Project ADVANCE
Grant Agreement 287563
"Advanced Design and Verification Environment for Cyber-physical System Engineering"

ADVANCE Report

Tutorial on ADVANCE Safety Analysis Process

John Colley, University of Southampton

25 November 2014

http://www.advance-ict.eu

1 Introduction

This report provides an overview of the ADVANCE process flow, combining formal modelling with requirements analysis and safety analysis, a process which has now been applied to both industrial case studies. We present a tutorial to guide the user through the ADVANCE process flow covering STPA-based safety analysis, formal modelling and refinement in Event-B, verification and testing. We have chosen the landing gear case study as the basis for this ADVANCE process tutorial because it typifies the kind of safety-critical system that ADVANCE addresses, is easy to understand without necessarily being a domain expert and a comprehensive, well-defined set of requirements is available in the public domain [BW14].

2 Requirements and Safety Analysis

In the ADVANCE process, requirements and safety analysis are closely integrated. We consider the requirements systematically in terms of the

- Monitored Phenomena
- Controlled Phenomena
- Commanded Phenomena
- Mode Phenomena

The Controlled Phenomena

It is the *Controlled Phenomena* which provides the link to the safety analysis process. Consider the requirements for the *Door Sub-system* of the aircraft landing gear.

- The Controller will open the Doors when the Pilot moves the Lever to Extend or Retract the Landing Gear
- The Controller will then close the Doors when the Landing Gear is fully Extended or Retracted
- The Doors will remain open while the Landing Gear is Extending or Retracting

The ADVANCE process introduces safety analysis at the very beginning, to ensure that safety considerations are addressed as early as possible. We use SystemTheoretic Process Analysis (STPA) [Lev12] which is performed in two phases.

- Identify Potentially Hazardous Control Actions and derive the Safety Constraints
- Determine how Unsafe Control Actions could occur

Identifying Potentially Hazardous Control Actions

The landing gear system has effectively two controllers: the pilot, who has a high-level view of the position of the landing gear, according to the position of the extend/retract handle in the cockpit, and the digital controller, which controls the position of the doors, using the *actuators*, according to the position of the pilot handle and the landing gear. The digital controller monitors the position of the doors and updates the state of its internal process model using the *sensors*. If the sensor values are inconsistent with the process model, the controller can notify the pilot of a potential system failure. The process models are shown in Figure 1 below.

