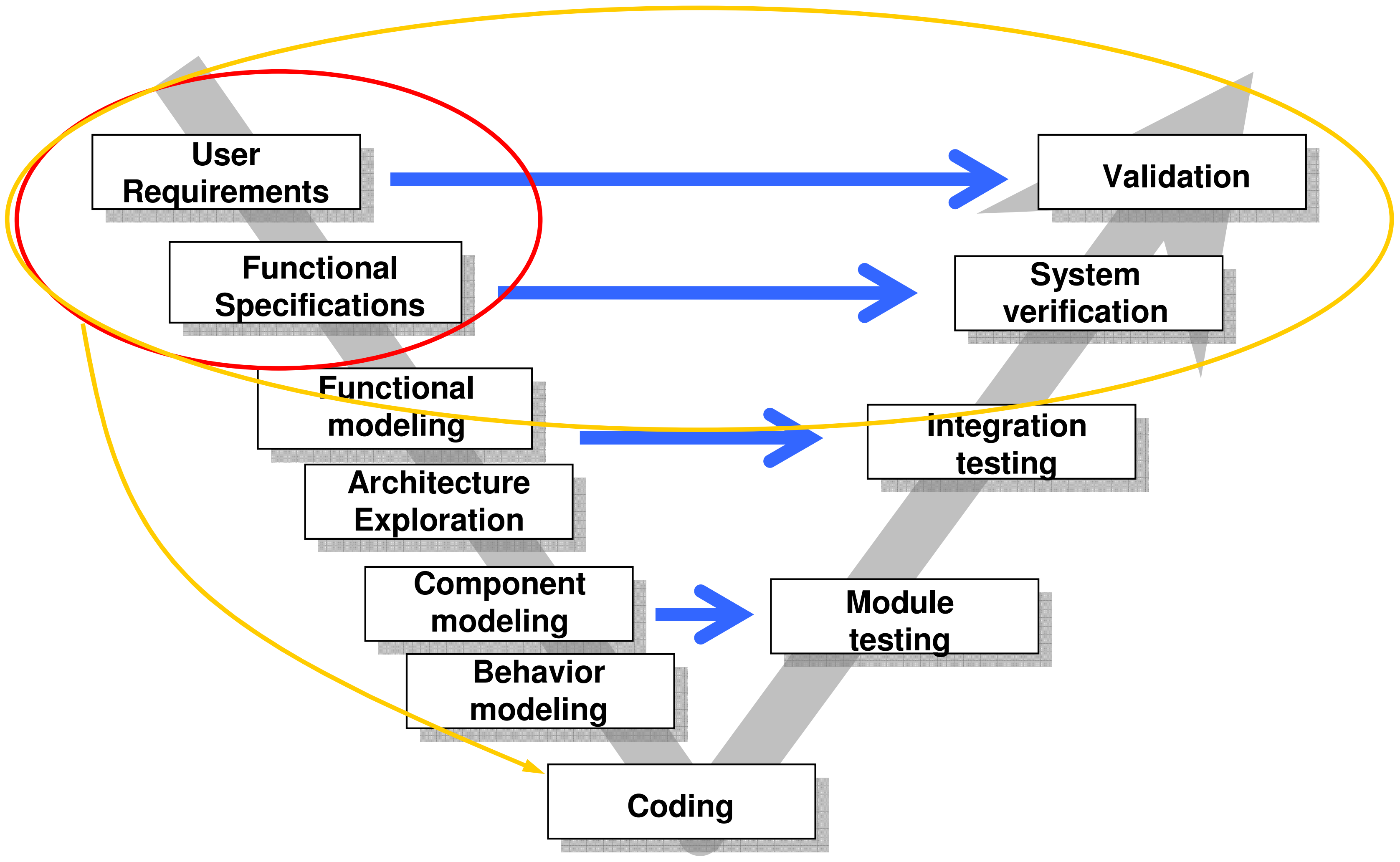
Scuola Superiore S. Anna- Pisa, Italy

Requirements Analysis

Purpose of this Lesson

- Establishing role of requirements and requirements vs testing
- Some methods for collecting Requirements in a structured way
  - Avoiding inconsistencies, redundancy, omissions
  - Go towards a formal model
    - Assumptions/Assertions
    - Preconditions/Postconditions/Invariants
    - FSM models
    - Other diagrams (MSCs, Context)
- Link one or more test cases to requirements
  - Requirements coverage
- Tracking requirements to implementation

The V-shape development cycle (V-model)



Requirements analysis

- (User) Requirements should be the description of what the system is supposed to do (what is its behavior), as opposed on how it will perform its tasks.
- In a cascade model (the V-model is no exception) requirements are the starting point. You don’t design until you know what you are supposed to produce.
- They should be obtained after discussion with marketing people, product specialists and other analysts.
- As such, they are better communicated in (informal) natural language (plain english).
- Unfortunately natural language is subject to misinterpretation due to ambiguity, inconsistencies, omissions and redundancy.
- Requirements analysis is quite often the result of an empirical process, subject to arbitriariety, imprecision, subjectivity.
- This is why formal requirements have been proposed
  - FSM, Z language …

Requirements analysis

- Conflicting needs:
  - Communicate with non-technical (non computer scientists) people using non-technical languages
  - Describe the specification of the system in a formal way to avoid ambiguities/inconsistencies …
- In addition, requirements are often incompletely defined at the beginning of a project and completed iteratively while the system is developed (and the needs become clearer)
  - Or often, never completed at all!
- Requirements are alive, they evolve and are (iteratively) refined, redefined or reorganized.

Requirements analysis

In reality, in most cases, requirements documents are produced after the system is developed and most often, they are written by the developers themselves, documenting what the system does …

I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
E-mail: SCOTTADAMS@AOL.COM
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
© 2006 Scott Adams, Inc./Dist. by UFS, Inc.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
www.dilbert.com
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?

Requirements analysis: A proposed approach

Several techniques can be used to drive the requirements capture and analysis stage and to organize the requirements.

The objectives are:

- Clarity
- Completeness
- Avoiding abiguities and inconsistencies
- Avoiding redundancy

Different techniques may be used depending on the type of system that must be developed (database vs embedded) and according to the methodology and development process (object oriented vs functional)

For example, techniques may be classified according to the starting points for the analysis

- Working modes or states of the system
- The model of the data or information that the system must handle
- The Use cases of the system and the actors interacting with the system
- The activities of the system (or its reactions), typically associated with
- Le attivita’ che il sistema si suppone debba realizzare, o la reazione del
  - The entities or actors interacting with the system
  - The events to which the system must react
- The objectives that the system must ultimately satisfy

Requirements analysis: A proposed approach

The concept:

- Write requirements in “structured” natural language
- Enforce writing style that tends towards a FSM description
- Enforce use of data dictionary
- Simple rules that try to avoid redundancy/inconsistency
- Complement/refine text with formal diagrams (semiformal approach)
- Ease definition of requirements-driven test cases

Requirements analysis: A proposed approach

- Requirements can be organized in chapters for subsystems and sections referring the system-level working modes.
- Inside each section, we place the requirements that apply to each working mode (system-level “states”)
  - This drives the definition of a finite state machine model
- Generally, at least a subset of the system reactions does not happen in isolation, but there is a protocol or sequence of events and reactions that are organized in a scenario.
- The scenarios defining the working or the use of the system can be organized in Sequence diagrams according to the UML standard or in Message Sequence Charts (in SDL).
  - Typical examples, error management and diagnostics
- Typically, the requirements express the large scale partitioning of the system in its main subsystems (functionally, not physically)
- Such a partitioning should highlight the definition of the inputs and outputs at the system/subsystem level

Requirements analysis: The data dictionary

- In the definition of the activities and states of the system and, especially of inputs, outputs, events and system parameters it is necessary to refer to names without ambiguity. This structure can be obtained in a very simple way, with the management of a data dictionary.
- Names in the dictionary do not refer program variables, but functional entities. We do not refer concrete types (float, double, int16), but abstract types (reale, intero ...)
- An example of data dictionary for input/output data and events is

Id	Signal Name	Text description	Is Trigger	Direction	Width (bits)	Dest/Src	Period (ms)	Data Type	Units	Min	Max	Resolution	Offset	Notes
I1	B_ECO	tasto ECO	TRUE	Input	1	User		Boolean		0	1	1	0	
I2	B_OFF	tasto ON/OFF	TRUE	Input	1	User		Boolean		0	1	1	0	
I3	B_MODE	tasto MODE	TRUE	Input	1	User		Boolean		0	1	1	0	
I10	T_NTC1	Misura della temperatura alla quale si trova la sonda NTC1	FALSE	Input	16	Plant	1000	Real	DegC	-30	128	0,00390625	0	
I11	T_NTC2	Misura della temperatura alla quale si trova la sonda NTC2	FALSE	Input	16	Plant	1000	Real	DegC	-30	128	0,00390625	0	
O1	L_ECO	led ECO		Output		User		Boolean		0	1	1	0	

- For parameters the schema is slightly different. There is no need to represent the direction (input/output) and there are no events.

Id	Parameter Name	Text description	Width (bits	Data Type	Units	Default	Min	Max	Resolution	Offset	Notes
alpha	t_step_eco		1	Boolean	DegC	3	0	0	1	0	
beta	t_confort		1	Boolean	DegC	53	0	0	1	0	
gamma	t_sv_ottimo	temperatura di svuotamento ottimo	1	Boolean	DegC	21	0	0	1	0	
delta	t_isteresi	isteresi di funzionamento	0	Real	DegC	8	0	0	1	0	

Redundant definitions

- *If unlocking is performed 5 times in less than 15 minutes, it is impossible to unlock the boiler for the next 5 minutes. Any further attempt to unlock it after the fifth increases the waiting time before the boiler can be unlocked again by 5 minutes. After the waiting time, it is possible to unlock again, on condition that less than 5 unlocks happened in the last 15 minutes. Otherwise, it is necessary to wait 15 minutes from the latest unlock.*

Unnecessary descriptions

- what the system doesn’t do
  - Of course it doesn’t do a lot of things ….
- If the interface is off all leds are off and *button presses do not have effect*
- In setup mode, *pressing the Economy button has no effect*. Pressure of the ON button in setup mode causes the selection of the last temporary setpoint and turns the device off.

Multiple names (states)

- After the acknowledgement of a non volatile error, the error signal persists even after cutting off and restoring power, whether the cause of the error persists or not.
- After the acknowledgement of a non volatile error, the error signal persists even if the cause of the error is no more present or (?) there is a power failure, until unlocking.
- When the cause of the error is no more present, the error signal will be removed only after a reset, that is a shutdown, followed by a startup.

