SysML Modeling Of The Landing Gear System

Sami MADDOURI [*1], Shiva NEJATI [†1], and Lionel BRIAND [‡1]

[1]SNT Centre, University of Luxembourg, Luxembourg

1 Introduction

This report presents a public domain case study: **landing gear system** conducted by the Software Verification and Validation lab from the University of Luxembourg. The purpose of this study is to apply a SysML-based approach to support change impact analysis on the landing gear case study and to evaluate the proposed methodology. Change impact analysis is defined by [?] as "the determination of potential effects to a subject system resulting from a proposed software change". The context for change impact analysis is change control, which is part of software configuration management, and plays an important role in software development and maintenance [?]. Change impact analysis is a crucial software development activity due to the often massive amount of change a software system is exposed to during its life cycle [?]. Two aspects are integrated in change impact analysis: (1) the assessment of the consequences of altering the functionality/ capabilities of the software system, and (2) the identification of software artefacts that are affected by the change. According to case studies conducted in eleven different industry sectors, [?] observe that software engineers tended to perform impact analysis intuitively. But, many engineers do not predict the complete change impact analysis because of the additional semantic and time-sensitive dependencies involved in embedded software development. Thus, there is a huge gap between what engineers assume to be impacted and what is really impacted after a change. So, there is a need

[*]sami.maddouri@uni.lu

[†]shiva.nejati@uni.lu

[‡]lionel.briand@uni.lu

to automate change impact analysis in order to reduce the error margin of capturing impacted elements. Finally, DO-178B (Standard for safety of software used in airborne systems) in sections 7,8 and 12 recommends change impact analysis as a technic to support safety in airborne systems. This study is dealing with change impact analysis of critical embedded systems in the avionic domain. In this report, a SysML approach already defined is applied to describe and illustrate a public case study which is the landing gear system. After applying the proposed approach on the case study, an evaluation is elaborated in order to check the efficiency of the proposed methodology.

2 Landing gear system

According to the Federal Aviation Administration, the landing gear system is the undercarriage of an aircraft that supports the entire weight of this aircraft during landing and ground operations. The LGS is in charge of maneuvering landing gears and associated doors. The system is controlled digitally in nominal mode. In nominal mode, the landing sequence is: (1) open the doors, (2) extend the landing gears and (3) close the doors. The second operation is the retraction sequence which is proceeded: (1) open the doors, (2) retract the gears and (3) close the doors. Aircraft landing gear is one of the most critical systems in an aircraft which requires performing proper traceability links in order to support change impact analysis. As described in the figure 1, the Landing gear system is composed of: (1) a mechanical part which contains all the mechanical devices such as doors, gears, electro-valves and cylinders, (2) a digital part which represents the software part of the LGS and (3) a pilot interface.

3 The SysML-based Methodology

3.1 Specifying the context

Figure 2 shows a SysML Internal Block Definition capturing the context of the landing gear system. The purpose of the context diagram is to specify the boundary between the system under analysis and the external environment.

As shown in the figure 2, the diagram is composed of the landing gear system, which is the system under analysis and the following four external blocks directly interacting with LGS: the pilot, the

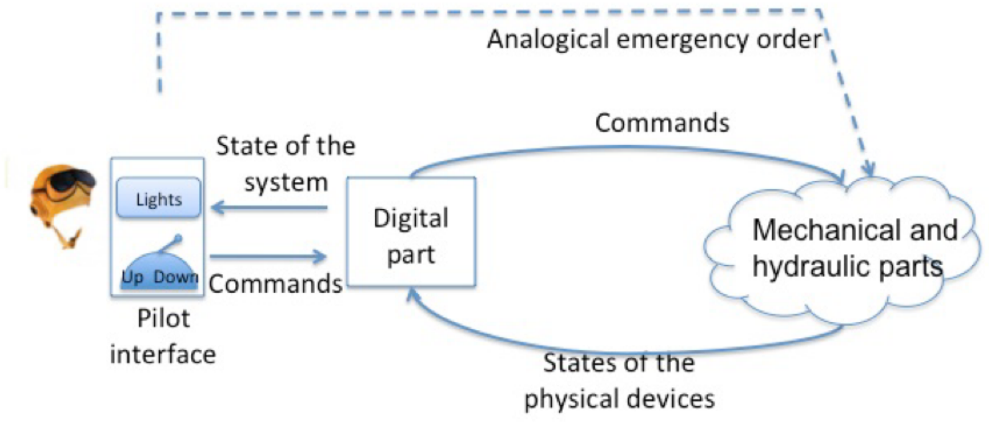

Analogical emergency order
Commands
State of the system
Lights
Up Down
Commands
Digital part
Mechanical and hydraulic parts
Pilot interface
States of the physical devices