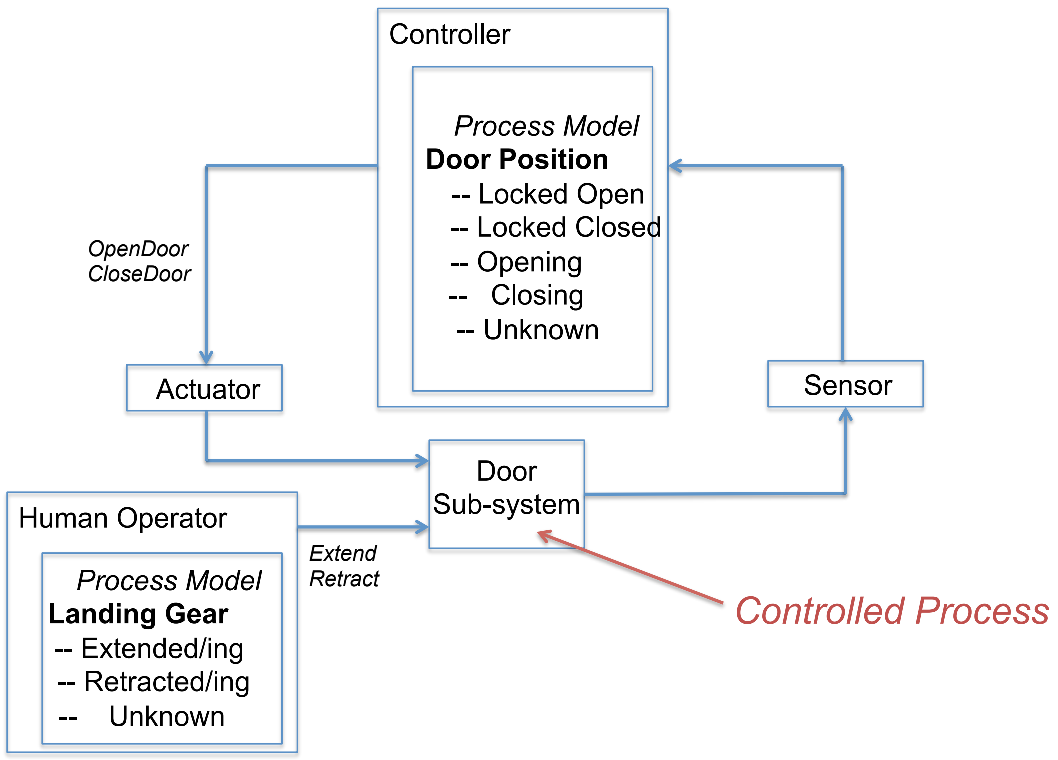

Figure 1: The Landing Gear Doors Process Model

For each of the controller door actions, *Open Door* and *Close Door*, we identify in a systematic way, how these actions can be hazardous as shown in Figure 2.

Controller Action	Not Providing Causes Hazard	Providing Causes Hazard	Wrong Timing or Order Causes Hazard	Stopped too soon/Applied too long
Open Door	Cannot extend Landing Gear for landing	Not Hazardous	Damage to Landing Gear	Damage to Landing Gear/ Not Hazardous
Close Door	Not Hazardous	Damage to Landing Gear	Damage to Landing Gear	Not Hazardous/ Not Hazardous

Figure 2: Safety Analysis for Door Sub-system

The controller not opening the door when it should is hazardous as the landing gear cannot be extended for landing, but opening the door when it shouldn't is not hazardous. Opening the doors after the landing gear has begun to extend or retract is hazardous, as is failing to complete the opening procedure. A similar analysis is then performed on the *Close Door* action.

From this table we are able to derive the natural language safety constraints.

1. If the Landing Gear is Extending, the Door must be Locked Open
2. If the Landing Gear is Retracting, the Door must be Locked Open
3. A *Close Door* command must only be issued if the Landing Gear is Locked Up or Locked Down
4. An *Open Door* command must only be issued if the Landing Gear is Locked Up or Locked Down

3 Modelling the Landing Gear System

The Abstract Model

We begin with an abstract model of the system which represents just the *gearstate*. The landing gear may be *locked_up*, *locked_down* or, because we wish to model the temporal nature of the system, *extending* or *retracting*. Four events define the transitions between these states: *Extend* and *Retract* represent a requested operation, initiated by the pilot, and *CompleteExtend* and *CompleteRetract* are observed when the requested operation is completed.

The abstract model is illustrated by the state machine shown in Figure 3 below. Notice that when the landing gear is in the process of *extending* or *retracting*, the pilot can at any time move the landing gear handle position to reverse the command.

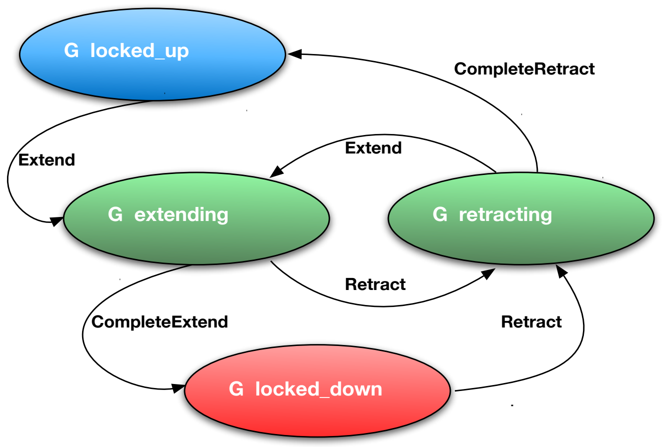

Figure 3: The Abstract Model

The First Refinement

We now introduce the door and establish formally the relationship between the gear and the door from the natural language safety constraints.