Multiple names (I/O)

Device names

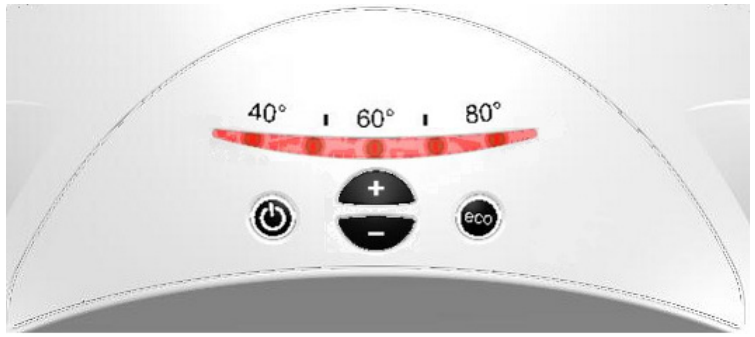
NTCs are called

- NTC1 and NTC2
- NTC_H and NTC_L

Device identification

“The highest LED”

(?)

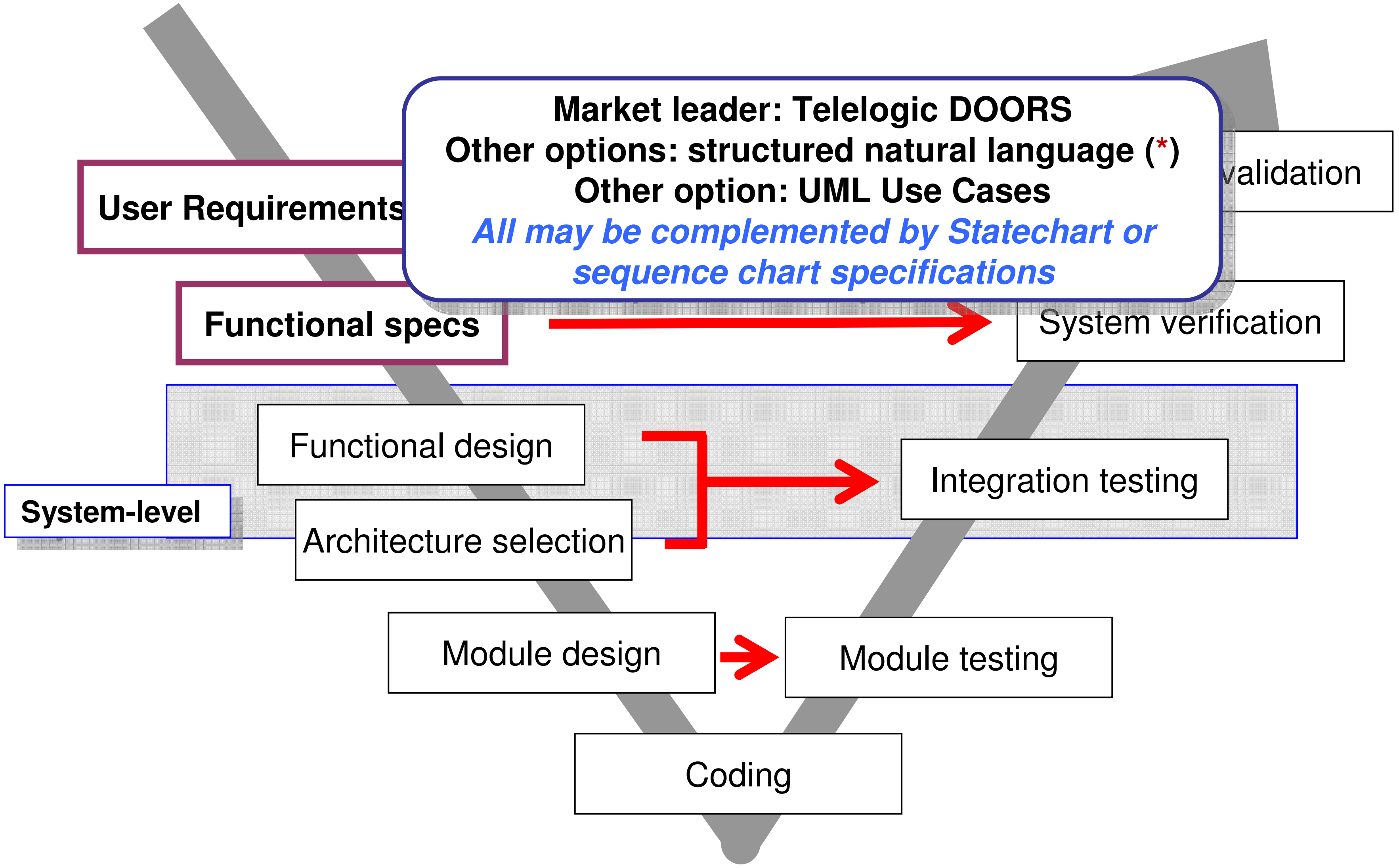


Multiple names (I/O)

How many temperatures are there?

Desired temperature
Working temperature
Setpoint temperature
“Thermostating” temperature
Boiler temperature
Boiler internal temperature
NTC temperature
Current temperature
Measured temperature
… (*more*)

Requirements Analysis



Market leader: Telelogic DOORS
Other options: structured natural language (*)
Other option: UML Use Cases
All may be complemented by Statechart or sequence chart specifications

Requirements written in structured natural language

- User Requirements
  - a set of high level requirements as perceived by the final user of the system used to drive the feature design.
    - Interactions with the end user
- Functional requirements
  - a more detailed set of requirements for the <u>mechanisms</u> and <u>procedures</u> needed to achieve the user requirements.
    - Typically a technical description possibly including a description of the evolution of the states of the system as well as scenarios of interactions with the system, including sequences of activities

- Consist of:
  - Document templates (General structure and formatting rules)
  - Rules for writing and constraining the natural language
  - Tracking, management and versioning

Requirements in natural language: Templates

User Requirements – high level requirements used to drive the feature design.

- Validation cases: a validation test should be defined for each user requirement

Functional Requirements – detailed set of requirements for the <u>mechanisms</u> and <u>procedures</u> needed to achieve the user requirements.

- Verification case: a verification test must be written for each requirement

Safety Requirements – a set of requirements that address the possible failure conditions and hazards. This can only be achieved by making assumptions about the environment.

Diagnostic Requirements – a set of requirements that address the on-board and off-board diagnostics required for this feature.

Para-Functional Requirements – a set of requirements that address feature characteristics that can be measured like timing, usability. scalability, speed,

State Chart Diagram – depicts the required behavior of the feature with the states, triggers, conditions and transitions *(base per la definizione del comportamento del sistema per raffinamento)*

Functional Context Diagram – listing all of the external I/O signals in & out of this feature. The input signals on the left side of the drawing and the output signals on the right side of the drawing. La descrizione deve essere puramente funzionale!. *(base per la definizione dell'architettura di sistema e la scomposizione in sottosistemi)*

Sequence diagrams, scenarios – list the sequence of actions/events identified for several typical working cases of the system *(base per la definizione dei casi di test)*

Requirements in natural language: Writing rules

1. Requirements shall be unique and identifiable
   - Itemized and identified by a label
2. Requirements shall be testable or verifiable.
   - A test case for each requirement
3. Requirement shall be brief and clear and use one of these verbs:
   - **shall** A mandatory requirement for the application, subsystem, process or organization being described.
   - **must** A requirement for some other application, subsystem, process or organization, which is treated as an *assumption* for our application, subsystem, process or organization.
   - **will** Behavior which is guaranteed to happen based on the immutable laws of physics.
   - **should** A desired, but not required, characteristic of our application, subsystem, process or organization.
4. Negative requirements shall not be written.
   - They are difficult to understand (sometimes double negations!), difficult to test, negative forms are typical of safety requirements
5. Requirements shall only stated for outbound state transitions.
   - Risk of writing them down twice, with redundancy and inconsistency upon change

Natural Language Requirements: Key Concepts

Contract-based definitions

System functioning may be considered as a contract between the system (subsystem) and the environment (other subsystems)

Assumptions/Assertions

- The contract consists of a set of “assumptions” on the environment or the other subsystems (what the “users” of the subsystem promise to be or to behave)
- If the environment and/or the subsystems satisfy the assumptions, the system (subsystem) under specification will have the duty to keep its side of the contract, that is, a set of assertions (what it promises to provide/how it promises to behave)
- Assumptions and assertions can be specified formally or informally using several languages.

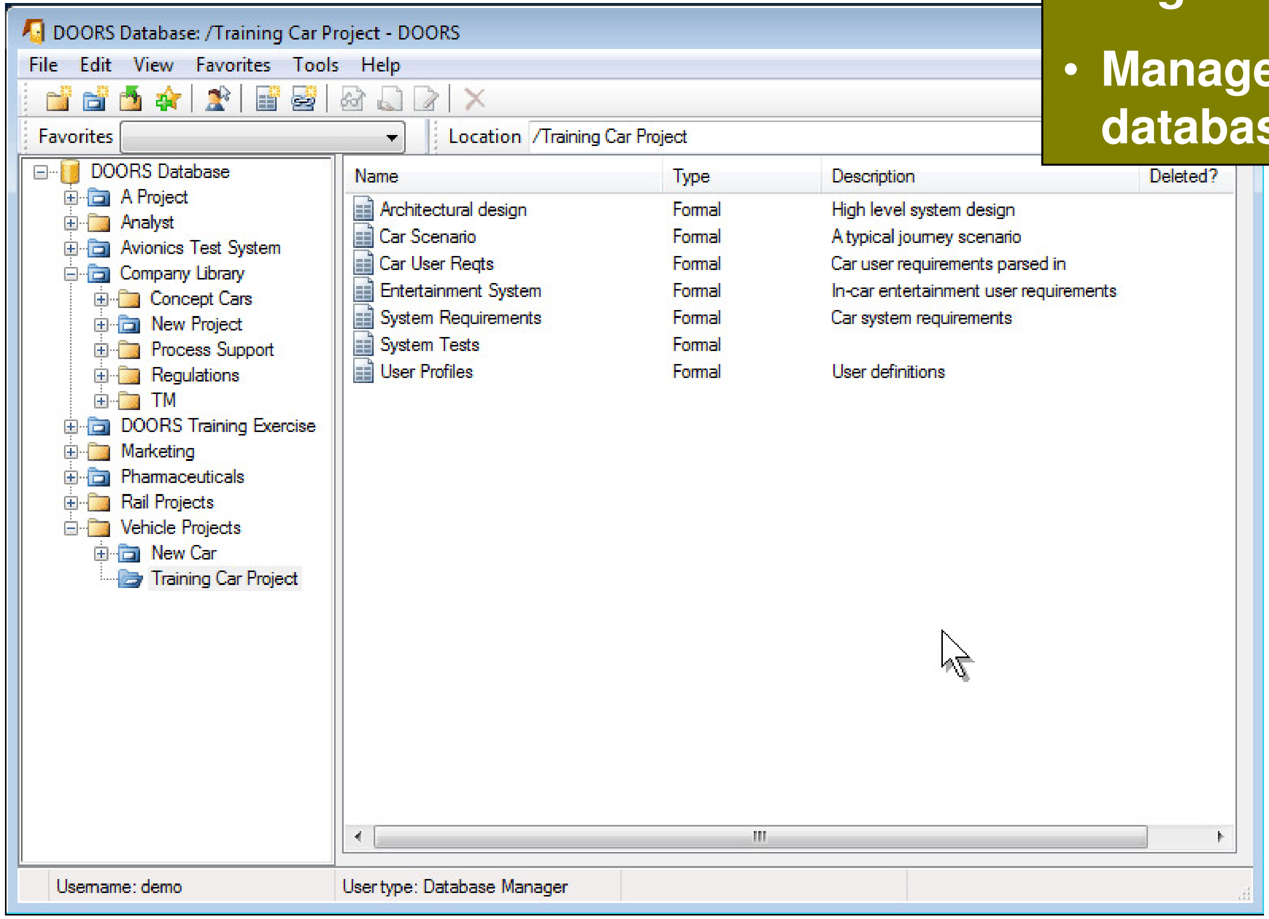
Natural Language Requirements: Key Concepts