Figure 1: The global architecture of the landing gear system

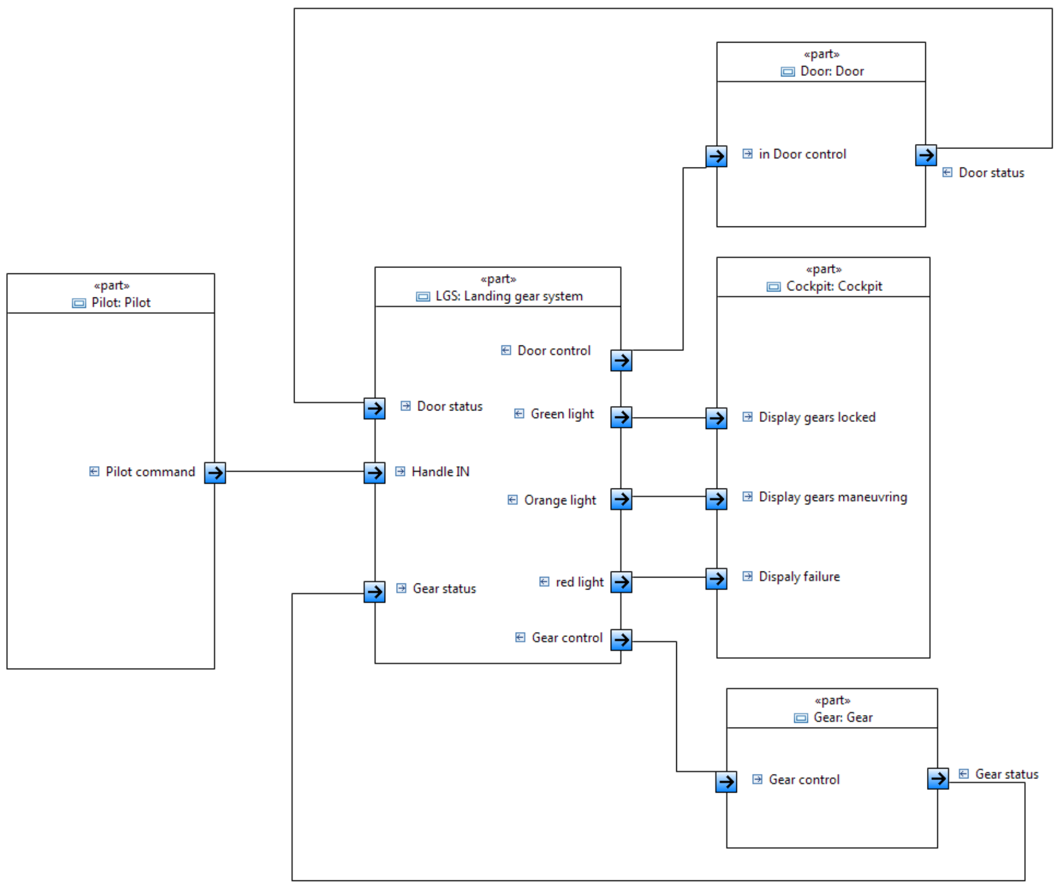


Figure 2: The context diagram for the landing gear system

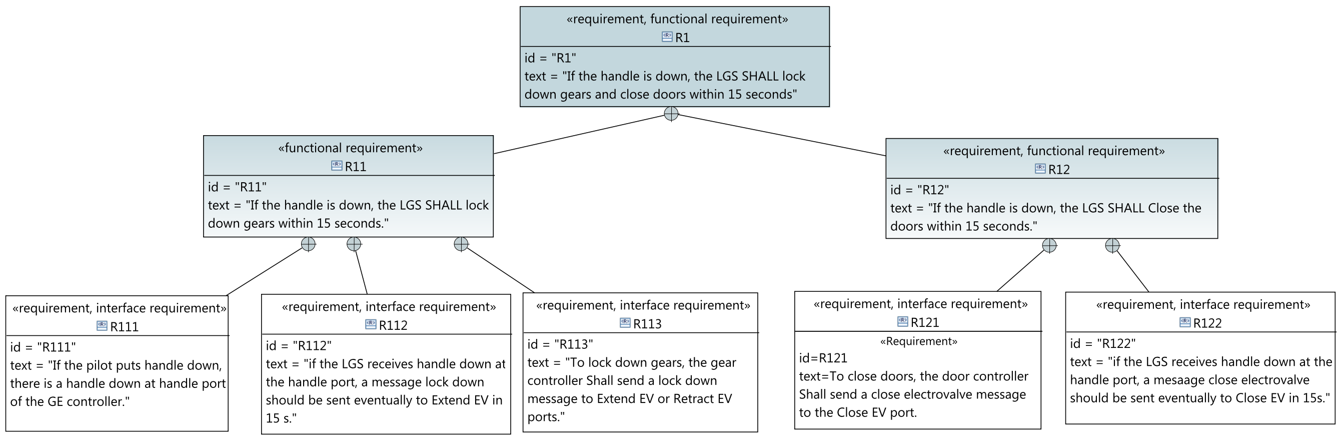
cockpit, the door and the gear. The data exchange between LGS and each of pilot, cockpit, door and gear is captured using ports and connectors. The connectors connect the ports of different blocks. For instance, the pilot commands the handle up or down in order to stimulate the LGS. Pilot command is the output port of the pilot block. It sends the command to Handle IN which is the input port of the LGS block. The LGS interacts with the cockpit through three connections. The LGS sends the different states of gears to the cockpit. The cockpit contains three lights: green, orange and red. For instance, if gears are locked down, the LGS will put the green light ON in the cockpit interface. Else if the gears are maneuvering, the LGS will put the orange light ON in the cockpit interface. Otherwise, the LGS will put the red light ON to say that there is anomaly in the landing gear system. The LGS controls both door and gears through door control and gear control. And gear and doors send their status to the LGS. The LGS represents the system under analysis as shown in figure 2. It will be decomposed further into a system architecture in the "Specify the architecture" section.

3.2 Specifying Requirements

Figure 3 shows a SysML requirement diagram capturing the specifications of the landing gear system. The purpose of requirements diagrams is to provide a mechanism to list the current requirements, to decompose the requirements into finer-grained requirements, to derive new requirements, to specify how the requirements are related, to differentiate between hardware and software requirements, to reduce the ambiguity of the requirements, and to validate the requirements with the customers.

Figure 3 represents the requirements related to the main functionality of the landing gear system: the outgoing sequence. Two types of requirements are shown on figure 3: functional requirement and interface requirement. The functional requirements (e.g, R1, R11 and R12) describe high level requirements related to the functional aspects of the landing gear system, and concern both software and hardware blocks.

R1 is decomposed into two sub functional requirements: R11 and R12 using a decompose link. R11 is decomposed into interface requirements: R111, R112 and R113 through a decompose traceability link. R12 is decomposed in his side in two interface requirements: R121 and R122 trough a decompose traceability link. Each requirement has an unique ID. Also, in order to reduce ambiguity of


«requirement, functional requirement»
R1
id = "R1"
text = "If the handle is down, the LGS SHALL lock down gears and close doors within 15 seconds"

«functional requirement»
R11
id = "R11"
text = "If the handle is down, the LGS SHALL lock down gears within 15 seconds."

«requirement, functional requirement»
R12
id = "R12"
text = "If the handle is down, the LGS SHALL Close the doors within 15 seconds."

«requirement, interface requirement»
R111
id = "R111"
text = "If the pilot puts handle down, there is a handle down at handle port of the GE controller."

«requirement, interface requirement»
R112
id = "R112"
text = "if the LGS receives handle down at the handle port, a message lock down should be sent eventually to Extend EV in 15 s."

«requirement, interface requirement»
R113
id = "R113"
text = "To lock down gears, the gear controller Shall send a lock down message to Extend EV or Retract EV ports."

«requirement, interface requirement»
R121
«Requirement»
id=R121
text=To close doors, the door controller Shall send a close electrovalve message to the Close EV port.

«requirement, interface requirement»
R122
id = "R122"
text = "if the LGS receives handle down at the handle port, a mesaage close electrovalve should be sent eventually to Close EV in 15s."

Figure 3: The requirement diagram of the outgoing sequence for the landing gear system

requirements texts, the Boiler-Plate template is used to write requirements.

3.3 Specifying the system architecture

Figure 4 shows a SysML Internal Block Diagram (IBD) capturing the LGS system architecture. The purpose of this step is to develop an architecture diagram to provide an overview of the system structure consisting of both software and hardware blocks. The LGS system architecture consists of both hardware and software blocks. Specifically, all the blocks except the microcontroller block are hardware blocks, and the microcontroller block represents the software part of the LGS. The different sub-blocks that compose the system under analysis are connected through ports and connectors. According to the proposed methodology, all the ports of the LGS shown in the context diagram should be preserved in figure 2. These ports are : Handle IN, green light, orange light,red light, door status, gear statute, door control and gear control.

As shown in Figure 4, the landing gear system architecture is composed of 11 blocks. Each block is responsible for performing some tasks. For instance, gear sensor is in charge of providing the different states of gears: retracted, extended or gear shock absorbed and sending them the microcontroller. Door sensor is responsible of providing the states of the doors: opened or closed and sending them to the microcontroller. Handle will receive the pilot command in order to perform both sequences: retraction or outgoing sequence. The general electro-valve will receive an electrical order from the microcontroller in order to activate the general hydraulic circuit. The general electro-valve will send an hydraulic input pressure to the different electro-valves. These same electro-valves will receive