1. If the Landing Gear is Extending, the Door must be Locked Open
2. If the Landing Gear is Retracting, the Door must be Locked Open
3. A *Close Door* command must only be issued if the Landing Gear is Locked Up or Locked Down
4. An *Open Door* command must only be issued if the Landing Gear is Locked Up or Locked Down

The first two safety constraints are represented by the invariant *inv3* below.

inv3 : $gearstate \in \{extending, retracting\} \Rightarrow doorstate = locked\_open$

The second two safety constraints are modelled by the guards *grd1* in the *Open* and *Close* events.

event $Open \mathrel{\widehat{=}}$

  when

    grd1 : $gearstate \in \{locked\_down, locked\_up\}$
    grd2 : $doorstate \in \{closing, locked\_closed\}$

  then

    act1 : $doorstate := opening$

  end

event $Close \mathrel{\widehat{=}}$

  when

    grd1 : $gearstate \in \{locked\_down, locked\_up\}$
    grd2 : $doorstate \in \{opening, locked\_open\}$

  then

    act1 : $doorstate := closing$

  end

Running the Rodin automatic provers establishes that the formal system-level safety constraints are preserved by the refinement. We can represent the refined model as a state machine in terms of the two variables *gearstate* and *doorstate* as shown in Figure 4 below.

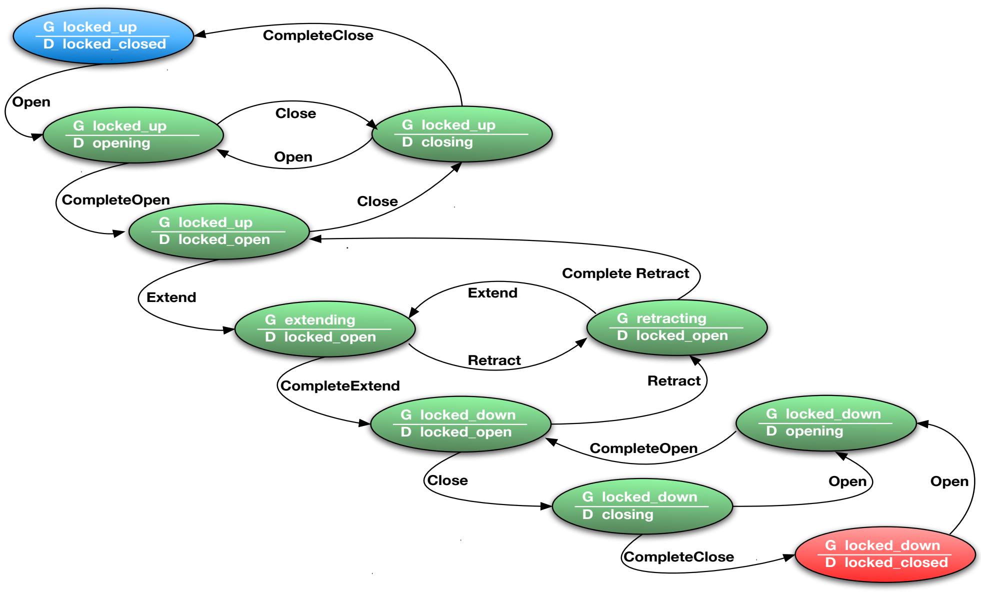

Figure 4: The First Refinement

The Second Refinement

In this refinement, we introduce the pilot *handle* and model the synchronous timing and synchronisation necessary to represent the concurrency of the system. We do not at this stage model actual times or delays but simply introduce the notion of a *tick*. For any given tick of the system, the handle may or may not be moved; the digital controller responds to any handle change in the next tick.

To represent the latency of the system in an abstract way, we introduce the event *Idle* as shown in the extended finite state machine, defined in terms of the three variables *gearstate*, *doorstate* and *handle*, in Figure 5 below.

Only 12 of the 32 possible states for *(gearstate;doorstate;handle)* are valid. To ensure that the system never makes a transition to one of the 20 invalid states we introduce and prove a set of invariants. For instance, invariant *inv10* below represents the fact that the door cannot be opening if the gear is locked up and the handle is up.

$$\texttt{inv10} : \neg(gearstate = locked\_up \land doorstate = opening \land handle = UP)$$

We now run the ADVANCE model checker to ensure that all the valid states of the model are reachable and there is no deadlock. At this high level of abstraction an exhaustive model check can be completed in seconds.