Preconditions/Postconditions/Invariants

- The preconditions are the set of conditions that must be verified in advance for the reaction(s) described in the requirement to take place
- The postconditions are the set of conditions that are guaranteed to be true after the reaction(s) described in the requirement take place
- The set of invariants are the conditions that are guaranteed to be always true before or after the reaction (not necessarily during each atomic reaction)

Telelogic DOORS

- **Organizes hierarchies**
- **Manages requirements database**

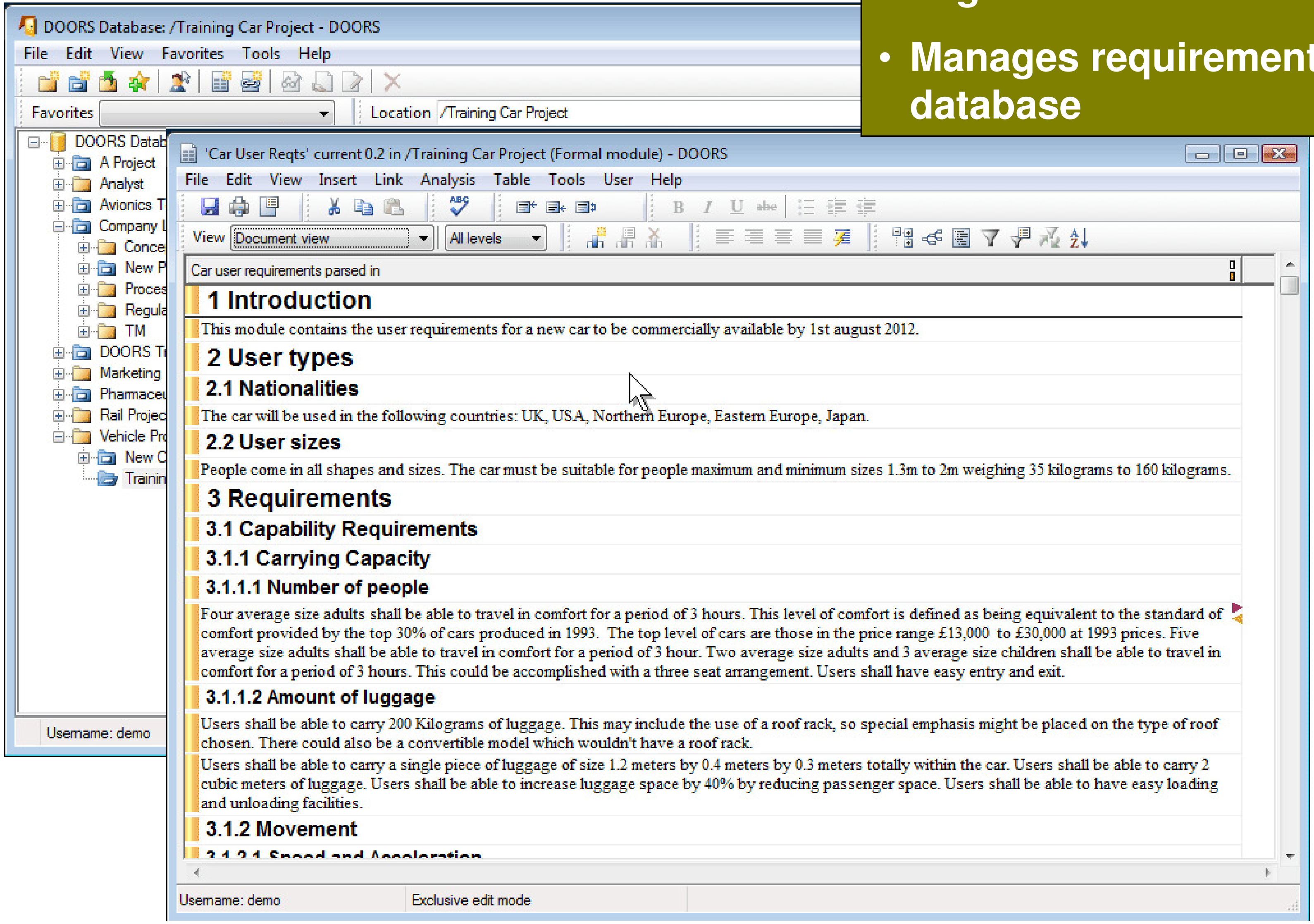

DOORS Database: /Training Car Project - DOORS

Name	Type	Description	Deleted?
Architectural design	Formal	High level system design	
Car Scenario	Formal	A typical journey scenario	
Car User Reqts	Formal	Car user requirements parsed in	
Entertainment System	Formal	In-car entertainment user requirements	
System Requirements	Formal	Car system requirements	
System Tests	Formal		
User Profiles	Formal	User definitions	

Username: demo
User type: Database Manager

Telelogic DOORS

- Organizes hierarchies
- Manages requirements database



DOORS Database: /Training Car Project - DOORS
File Edit View Favorites Tools Help
Favorites
Location /Training Car Project
DOORS Datab
A Project
Analyst
Avionics T
Company I
Conce
New P
Proces
Regula
TM
DOORS T
Marketing
Pharmace
Rail Projec
Vehicle Pro
New C
Trainin
Username: demo
'Car User Reqts' current 0.2 in /Training Car Project (Formal module) - DOORS
File Edit View Insert Link Analysis Table Tools User Help
View Document view
All levels
Car user requirements parsed in
Username: demo
Exclusive edit mode

1 Introduction

This module contains the user requirements for a new car to be commercially available by 1st august 2012.

2 User types

2.1 Nationalities

The car will be used in the following countries: UK, USA, Northern Europe, Eastern Europe, Japan.

2.2 User sizes

People come in all shapes and sizes. The car must be suitable for people maximum and minimum sizes 1.3m to 2m weighing 35 kilograms to 160 kilograms.

3 Requirements

3.1 Capability Requirements

3.1.1 Carrying Capacity

3.1.1.1 Number of people

Four average size adults shall be able to travel in comfort for a period of 3 hours. This level of comfort is defined as being equivalent to the standard of comfort provided by the top 30% of cars produced in 1993. The top level of cars are those in the price range £13,000 to £30,000 at 1993 prices. Five average size adults shall be able to travel in comfort for a period of 3 hour. Two average size adults and 3 average size children shall be able to travel in comfort for a period of 3 hours. This could be accomplished with a three seat arrangement. Users shall have easy entry and exit.

3.1.1.2 Amount of luggage

Users shall be able to carry 200 Kilograms of luggage. This may include the use of a roof rack, so special emphasis might be placed on the type of roof chosen. There could also be a convertible model which wouldn't have a roof rack.

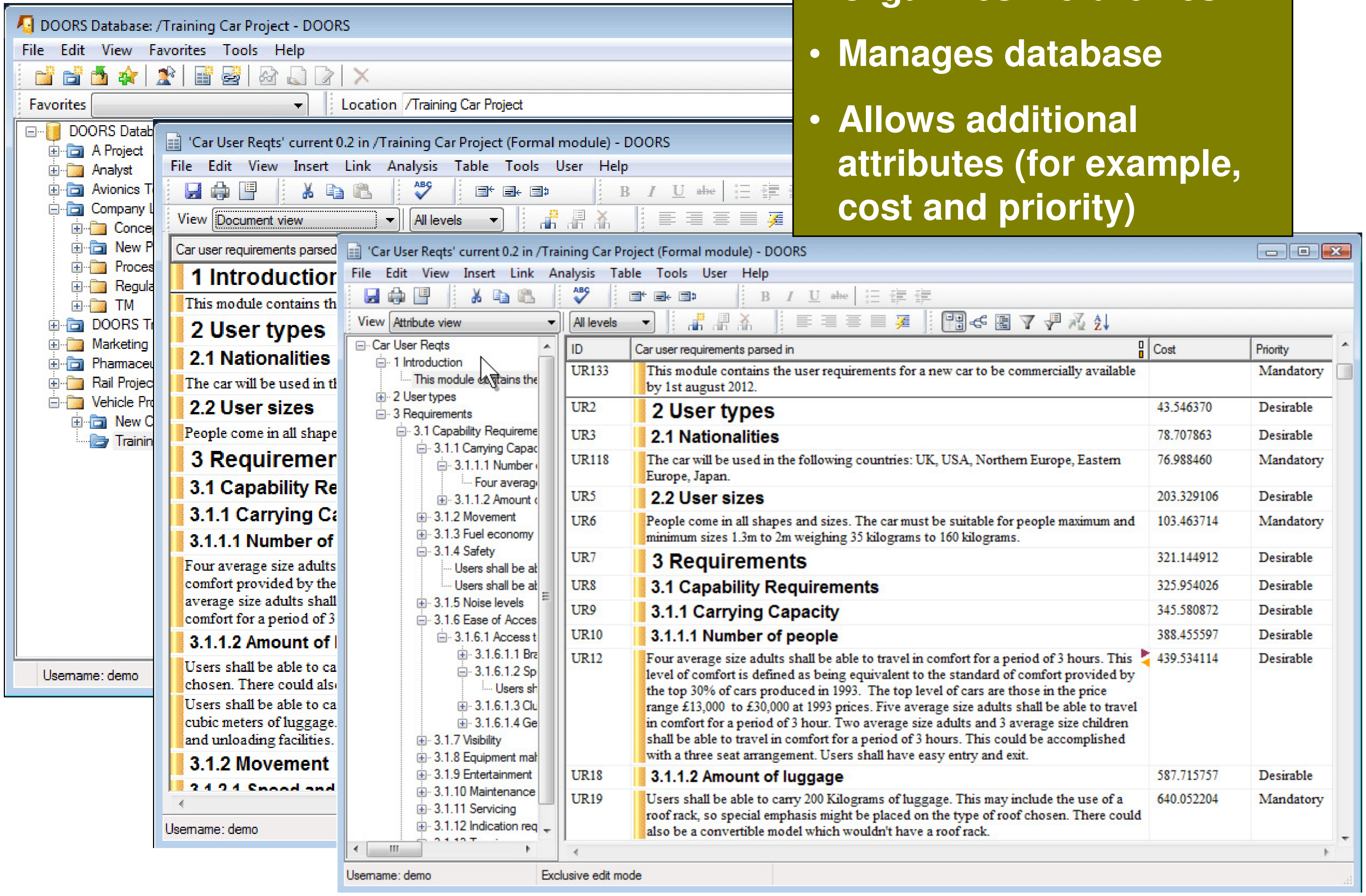
Users shall be able to carry a single piece of luggage of size 1.2 meters by 0.4 meters by 0.3 meters totally within the car. Users shall be able to carry 2 cubic meters of luggage. Users shall be able to increase luggage space by 40% by reducing passenger space. Users shall be able to have easy loading and unloading facilities.