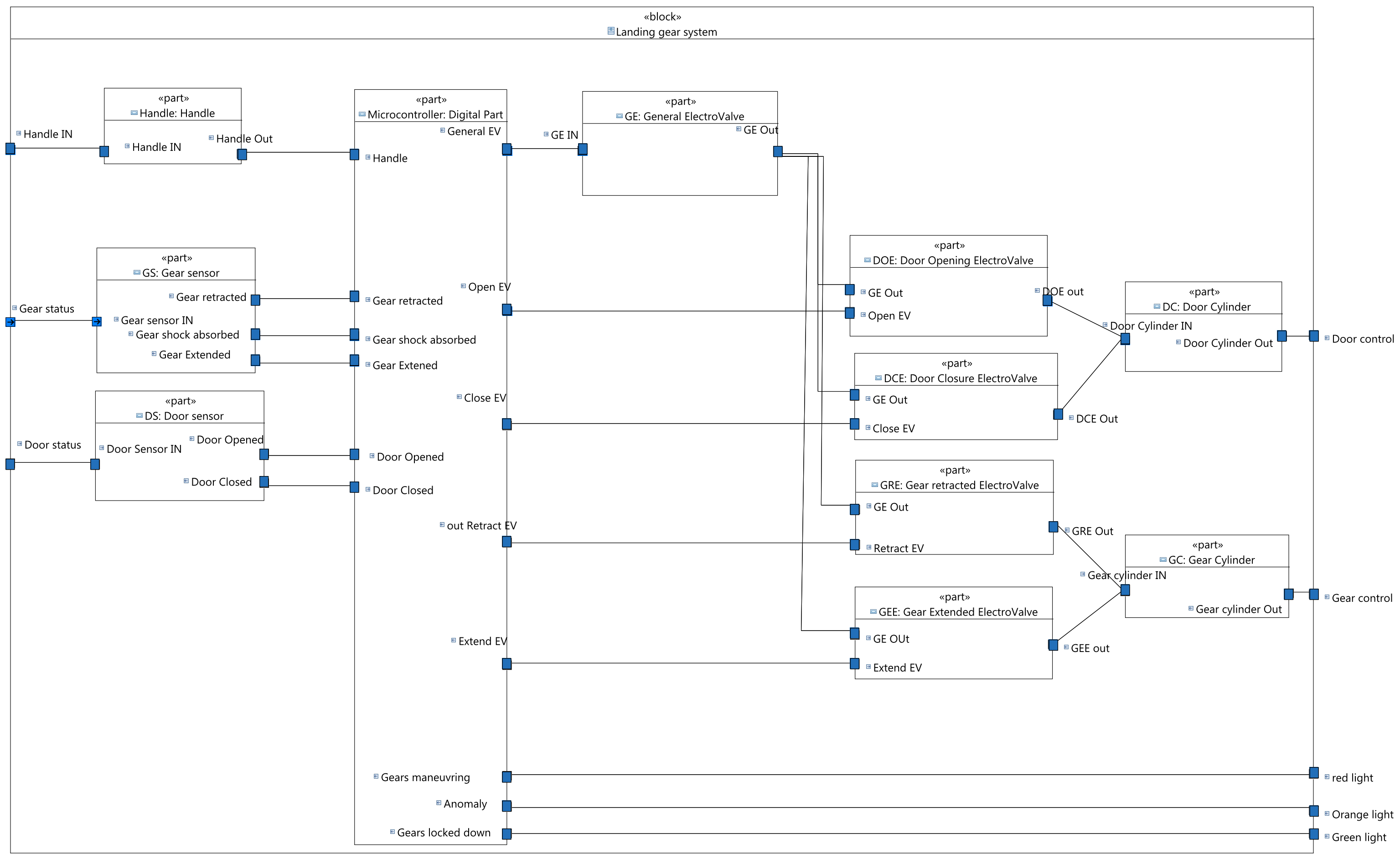


Figure 4: The system architecture diagram for the landing gear system

several electrical orders from the micro controller. Then the 4 electro-valves will stimulate both door and gear cylinders. The door cylinder will open or close doors and the gear cylinder will retract or extend the gears. In addition, the microcontroller will send three signals to the cockpit interface.

3.4 Specifying the software architecture

Figure 5 shows a SysML Block Diagram (IBD) capturing the microcontroller architecture. The purpose of this step is to describe the software architecture view by specifying the software blocks, software ports and their connectors. This step receives system architecture diagram and requirements diagrams as inputs and produces an Internal Block Diagram representing the software architecture as output.

As shown in Figure 5, general electro-valve controller, door controller and gear controller represent

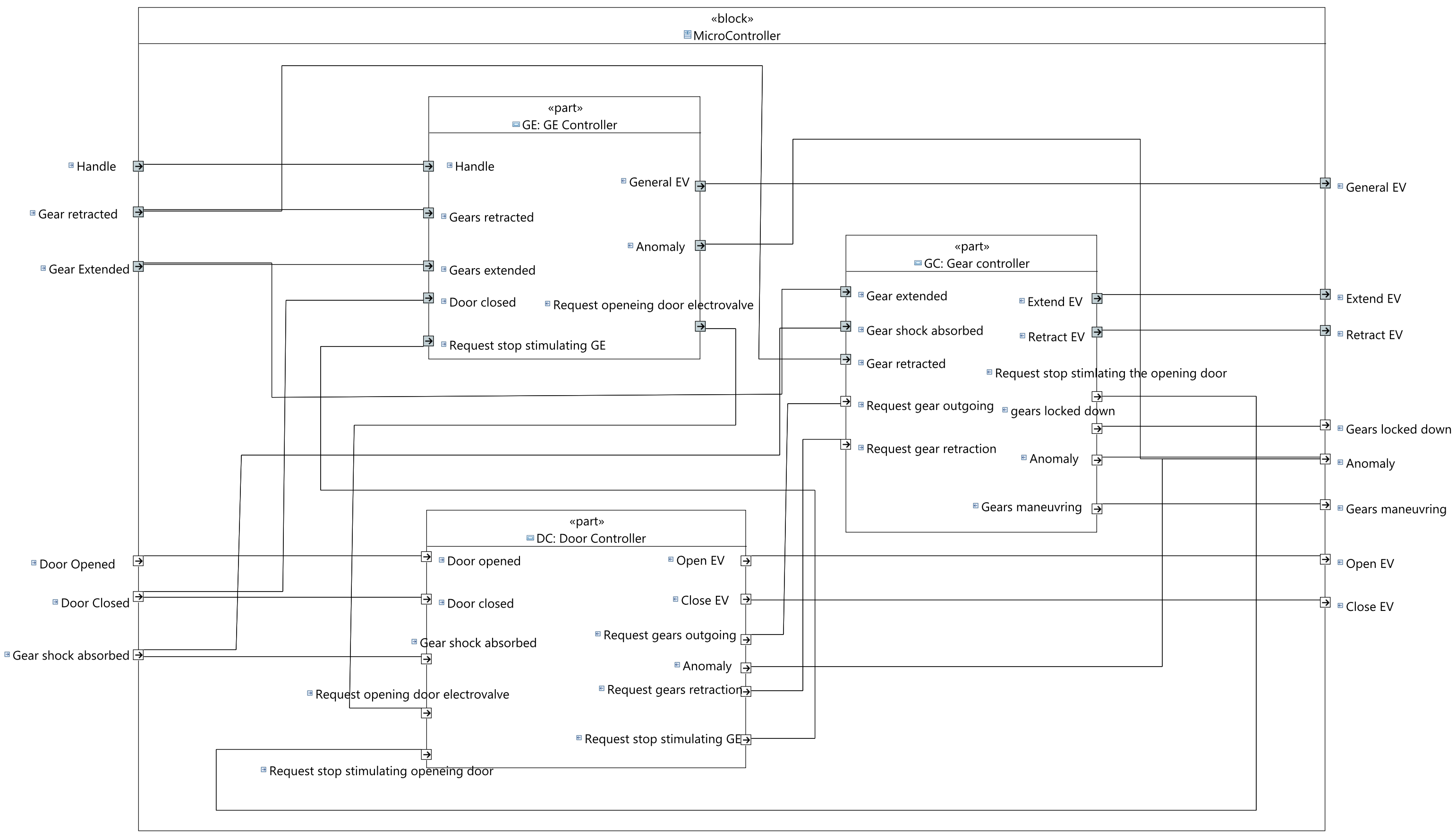


Figure 5: Software architecture diagram for the landing gear system.

the software blocks inside the landing gear system microcontroller from Figure 4. These software blocks interact with the hardware blocks in Figure 5 through the microcontroller ports: handle, a.switch, gear retracted, gear extended, gear shock absorbed, door opened, door closed, general EV, close EV, retract EV, extend EV, gears locked down, anomaly and gears manoeuvring. Thus, according to the proposed methodology, ports in Figure 4 must be preserved in Figure 5 and be represented as the external input/output ports of the microcontoller's IBD. "SoftwareBlock" stereotype is used to describe the three software sub-blocks of the microcontroller.

Also and according to the microcontroller behavior (outgoing and retraction sequence), the different software blocks interact among each others by sending requests. For instance, in order to stimulate the opening door electro-valve, the general electro-valve interacts with the door controller through a " Request opening door electro-valve ". The two ports of both door controller and general electro-valve need to have the same name. To stimulate the outgoing of gears, the door controller interacts with the gear controller through a " Request gears outgoing". To stimulate the gears retraction, the door controller interacts with the gear controller through a “Request gears retraction ”. To stop the stimulation of the general electro-valve, the gear controller interacts with the the general electro-valve controller through a “Request stop stimulating GE ”. To stop stimulating the opening door electro-valve, the gear controller interacts with the door controller through a “ Request stop stimulating the opening door electro-valve” .