3.1.2 Movement

3.1.2.1 Speed and Acceleration

Telelogic DOORS

- Organizes hierarchies
- Manages database
- Allows additional attributes (for example, cost and priority)

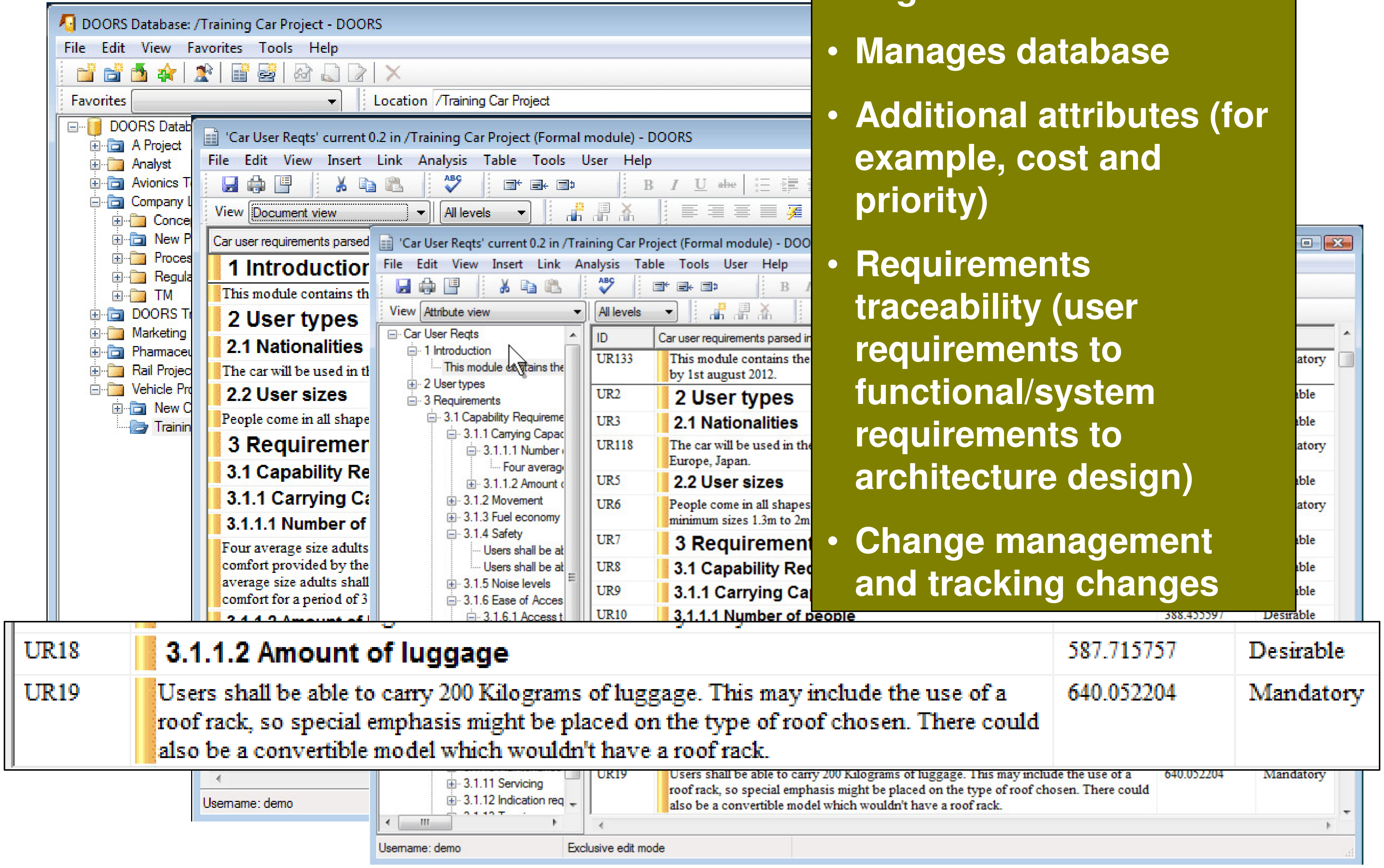

DOORS Database: /Training Car Project - DOORS
File Edit View Favorites Tools Help
Favorites
Location /Training Car Project
'Car User Reqts' current 0.2 in /Training Car Project (Formal module) - DOORS
File Edit View Insert Link Analysis Table Tools User Help
View Document view
All levels
View Attribute view
Username: demo
Exclusive edit mode

Car User Reqts
- 1 Introduction
  - This module contains the
- 2 User types
- 3 Requirements
  - 3.1 Capability Requireme
    - 3.1.1 Carrying Capac
      - 3.1.1.1 Number
        - Four averag
      - 3.1.1.2 Amount
    - 3.1.2 Movement
    - 3.1.3 Fuel economy
    - 3.1.4 Safety
      - Users shall be ab
      - Users shall be ab
    - 3.1.5 Noise levels
    - 3.1.6 Ease of Acces
      - 3.1.6.1 Access t
        - 3.1.6.1.1 Bra
        - 3.1.6.1.2 Sp
          - Users sh
        - 3.1.6.1.3 Clu
        - 3.1.6.1.4 Ge
    - 3.1.7 Visibility
    - 3.1.8 Equipment mai
    - 3.1.9 Entertainment
    - 3.1.10 Maintenance
    - 3.1.11 Servicing
    - 3.1.12 Indication req

ID	Car user requirements parsed in	Cost	Priority
UR133	This module contains the user requirements for a new car to be commercially available by 1st august 2012.		Mandatory
UR2	2 User types	43.546370	Desirable
UR3	2.1 Nationalities	78.707863	Desirable
UR118	The car will be used in the following countries: UK, USA, Northern Europe, Eastern Europe, Japan.	76.988460	Mandatory
UR5	2.2 User sizes	203.329106	Desirable
UR6	People come in all shapes and sizes. The car must be suitable for people maximum and minimum sizes 1.3m to 2m weighing 35 kilograms to 160 kilograms.	103.463714	Mandatory
UR7	3 Requirements	321.144912	Desirable
UR8	3.1 Capability Requirements	325.954026	Desirable
UR9	3.1.1 Carrying Capacity	345.580872	Desirable
UR10	3.1.1.1 Number of people	388.455597	Desirable
UR12	Four average size adults shall be able to travel in comfort for a period of 3 hours. This level of comfort is defined as being equivalent to the standard of comfort provided by the top 30% of cars produced in 1993. The top level of cars are those in the price range £13,000 to £30,000 at 1993 prices. Five average size adults shall be able to travel in comfort for a period of 3 hour. Two average size adults and 3 average size children shall be able to travel in comfort for a period of 3 hours. This could be accomplished with a three seat arrangement. Users shall have easy entry and exit.	439.534114	Desirable
UR18	3.1.1.2 Amount of luggage	587.715757	Desirable
UR19	Users shall be able to carry 200 Kilograms of luggage. This may include the use of a roof rack, so special emphasis might be placed on the type of roof chosen. There could also be a convertible model which wouldn't have a roof rack.	640.052204	Mandatory

Telelogic DOORS

- Organizes hierarchies
- Manages database
- Additional attributes (for example, cost and priority)
- Requirements traceability (user requirements to functional/system requirements to architecture design)
- Change management and tracking changes



1 Introduction
This module contains th
2 User types
2.1 Nationalities
The car will be used in th
2.2 User sizes
People come in all shape
3 Requiremer
3.1 Capability Re
3.1.1 Carrying Ca
3.1.1.1 Number of
Four average size adults
comfort provided by the
average size adults shall
comfort for a period of 3

ID	Car user requirements parsed i		
UR133	This module contains the by 1st august 2012.		atory
UR2	2 User types		ble
UR3	2.1 Nationalities		ble
UR118	The car will be used in the Europe, Japan.		atory
UR5	2.2 User sizes		ble
UR6	People come in all shapes minimum sizes 1.3m to 2m		atory
UR7	3 Requirement		ble
UR8	3.1 Capability Req		ble
UR9	3.1.1 Carrying Ca		ble
UR10	3.1.1.1 Number of people	388.455597	Desirable
UR18	3.1.1.2 Amount of luggage	587.715757	Desirable
UR19	Users shall be able to carry 200 Kilograms of luggage. This may include the use of a roof rack, so special emphasis might be placed on the type of roof chosen. There could also be a convertible model which wouldn't have a roof rack.	640.052204	Mandatory

Requirements analysis: A proposed approach

- As a result of the analysis, user requirements are identified, first informally (as a structured text document) then, more formally, using diagrams.
- Requirements are then expressed as items
- Any requirement referring a reaction of the system may be further refined highlighting (possibly explicitly) the assumptions for the requirement or the preconditions that apply and the postconditions that are guaranteed after thje reaction described by the requirement.
- For each reaction, the definition of the preconditions helps defining the initial state of the system to which the requirement applies and the condiotions on events on input data.