Environmental assumptions Environmental assumptions are very important to consider and specify, as not taking them into account can have serious consquences. [?] In the case of the landing gear system, several assumptions should be taken into account in order to model correctly requirements, structure and behaviour of the system. As shown in the system architecture, the landing gear system contains a door sensor and a gear sensor. Door sensor provides two boolean values: door opened and door closed. If door opened is true, the door is eventually opened. if door closed is true, the door is eventually closed. Also doors can be in a transition state. In this case, door opened is false and door closed is false. Gear sensor provides two boolean values: gear extended and gear retracted. If gear extended is true, gears are eventually extended. If gears retracted is true, gears are eventually retracted. Also gears can be in a transition state. In this case, gear extended is false and gear retracted is false. The microcontroller generates several electrical orders to the electro-valves and to the cockpit interface. If General EV is true and Open EV is true, the door will be eventually opened. If General EV is true and Close EV is true, the door will be eventually closed. If General EV is true and Extend EV is true, the gear will be eventually extended. If General EV is true and Retract EV is true, the gear will be eventually retracted. If gears extended is true, it means that gears locked down is true, the green light in the cockpit will be activated. if gears extend is false and gear retracted is false, it means that gears manoeuvring is true, the orange light in the cockpit interface will be activated.

3.5 Specifying the software block behaviour

The purpose of this step is to represent the behaviour of individual software blocks using activity diagrams. In addition, by using activity diagrams, we show, for each software block, how its outputs are computed based on its inputs and its internal variables. This step receives software architecture diagrams and requirements diagrams as inputs and produces activity diagrams as outputs. In the landing gear system case study, we have three software blocks inside the microcontroller software block. Then, three activity diagrams should be produced in order to describe the behaviour of these three software blocks.

3.5.1 Specifying the general electro-valve controller (GE controller) behaviour

Figure 6 shows the activity diagram of the general electro-valve controller. First of all, handle status is checked using a decision node. Handle can be UP or DOWN following the pilot command. Using our defined profile, handle is activity parameter node and an input for the general electro-valve controller activity diagram.

That is why handle is shown in Figure 6 as in to say that is an input and data to say that handle is an activity parameter node because it is related to data flow and not to control flow. If handle is down, it means that the outgoing sequence is observed. Gear retracted is checked using another decision node. Gears retracted is boolean. If gears retracted is true, door closed status is checked using a decision node. Else, if gears retracted is false and if after 15 seconds, the value of gears retracted does not change, an anomaly is detected. 15 seconds represents a calibration variable. Anomaly is assigned to true. If door closed is true, General EV is assigned to true. Else, if door closed is false and if after 15 seconds, the value of door closed does not change, an anomaly is detected. Also 15 seconds is a calibration variable. Anomaly is assigned to true. After stimulating the general electro-valve, the general electro-valve controller needs to request opening door electro-valve. Thus, it will send a request to the door controller. Request opening door electro-valve is an activity parameter node and an output for general electro-valve controller. When this controller got the request stop stimulating GE from the door controller, General EV is assigned to false and then both sequences(outgoing and retraction) are finished.

3.5.2 Specifying the door controller behaviour

Figure 7 shows the activity diagram of the door controller. After receiving a request opening door door electro-valve from the general electro-valve controller, open EV is assigned to true and is sent through an object flow to Open EV which is the output of the door controller block. Open EV is an activity parameter node and output for this activity diagram. After that, door opened status is checked via a decision node. Door opened is boolean. In and data stereotype are used to define

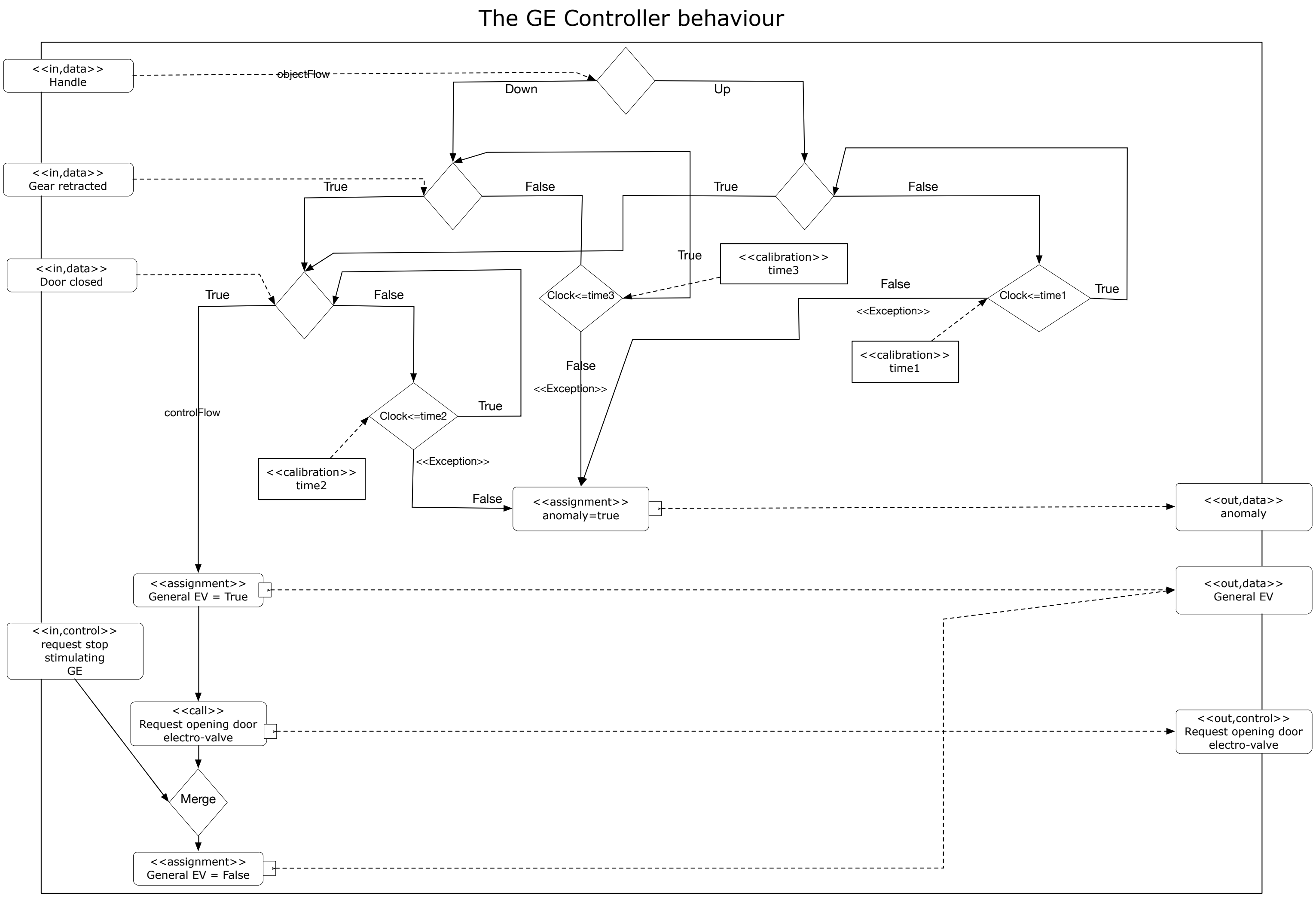
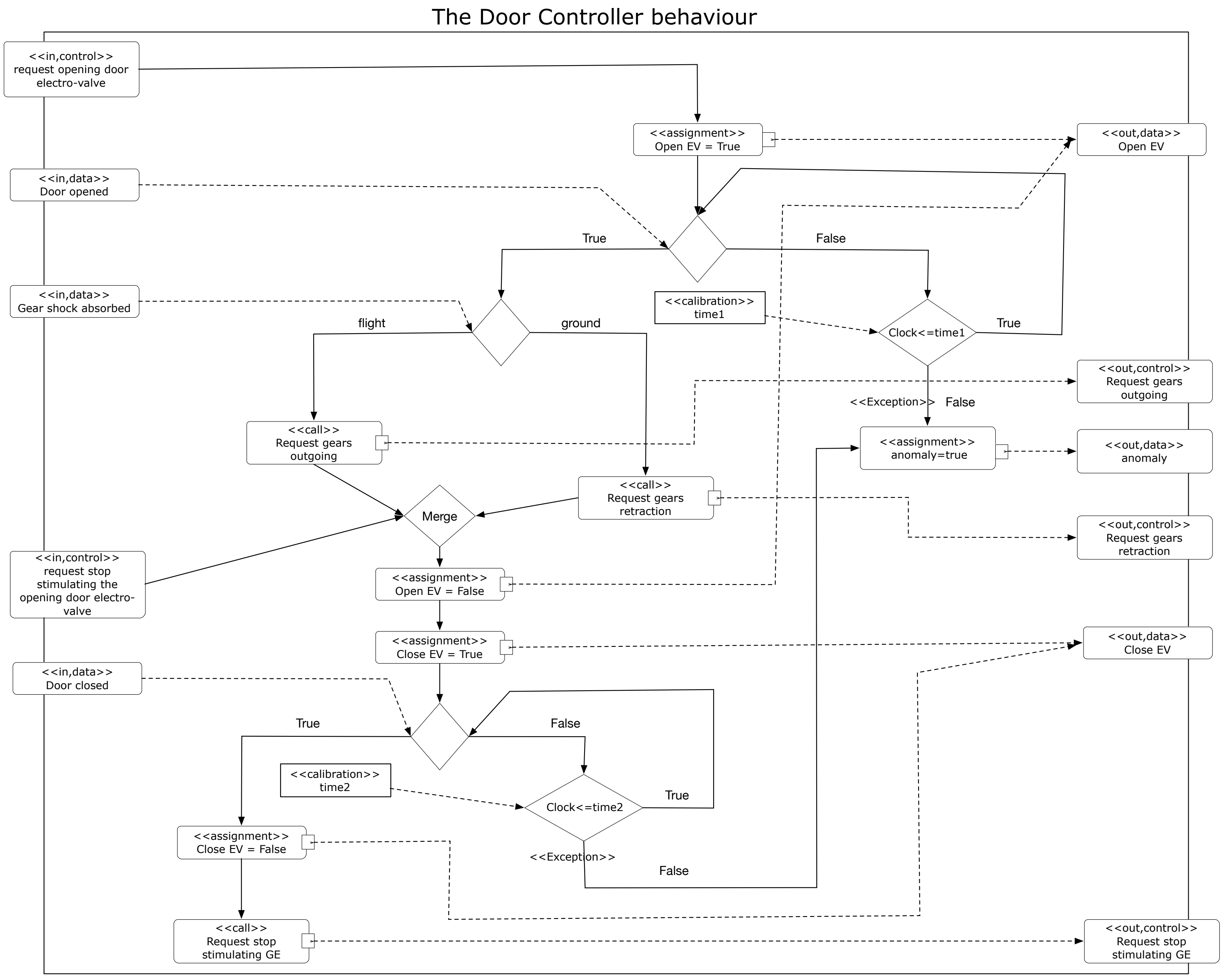