Requirements analysis: A proposed approach

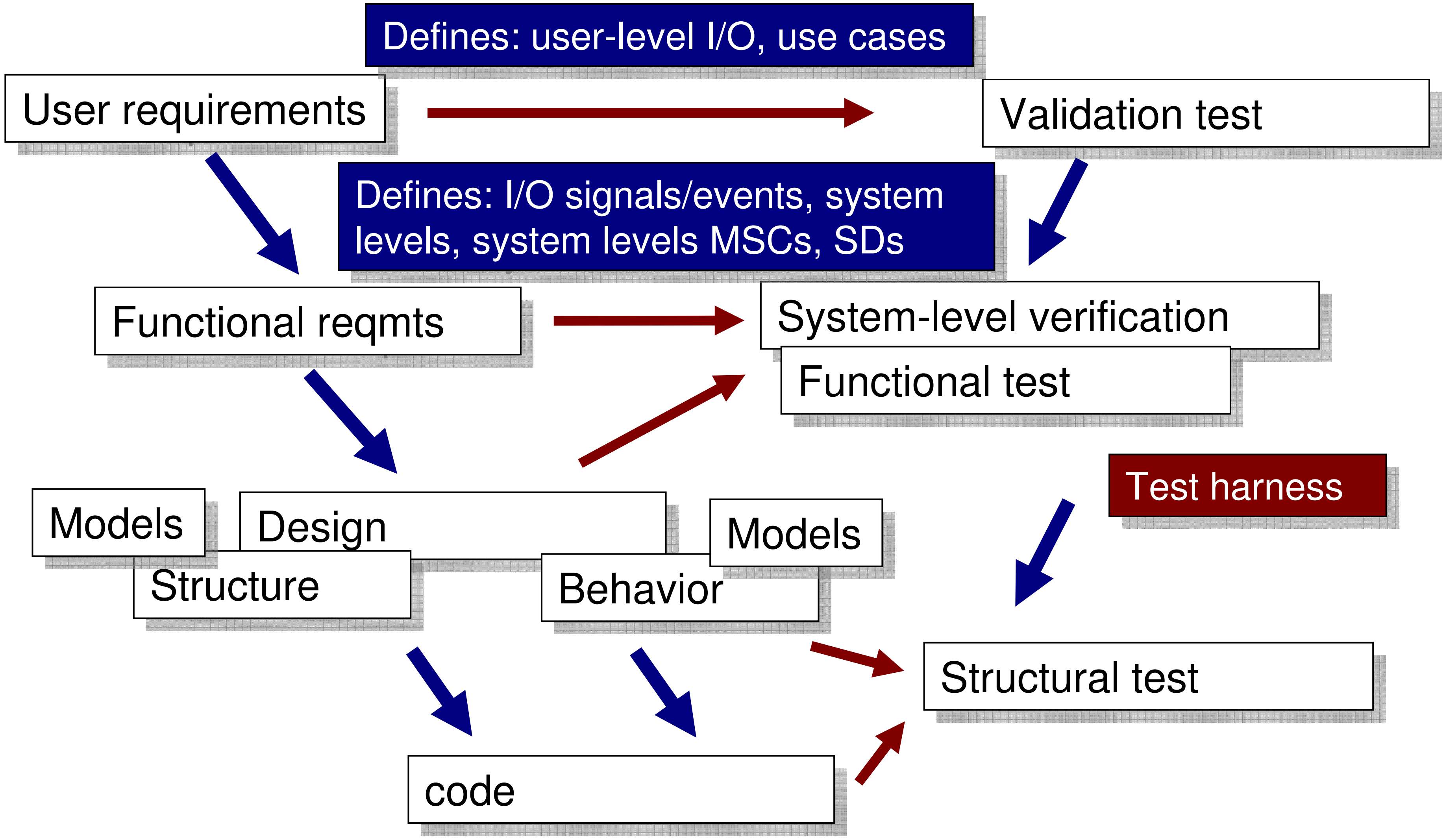
- In the definition of each requirement, it is always useful to ask questions related to:
  - Possible error conditions
  - Special cases/exceptions that apply to the requirement
  - What happens when one or more of the conditions for the requirement (or their combination) are not verified.
- To ensure that the corresponding cases are covered by other requirements (if of interest) and avoid omissions and incomplete requirements.

Requirements analysis: An example

Temperature setup

- When the boiler is in thermostat mode, it is possible to setup the operating temperature around which it must be maintained and it is possible to show the temperature actually measured by the sensors within the boiler. To set the temperature you need to activate the setting mode. We enter the setting mode by pressing the + or – buttons (C and D). All LED indicators of temperature (1, 2, 3, 4, 5) go out, except for one that is on, indicating the current set point. The pressure of the + and - buttons allows to temporarily change the set point. If no key is pressed for 5 seconds, the last set point becomes the temporary set point and the device returns in thermostat mode. In setup mode, the button Eco (B) is ignored. Instead, the ON button (A) pressed in setup mode leads to the selection of the latest temporary set point and to the shutdown of the boiler. If in setup mode the highest LED is on, associated with the highest set point, and the + key is pressed, the lowest LED is lit. Conversely, from the lowest position, pressing the button - leads to the highest position. The temperature setpoint can be selected from the interface only when the boiler is in thermostat mode and cannot be set if the eco mode is active.

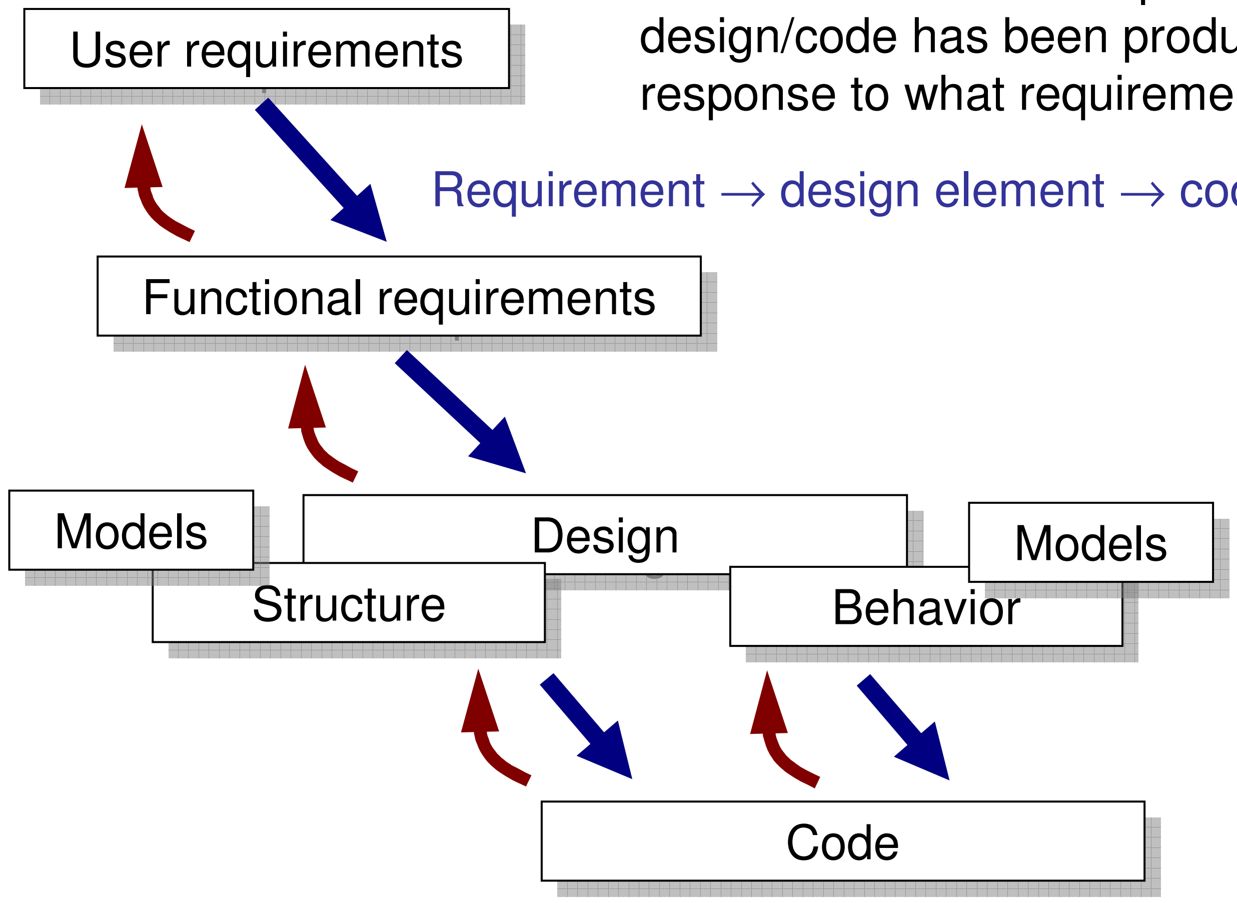
Requirements refinement and testing



Requirements tracking

We need to know what part of the design/code has been produced in response to what requirement

Requirement → design element → code section



Code section → design element → requirement

And what requirement required the design element or the code section

Requirements tracking

Launched

3 Jan 1999

Mission

Land near South Pole

Dig for water ice with a robotic arm

Fate

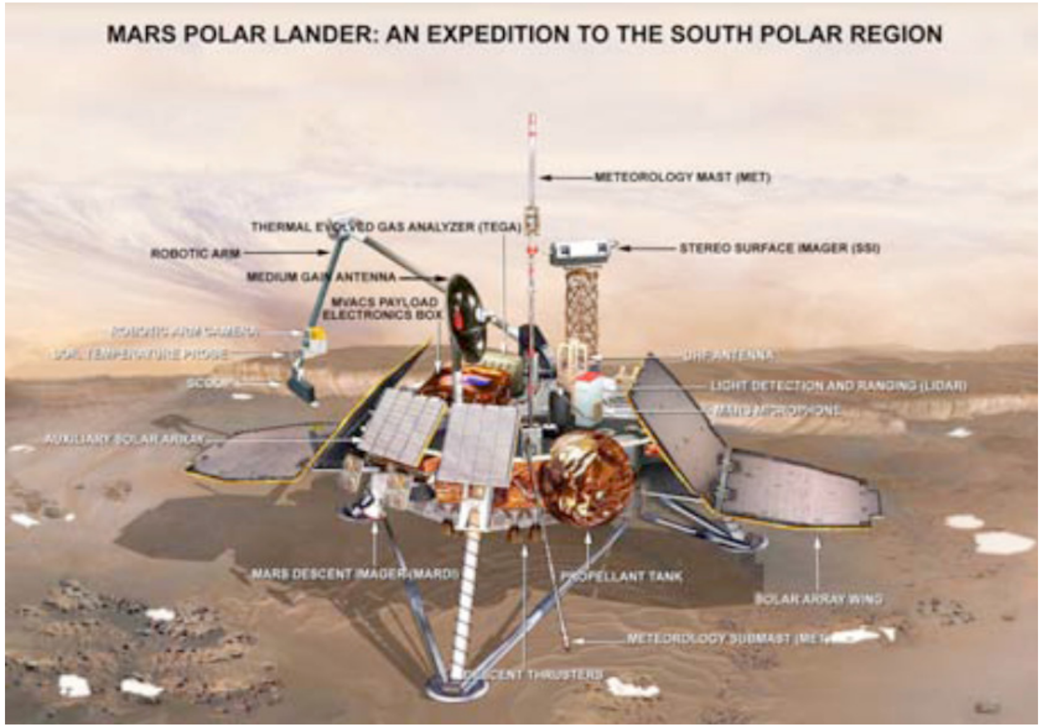
Arrived 3 Dec 1999

No signal received after initial phase of descent

Cause

Several candidate causes

Most likely is premature engine shutdown due to noise on leg sensors


MARS POLAR LANDER: AN EXPEDITION TO THE SOUTH POLAR REGION
METEOROLOGY MAST (MET)
THERMAL EVOLVED GAS ANALYZER (TEGA)
ROBOTIC ARM
STEREO SURFACE IMAGER (SSI)
MEDIUM GAIN ANTENNA
MVACS PAYLOAD ELECTRONICS BOX
ROBOTIC ARM CAMERA
SOIL TEMPERATURE PROBE
UHF ANTENNA
SCOOP
LIGHT DETECTION AND RANGING (LIDAR)
MARS MICROPHONE
AUXILIARY SOLAR ARRAY
MARS DESCENT IMAGER (MARDI)
PROPELLANT TANK
SOLAR ARRAY WING
METEOROLOGY SUBMAST (MET)
DESCENT THRUSTERS

Adapted from the "Report of the Loss of the Mars Polar Lander and Deep Space 2 Missions -- JPL Special Review Board (Casani Report) - March 2000".

See http://www.nasa.gov/newsinfo/marsreports.html

The Polar Lander Case: what happened?

We don’t know for sure: spacecraft not designed to send telemetry during descent (decision severely criticized by review boards)

Possible causes:

Lander failed to separate from cruise stage (plausible but unlikely)

Landing site too steep (plausible)

Heatshield failed (plausible)

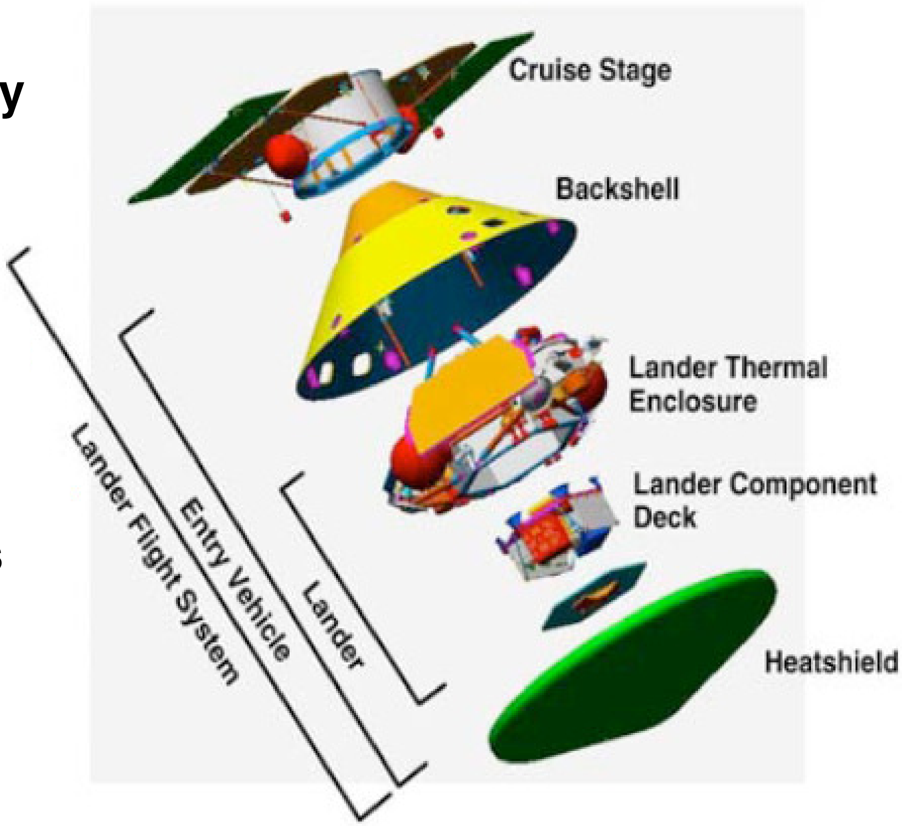
Loss of control due to dynamic effects (plausible)

Loss of control due to center-of-mass shift (plausible)

Premature Shutdown of Descent Engines (most likely!)

Parachute drapes over lander (plausible)

Backshell hits lander (plausible but unlikely)



The Polar Lander Case: Premature shutdown

Cause of error

- **Magnetic sensor on each leg senses touchdown**
- **Legs unfold at 1500m above surface**
- **software accepts transient signals on touchdown sensors during unfolding**

Factors

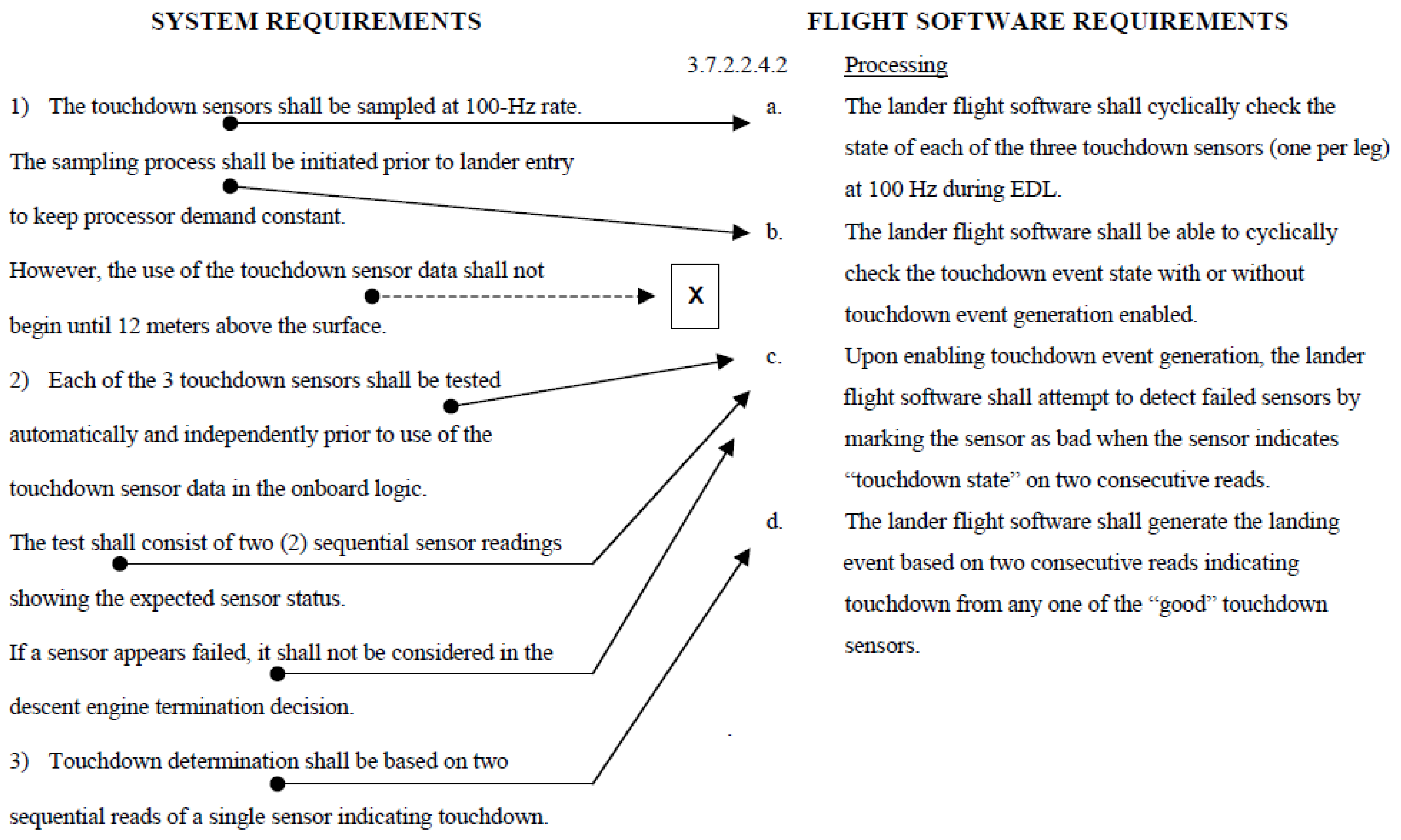
- **System requirement to ignore the transient signals**
- **the software requirements did not describe the effect**
- **Engineers present at code inspection didn’t understand the effect**
- **Not caught in testing because:**
- **Unit testing didn’t include the transients**
- **Sensors improperly wired during integration tests (no touchdown detected!)**

Result of error

- **Engines shut down before spacecraft has landed**
- **estimated at 40m above surface, travelling at 13 m/s**
- **estimated impact velocity 22m/s (spacecraft would not survive this)**
- **nominal touchdown velocity 2.4m/s**

Requirements Analysis

Lack of tracking / No testing



SYSTEM REQUIREMENTS

1) The touchdown sensors shall be sampled at 100-Hz rate.

The sampling process shall be initiated prior to lander entry to keep processor demand constant.

However, the use of the touchdown sensor data shall not begin until 12 meters above the surface.

2) Each of the 3 touchdown sensors shall be tested automatically and independently prior to use of the touchdown sensor data in the onboard logic.

The test shall consist of two (2) sequential sensor readings showing the expected sensor status.

If a sensor appears failed, it shall not be considered in the descent engine termination decision.

3) Touchdown determination shall be based on two sequential reads of a single sensor indicating touchdown.

FLIGHT SOFTWARE REQUIREMENTS

3.7.2.2.4.2 Processing

a. The lander flight software shall cyclically check the state of each of the three touchdown sensors (one per leg) at 100 Hz during EDL.

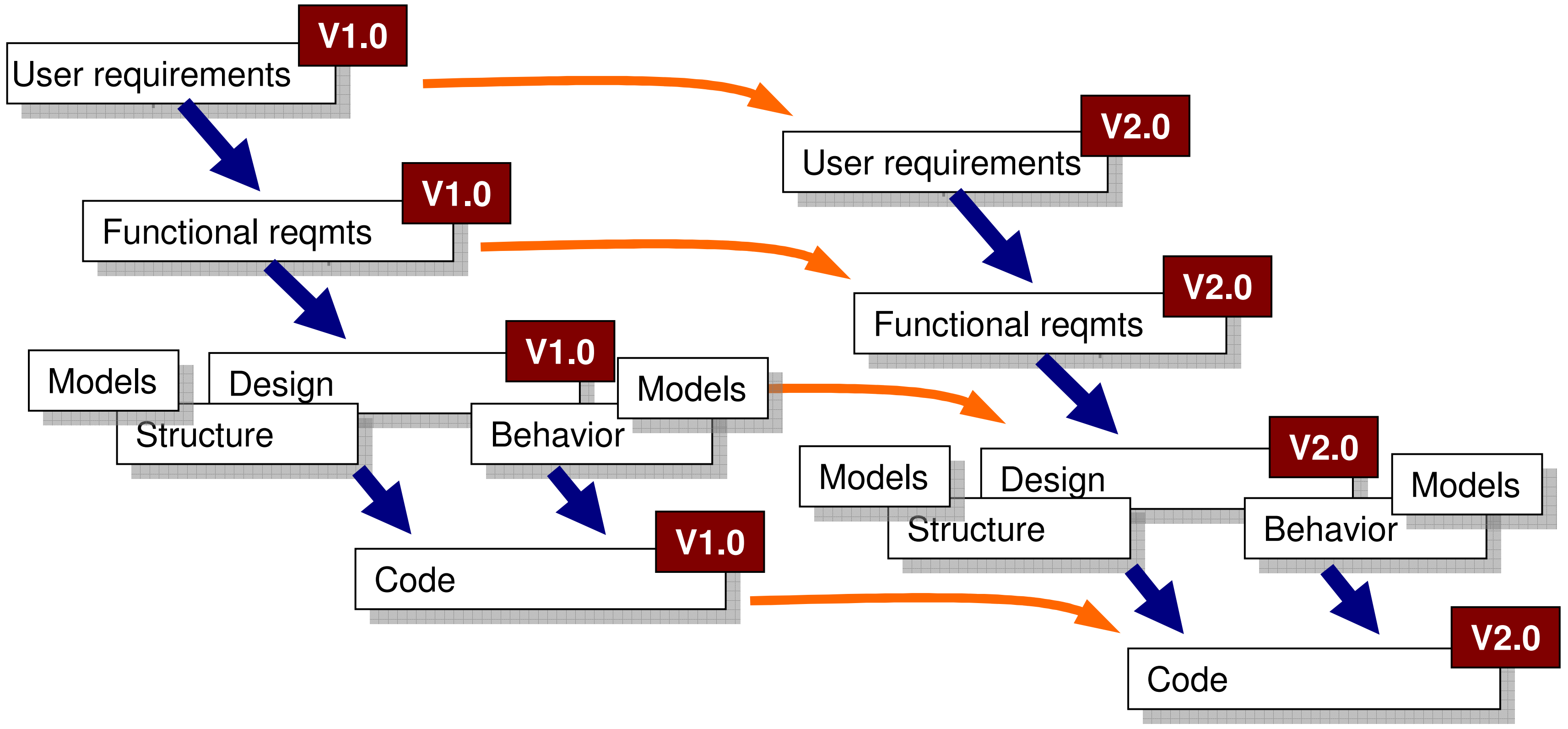
b. The lander flight software shall be able to cyclically check the touchdown event state with or without touchdown event generation enabled.