Figure 6: An activity diagram representing the behaviour of the general electro-valve controller software block.

door opened. In because door opened is an input for the activity diagram and data because door opened is a data flow. If door opened is true, gears shock absorbed is checked through a decision node. Else, if door opened is false and if after 7 seconds, the value of door opened does not change, an anomaly is detected. 7 seconds represents a calibration variable.

Anomaly is assigned to true. Gears shock absorbed is a sensor implemented on gears. It is true if and only if the aircraft is on ground. The values of the gears shock absorbed belong to the set ground,flight. If gears shock absorbed is equal to flight, the outgoing sequence is observed and then a request gear outgoing is sent to the gear controller. Else, the retraction sequence is observed and a request gear retraction is sent to the gear controller. After receiving a request stop stimulating the opening door from the gear controller, both scenarios(outgoing and retraction sequences) will have the same behaviour.

Open EV is assigned to false. After that, the door closure is stimulated and Close EV is assigned



The Door Controller behaviour
<<in,control>> request opening door electro-valve
<<assignment>> Open EV = True
<<out,data>> Open EV
<<in,data>> Door opened
True
False
<<in,data>> Gear shock absorbed
<<calibration>> time1
flight
ground
Clock<=time1
True
<<out,control>> Request gears outgoing
<<Exception>>
False
<<call>> Request gears outgoing
<<assignment>> anomaly=true
<<out,data>> anomaly
<<call>> Request gears retraction
Merge
<<out,control>> Request gears retraction
<<in,control>> request stop stimulating the opening door electro-valve
<<assignment>> Open EV = False
<<assignment>> Close EV = True
<<out,data>> Close EV
<<in,data>> Door closed
True
False
<<calibration>> time2
Clock<=time2
True
<<assignment>> Close EV = False
<<Exception>>
False
<<call>> Request stop stimulating GE
<<out,control>> Request stop stimulating GE

Figure 7: An activity diagram representing the behaviour of the door controller software block.

to true. Close EV is an activity parameter node and output for this activity diagram. Door closed is checked through a decision node. If door closed is true, Close EV is assigned to false to say that the door controller is stopping the stimulation of the door closure. Else, if door closed is false, and if after 7 seconds, the value of door closed does not change, an anomaly is detected. 7 seconds represents a calibration variable. Anomaly is assigned to true. After stopping the stimulation of doors closure, the door controller sends a request stop stimulating GE to the general electro-valve.

3.5.3 Specifying the gear controller behaviour

Figure 8 shows the activity diagram of the gear controller. two scenarios are foreseen.

If the gear controller receives a request gear outgoing, the outgoing sequence will be observed.

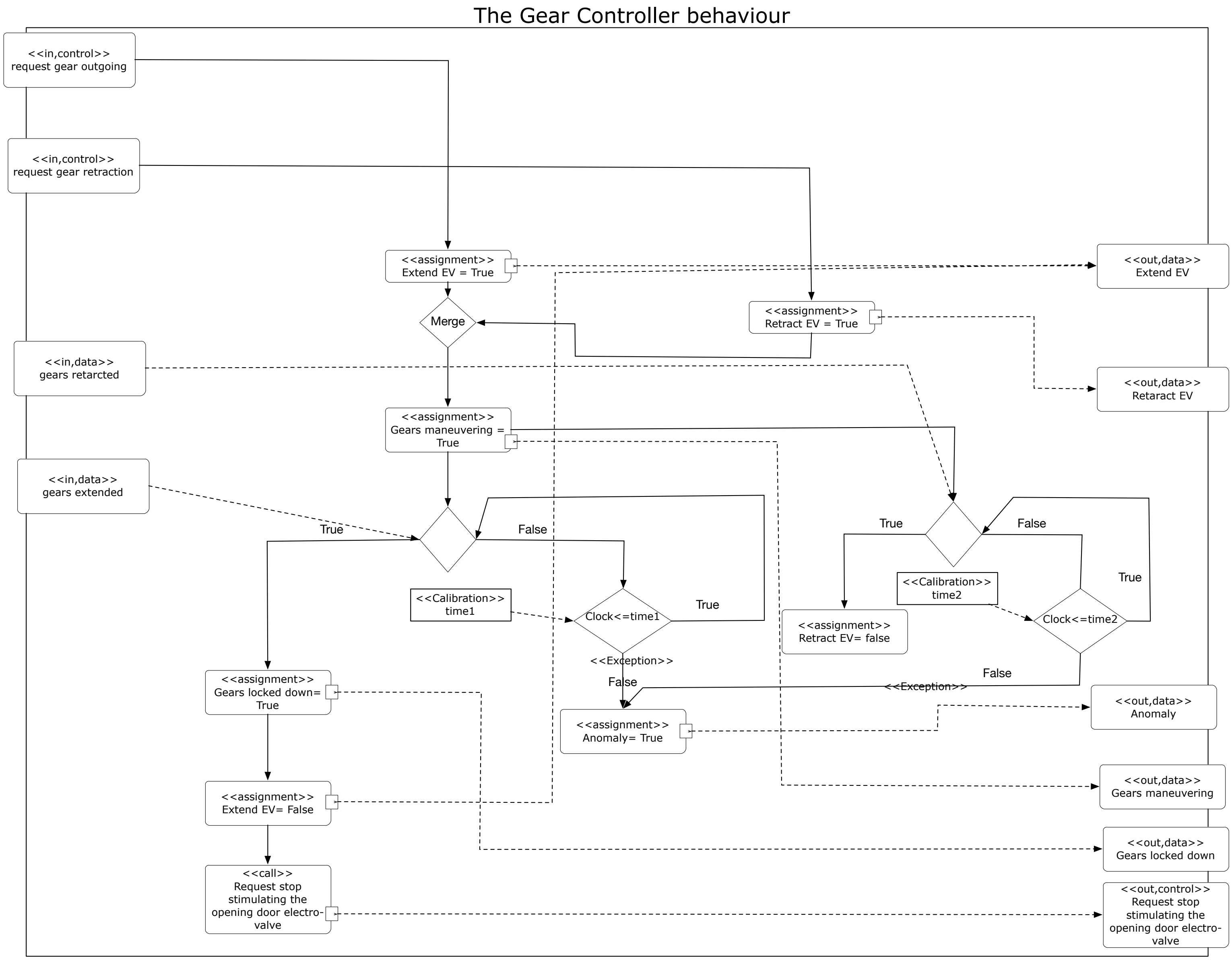

The Gear Controller behaviour
<<in,control>> request gear outgoing
<<in,control>> request gear retraction
<<assignment>> Extend EV = True
<<out,data>> Extend EV
Merge
<<assignment>> Retract EV = True
<<in,data>> gears retarcted
<<out,data>> Retaract EV
<<assignment>> Gears maneuvering = True
<<in,data>> gears extended
True
False
True
False
<<Calibration>> time1
Clock<=time1
True
<<Calibration>> time2
True
<<assignment>> Retract EV= false
Clock<=time2
<<Exception>>
False
False
<<Exception>>
<<assignment>> Gears locked down= True
<<out,data>> Anomaly
<<assignment>> Anomaly= True
<<assignment>> Extend EV= False
<<out,data>> Gears maneuvering
<<out,data>> Gears locked down
<<call>> Request stop stimulating the opening door electro-valve
<<out,control>> Request stop stimulating the opening door electro-valve

Figure 8: An activity diagram representing the behaviour of the gear controller software block.

If the gear controller receives a request gear retraction, the retraction will be observed. We start by the outgoing sequence. Extend EV is assigned to true. Extend EV is an activity parameter node and output for this activity diagram. While gears are extended, gears maneuvering is assigned to true. If gears extend is true, gears locked down is assigned to true. Else, if gears extended is false, and if after 7 seconds, the value of gears extended does not change, an anomaly is detected. 7 seconds represents a calibration variable. If gears are locked down, the gear controller should stop the stimulation of gears extension. Then, Extend EV is assigned to false and after that the gear controller needs to send a request to stop stimulating the opening door to the door controller. We move to the retraction sequence. Retract EV is assigned to true. Retract EV is an activity parameter node and output for this activity diagram. while gears are retracted, gears maneuvering is assigned to true. If gears retracted is true, the gear controller should stop stimulating the gears retraction and Retract EV is assigned to false. Else, if gears retracted is false, and if after 7 seconds, the value of gears retracted does not change, an anomaly is detected. 7 seconds represents a calibration variable. If gears are retracted, the gear controller needs to send a request to stop stimulating the opening door to the door controller.

3.6 Specifying requirements to blocks traceability links

Figures 9, 10, 11 and 12 describe four different examples of requirements to blocks traceability. The purpose of this step is to develop traceability links from hardware and software requirements to hardware and software blocks. It receives requirements diagrams, system architecture diagram and software architecture diagram as inputs and produces traceability links from requirements to blocks as outputs. The example in Figure 9 shows that the interface requirement R111 is traced to the GE Controller software block, the Gear Controller software block and the Door Controller software block. In this figure, more detailed traceability information are provided through rationales. R111 is related to the ports Handle, General EV and Request door opening electo-valve of the GE Controller software block. More detailed information can be shown in the rationale such as activity nodes that describe the behaviour of a software block. As shown in Figure 9, R111 is related to the GE Controller software block trough two activity nodes: an assignment General EV and a call Request opening door electro-valve.

R111 is related to the ports Request gears outgoing, Request gears retraction, Extend EV and Retract EV of the Gear Controller software block. R111 is related to the Gear Controller trough two

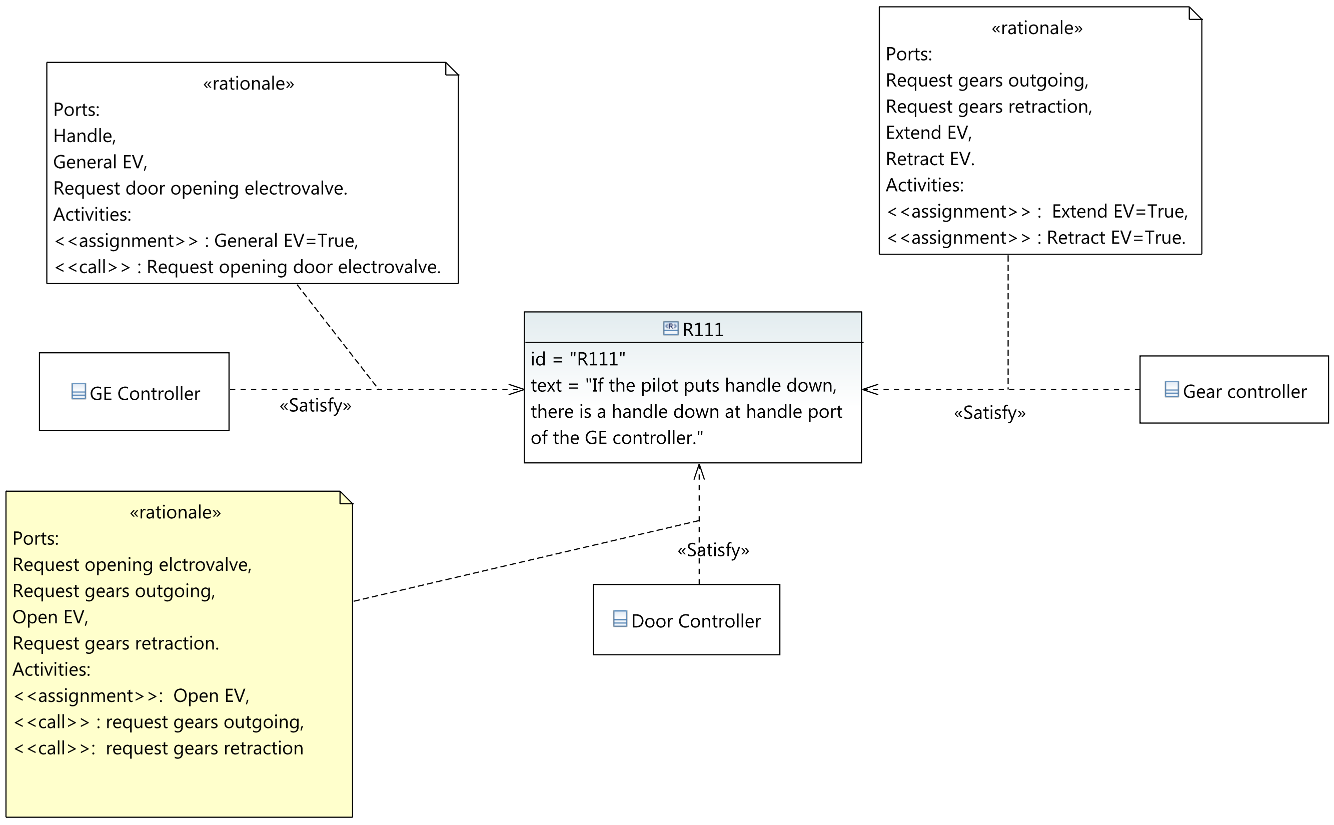

«rationale»
Ports:
Handle,
General EV,
Request door opening electrovalve.
Activities:
<<assignment>> : General EV=True,
<<call>> : Request opening door electrovalve.
«rationale»
Ports:
Request gears outgoing,
Request gears retraction,
Extend EV,
Retract EV.
Activities:
<<assignment>> : Extend EV=True,
<<assignment>> : Retract EV=True.
R111
id = "R111"
text = "If the pilot puts handle down, there is a handle down at handle port of the GE controller."
GE Controller
«Satisfy»
«Satisfy»
Gear controller
«rationale»
Ports:
Request opening elctrovalve,
Request gears outgoing,
Open EV,
Request gears retraction.
Activities:
<<assignment>>: Open EV,
<<call>> : request gears outgoing,
<<call>>: request gears retraction
«Satisfy»
Door Controller