c. Upon enabling touchdown event generation, the lander flight software shall attempt to detect failed sensors by marking the sensor as bad when the sensor indicates "touchdown state" on two consecutive reads.

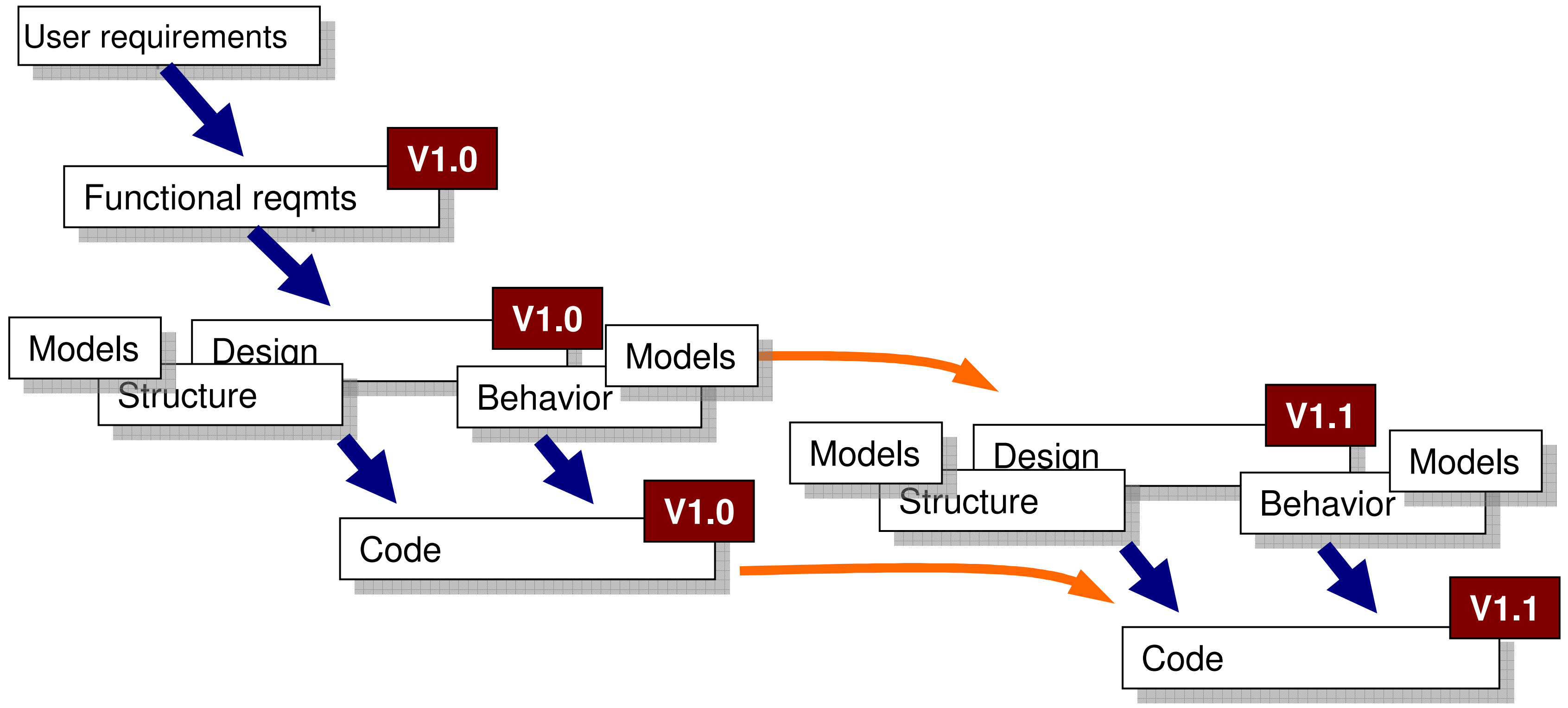
d. The lander flight software shall generate the landing event based on two consecutive reads indicating touchdown from any one of the "good" touchdown sensors.

Figure 7-9. MPL System Requirements Mapping to Flight Software Requirements

Requirements tracking: Change management

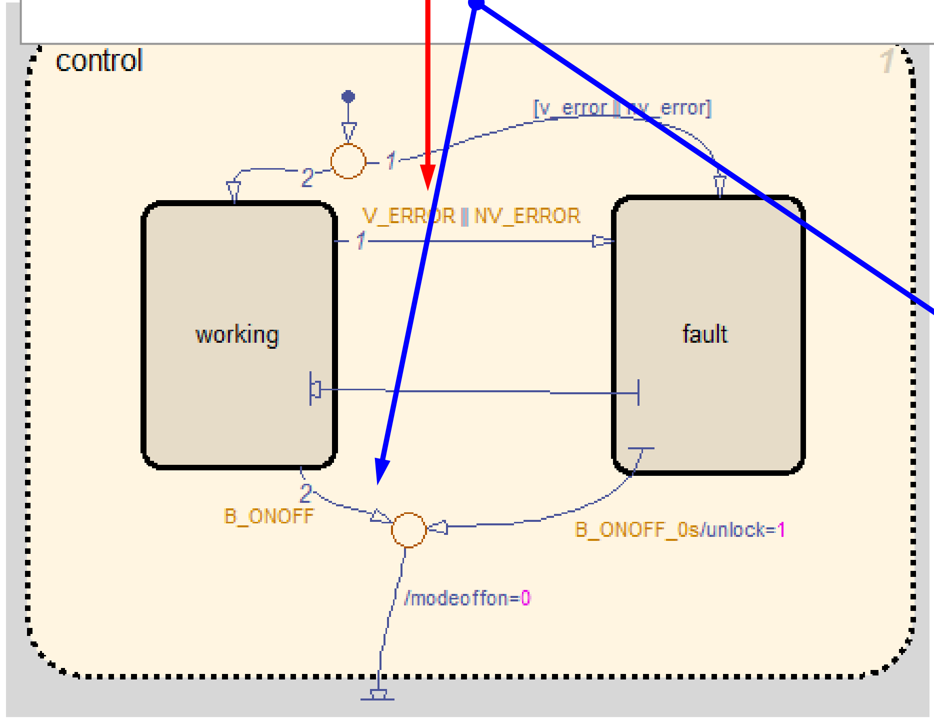


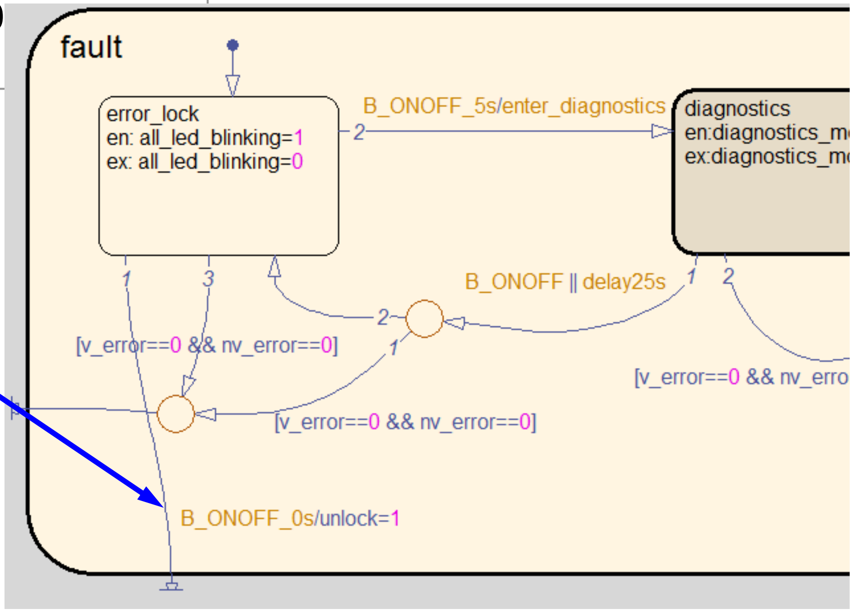
Requirements tracking: Change management



An example (tracking requirements to design)

- Working mode
  - ***Entering mode***
  - ***While in mode***
    - if the active mode is not diagnostic and an error signal is input from the controller with the signal of volatile error VolatileError being not zero or with the signal of non volatile error NonVolatileError being not zero, then the interface should enter the Error mode.
  - ***Exiting mode***
    - If the active mode is not diagnostic, the pressure of the button B_OFFON should bring the system in mode Standby. The operation should be confirmed to the controller by setting the signal modeOffOn to 0





Requirements tracking

Requirements change

THE PROJECT REQUIREMENTS ARE FORMING IN MY MIND.
NOW THEY'RE CHANGING... CHANGING... CHANGING... CHANGING... OKAY. NO, WAIT... CHANGING... CHANGING... DONE.
NATURALLY, I WON'T BE SHARING ANY OF THESE THOUGHTS WITH ENGINEERING.
I BUDGETED FOR SOME GOONS TO BEAT IT OUT OF YOU.
S. ADAMS E-MAIL: SCOTTADAMS@AOL.COM
© 1994 United Feature Syndicate, Inc.
9-22

Wrong reuse of specifications

- Ariane 5 is a European expendable launch system designed to deliver payloads into geostationary transfer orbit or low Earth orbit. It succeeded Ariane 4, but does not derive from it directly. Its development took 10 years and cost $7 billion.
- Ariane 5's first test flight (Ariane 5 Flight 501) on 4 June 1996 failed, with the rocket self-destructing 37 seconds after launch because of a malfunction in the control software, which was arguably one of the most expensive computer bugs in history. A data conversion from 64-bit floating point to 16-bit signed integer value had caused a processor trap (operand error). The floating point number had a value too large to be represented by a 16-bit signed integer.
- ***The Ariane 5 software reused the specifications from the Ariane 4, but the Ariane 5's flight path was considerably different and beyond the range for which the reused code had been designed.*** Specifically, the Ariane 5's greater acceleration caused the back-up and primary inertial guidance computers to crash, after which the launcher's nozzles were directed by spurious data. Pre-flight tests had never been performed on the re-alignment code under simulated Ariane 5 flight conditions, so the error was not discovered before launch.