Figure 9: A traceability link from a interface requirement to the Door Controller, Gear Controller and GE Controller software blocks.

assignments: Extend EV and Retract EV. R111 is related to the ports Request opening electro-valve, Request gears outgoing, Open EV and Request gears retraction of the Door Controller software block. R111 is related to the Door Controller trough an assignment (Open EV) and two calls (Request gears outgoing and Request gears retraction).
The example in Figure 10 shows that the interface requirement R113 is traced to the Gear Controller software block and to the Gear retracted electro-valve, Gear Cylinder, Gear sensor, Gear Extended Electro-valve hardware blocks. In this figure, more detailed traceability information are provided through rationales. R113 is related to the ports Extend EV and Retract EV of the Gear Controller. R113 is related to the Gear Controller through two assignments: Extend EV and Retract EV. R113 is related to the ports Gear cylinder IN and Gear cylinder out of the Gear Cylinder hardware block. R113 is related to the ports Retract EV and GRE out of the Gear retracted Electro-valve hardware block.

And Finally, R113 is related to the ports Gear status, Gear extended and Gear retracted of

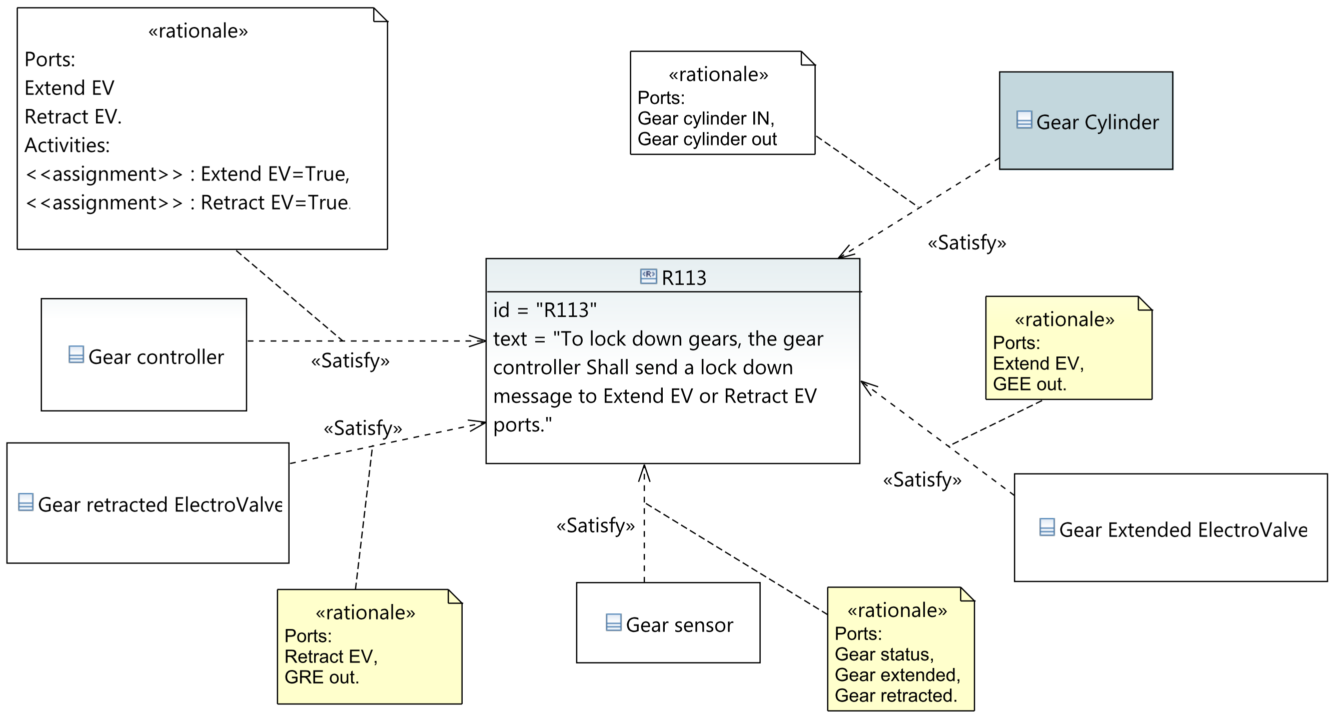


Figure 10: A traceability link from an interface requirement to the Gear Controller software block and to the Gear retracted electro-valve, Gear Cylinder, Gear sensor, Gear Extended Electro-valve hardware blocks.

the Gear sensor hardware block. The example in Figure 11 shows that the interface requirement R121 is traced to the Door Controller software block and to the Door Sensor, Door Cylinder and Door Closure Electro-valve hardware blocks. R121 is related to the ports Close EV, Request gears outgoing, Request gears retraction and Request stop stimulating the opening door of the Door Controller software block. R121 is related to the Door Controller through an assignment (Close EV) and two calls (Request gears outgoing and Request gears retraction). R121 is related to the ports Close EV and DCE out of the Door Closure Electro-valve hardware block. R121 is related to the ports Door Cylinder IN and Door Cylinder out of the Door Cylinder hardware block. And finally, R121 is related to the ports Door status and Door closed of the Door sensor hardware block. The example in Figure 12 shows that the interface requirement R122 is traced to the GE Controller, Door Controller and Gear Controller software blocks. R122 is related to the ports Handle, General EV and Request opening door electro-valve of the GE Controller. R122 is related to the GE Controller through an assignment General EV and a call Request opening door electro-valve. R122 is related to the ports Request gear outgoing, Request gears retraction, Extend EV, Retract EV,

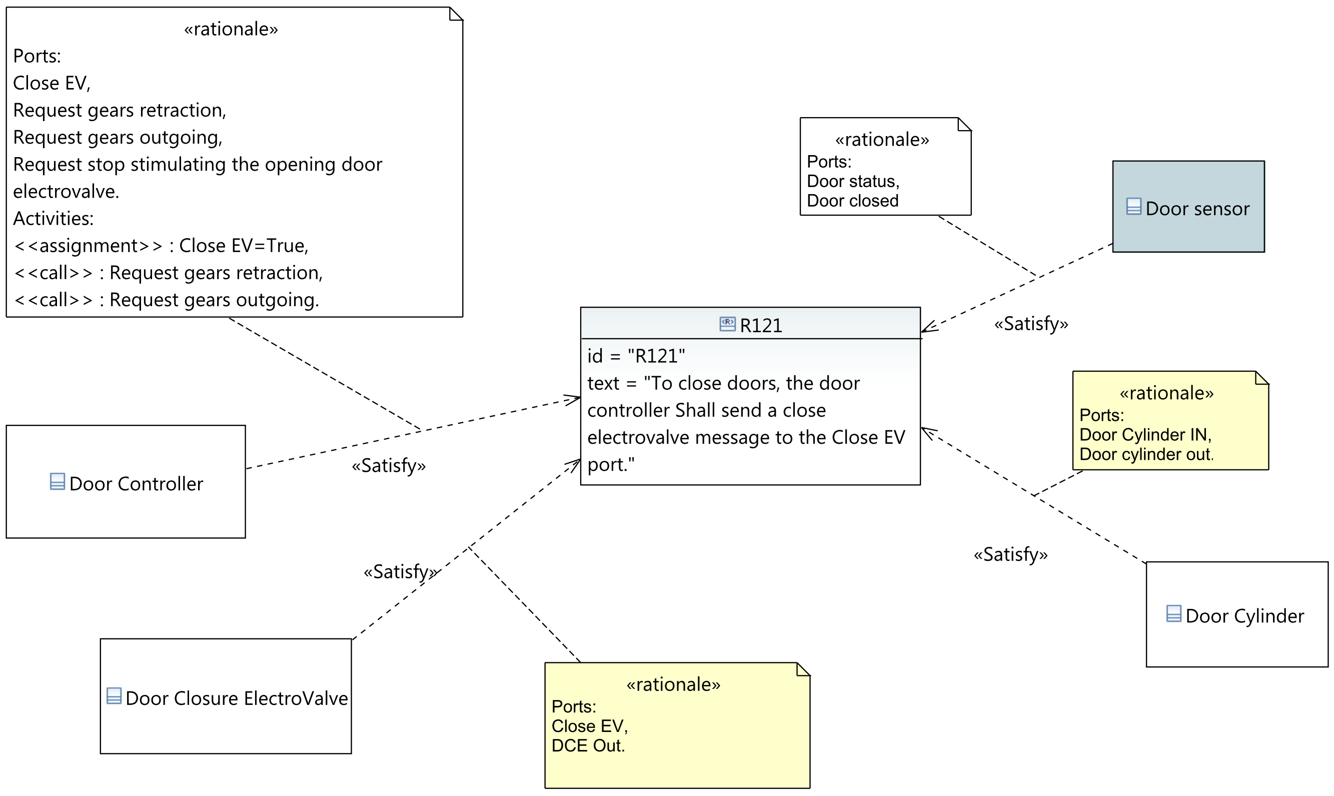

«rationale»
Ports:
Close EV,
Request gears retraction,
Request gears outgoing,
Request stop stimulating the opening door electrovalve.
Activities:
<<assignment>> : Close EV=True,
<<call>> : Request gears retraction,
<<call>> : Request gears outgoing.
«rationale»
Ports:
Door status,
Door closed
Door sensor
R121
id = "R121"
text = "To close doors, the door controller Shall send a close electrovalve message to the Close EV port."
«Satisfy»
«rationale»
Ports:
Door Cylinder IN,
Door cylinder out.
Door Controller
«Satisfy»
«Satisfy»
«Satisfy»
Door Cylinder
Door Closure ElectroValve
«rationale»
Ports:
Close EV,
DCE Out.

Figure 11: A traceability link from an interface requirement to the door cylinder, door sensor and door closure electro-valve hardware blocks and to the door controller software block.

Request stop stimulating the opening door electro-valve, Gears manoeuvring and Gears locked down of the Gear Controller. R122 is related to the Gear Controller through four assignments : Extend EV , Gears locked down, gears manoeuvring and Retract EV and a call Request stop stimulating the opening door electro-valve. R122 is related to the ports Open EV, Request gears outgoing, Request gears retraction, Close EV and Request stop stimulating the opening door electro-valve of the Door Controller. Finally, R122 is related to the Door Controller through two assignments: Open EV and Close EV and two calls: Request gears outgoing and Request gears retraction.

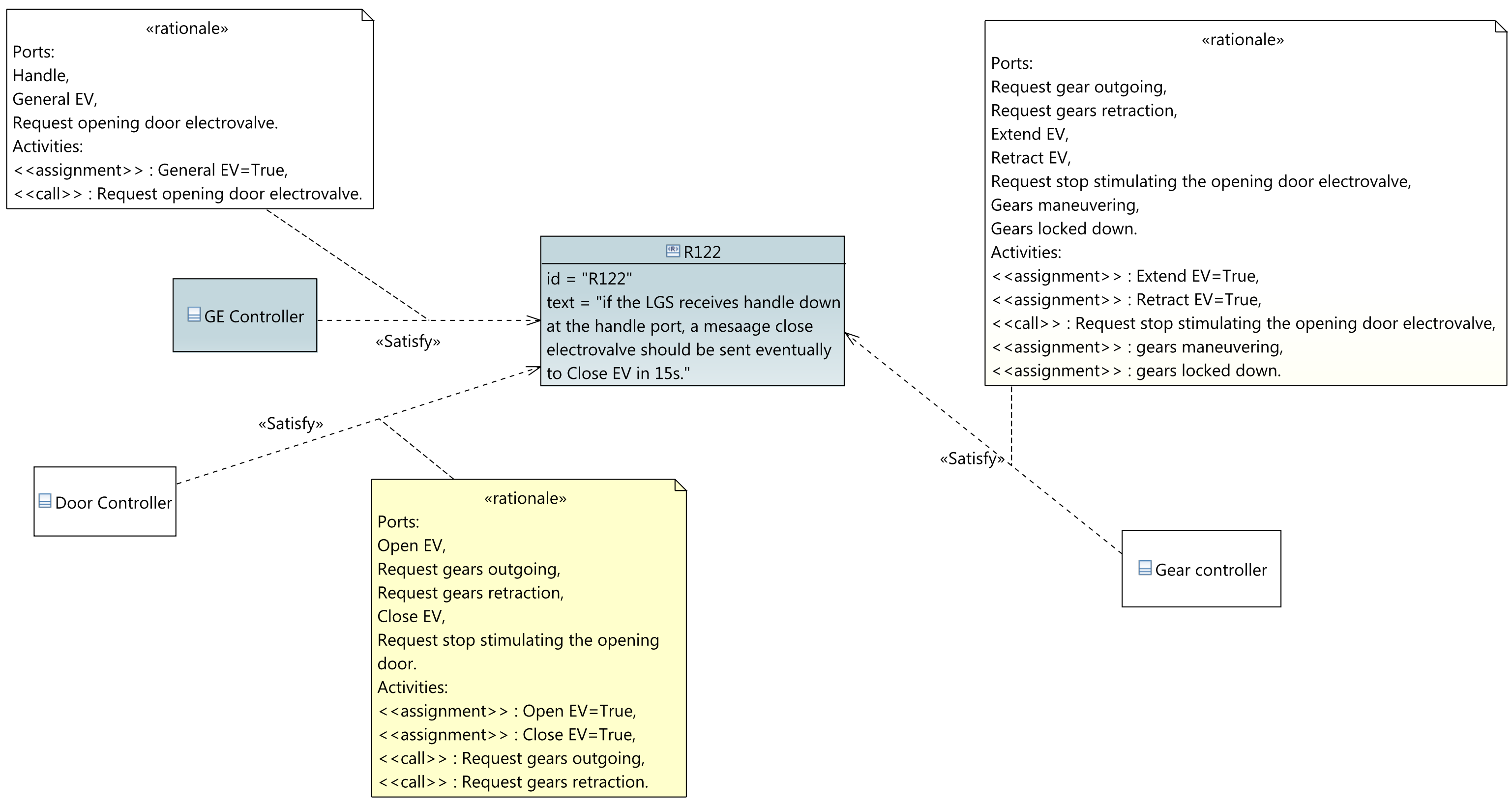

«rationale»
Ports:
Handle,
General EV,
Request opening door electrovalve.
Activities:
<<assignment>> : General EV=True,
<<call>> : Request opening door electrovalve.
GE Controller
«Satisfy»
R122
id = "R122"
text = "if the LGS receives handle down at the handle port, a mesaage close electrovalve should be sent eventually to Close EV in 15s."
«Satisfy»
Door Controller
«rationale»
Ports:
Open EV,
Request gears outgoing,
Request gears retraction,
Close EV,
Request stop stimulating the opening door.
Activities:
<<assignment>> : Open EV=True,
<<assignment>> : Close EV=True,
<<call>> : Request gears outgoing,
<<call>> : Request gears retraction.
«rationale»
Ports:
Request gear outgoing,
Request gears retraction,
Extend EV,
Retract EV,
Request stop stimulating the opening door electrovalve,
Gears maneuvering,
Gears locked down.
Activities:
<<assignment>> : Extend EV=True,
<<assignment>> : Retract EV=True,
<<call>> : Request stop stimulating the opening door electrovalve,
<<assignment>> : gears maneuvering,
<<assignment>> : gears locked down.
«Satisfy»
Gear controller

Figure 12: A traceability link from an interface requirement to the GE Controller, Door Controller and Gear Controller software blocks.