Itemized requirements

6. Temperature setup mode

6.1 Entering mode

6.1.1 The temporary setpoint temperature tTemporarySetpoint must be set to the current value of the temperature setpoint tUserSetpoint.

6.2 While in mode

6.2.1 All LEDs must be off except one corresponding to the value of tTemporarySetpoint

6.2.2 When pressing the button B_UP the value of tTemporarySetpoint must be set to the value immediately higher, when pressing B_DOWN to the level immediately lower. If the temperature is at the maximum level, pressing the button B_UP will move tTemporarySetpoint to the minimum value. From the lowest position, pressing the button B_DOWN brings the temperature tTemporarySetpoint to the maximum value.

6.3 Exit from mode

6.3.1 If no key is pressed for 5 seconds, tUserSetpoint assumes the current value of tTemporarySetpoint, the interface must exit from setup mode and go back to thermostat mode.

6.3.2 after pressing the button B_OFFON the interface exits from temperature Setup mode, the value of tUserSetpoint is set to the value of tTemporarySetpoint and the interface goes in Shutdown mode.

6.3.3 In seguito ad un distacco di alimentazione e successivo ripristino il valore di tUserSetpoint rimane quello precedente al distacco.

Itemized requirements

6. Temperature setup mode

    6.1 Entering mode

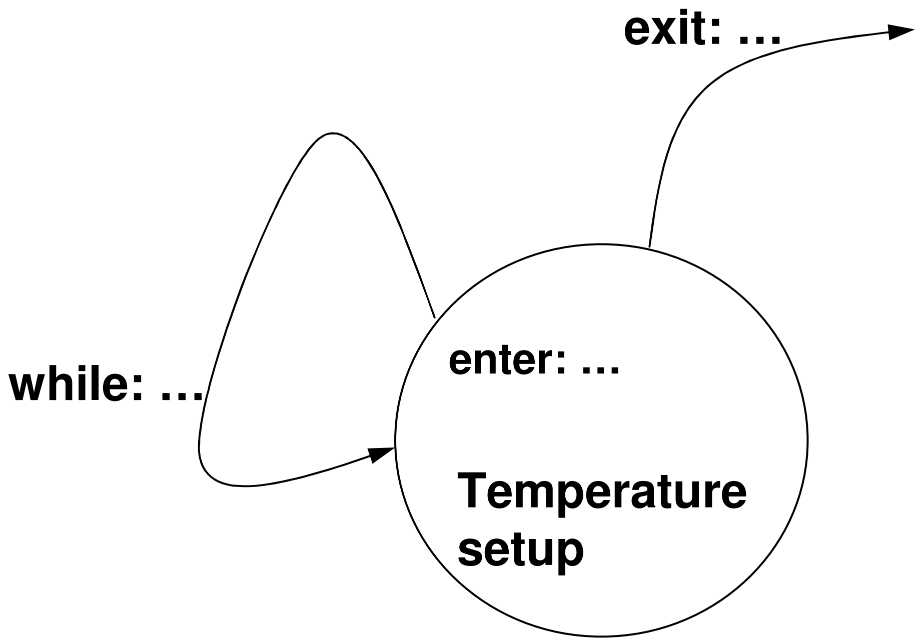
    …

    6.2 While in mode

    …

    6.3 Exit from mode

    …



Going formal

Regular Requirements

R.3.2.1 if the active mode is not diagnostic and an error signal is input from the controller with the signal of volatile error VolatileError being set or with the signal of non volatile error NonVolatileError being set, then the interface should enter the Error mode.

R.3.2.1 For all modes not Diagnostic, if VolatileError=true or NonVolatileError=true, then mode=Error **_English_**

R.3.2.1 $\neg$(mode(n)=‘Diagnostic’) $\wedge$ VolatileError(n) $\wedge$ NonVolatileError(n) $\rightarrow$ mode(n+1) = ‘Error’ **_Predicate logic_**

R.3.2.1 A(($\neg$(mode=‘Diagnostic’) $\wedge$ VolatileError $\wedge$ NonVolatileError) $\rightarrow$ X(mode = ‘Error’)) **_CTL_**

Going formal

Safety Requirements

S.1 if an error signal is input from the controller with the signal of volatile error VolatileError being set or with the signal of non volatile error NonVolatileError being set, then the resistor must be off.

S.1 For all modes if VolatileError=true or NonVolatileError=true, then ResistorOnOff=false

S.1 $(\mathrm{VolatileError}(n) \vee \mathrm{NonVolatileError}(n)) \rightarrow \neg \mathrm{ResistorOnOff}(n)$

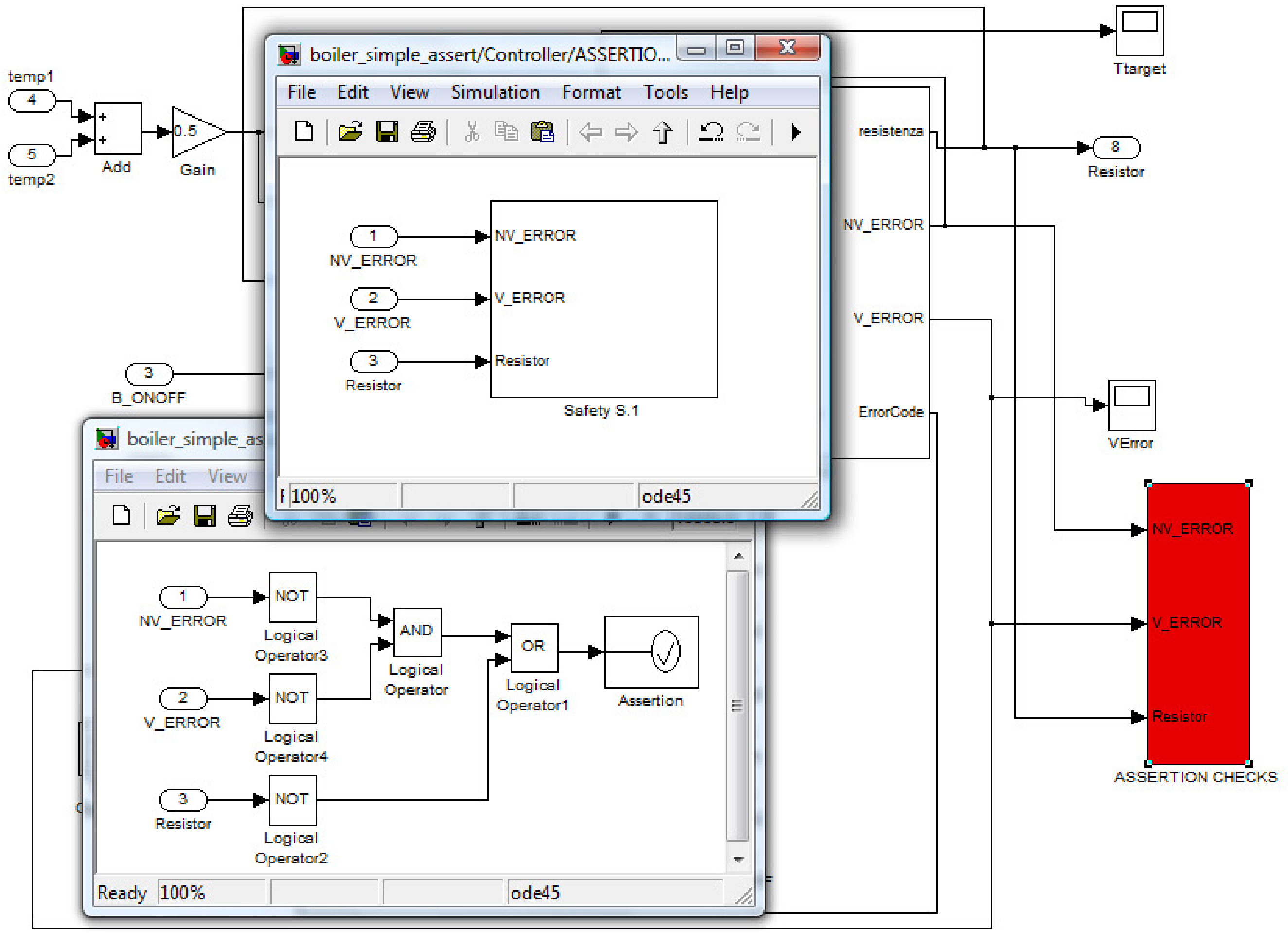
S.1 $A((\mathrm{VolatileError} \vee \mathrm{NonVolatileError}) \rightarrow \neg \mathrm{ResistorOnOff})$

S.1 $\neg(\mathrm{VolatileError} \vee \mathrm{NonVolatileError}) \vee \neg \mathrm{ResistorOnOff}$

S.1 $(\neg \mathrm{VolatileError} \wedge \neg \mathrm{NonVolatileError}) \vee \neg \mathrm{ResistorOnOff}$ **DNF**

Going formal

S.1 (¬VolatileError ∧ ¬NonVolatileError) ∨ ¬ResistorOnOff **DNF**



System-level testing

6. Temperature setup mode

6.1 Entering mode

- **Pre-condition:** The system is in thermostat mode.
- **Input sequence:** press the button B_UP.
- **Expected outputs: ---**
- **Post-condition:** verify that the system enters in setup mode and tTemporarySetpoint = tUserSetpoint.

- **Pre-condition:** The system is in thermostat mode.
- **Input sequence :** press the button B_DOWN.
- **Expected outputs: ---**
- **Post-condition:** verify that the system enters in setup mode and tTemporarySetpoint = tUserSetpoint.

6.2 While in mode

- **Pre-condition:** the system is in Temperature Setup mode
- **Input sequence :** ---
- **Expected output: ---**
- **Post-condition:** All LEDs are off except the one encoding tTemporarySetpoint.

- **Pre-condizione:** the system is in Temperature Setup mode, tTemporarySetPoint = L_T_MIN, LED_T_MIN is on.
- **Input sequence:** press button B_UP
- **Expected outputs:** LED_T_MIN is off, LED_T_LOW is on, tTemporarySetPoint = L_T_LOW.
- **Post-condition:** tTemporarySetpoint = L_T_LOW, LED_T_LOW is on.