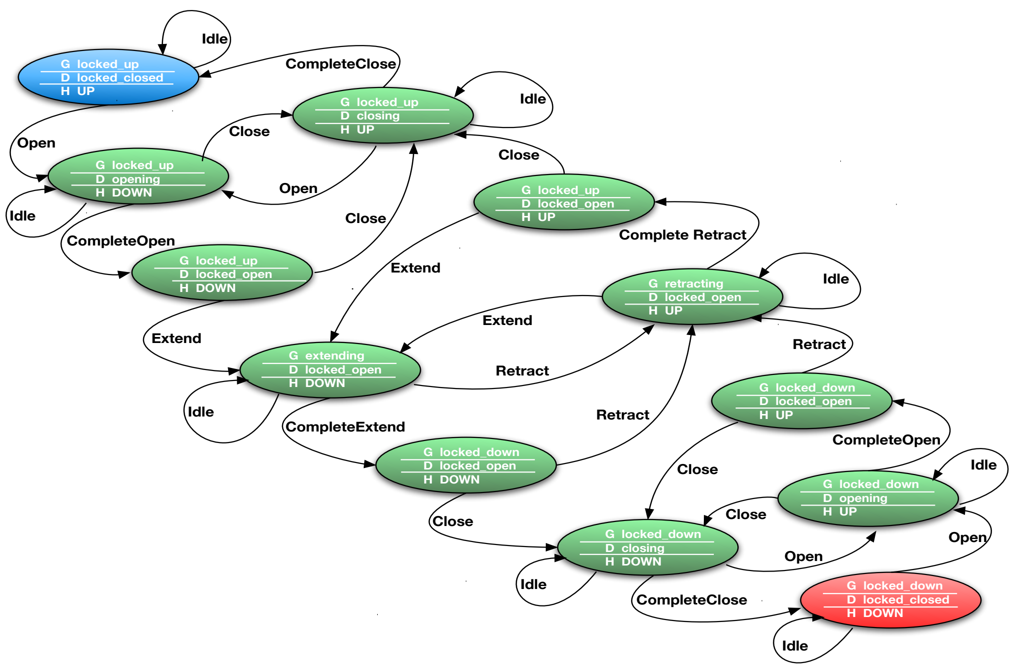

Figure 5: The Second Refinement

Introducing the Timing Deadlines

We now introduce the concrete signals between the Controller and the landing gear sub-system as shown in Figure 6 below.

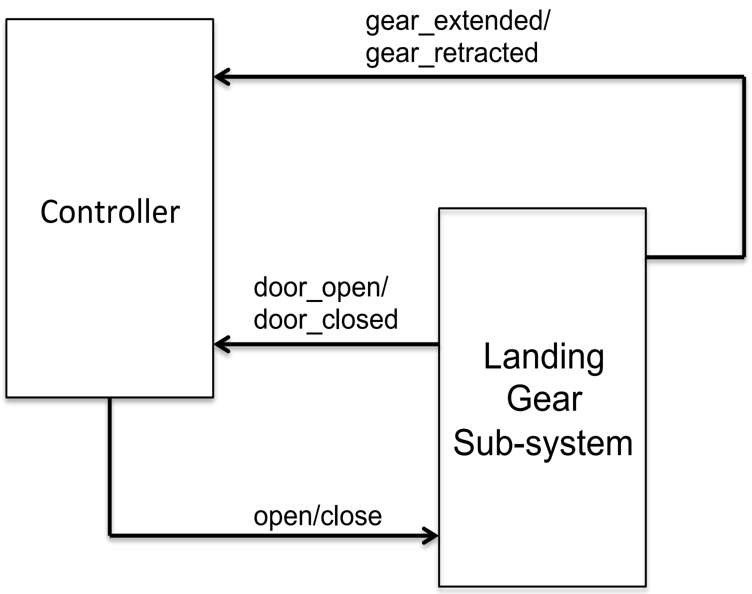

Figure 6: The Component View

The landing gear requirements detail a set of timing constraints for each of the mechanical and hydraulic procedures of the extending and retracting sequences. We introduce these constraints, systematically using refinement, as deadlines which refine the abstract *Idle* events.

Rather than idling indefinitely, the model sets an abstract timer *count* which is decremented if the confirmation signal from the landing gear sub-system has not been received. If confirmation is received before the deadline expires, the operation can complete. If, however, the count reaches zero without confirmation, the controller moves to a fail state and switches on a warning light on the control panel. The general refinement mechanism is shown in Figure 7 below.

Proving Liveness

Now that model behaviour is sufficiently constrained by the deadlines, we can prove liveness of the system. For each of the 12 valid states as shown in Figure 5 above we introduce an invariant to prove that at least one event is always enabled in that state. For instance, in the state *(locked_down, locked_open, DOWN)*, one of the events *Close* or *Retract* is always enabled. Similarly, in state *(locked_down, locked_open, UP)*, one of the events *Close* or *Retract* is also always enabled. We describe this using a single *theorem*.

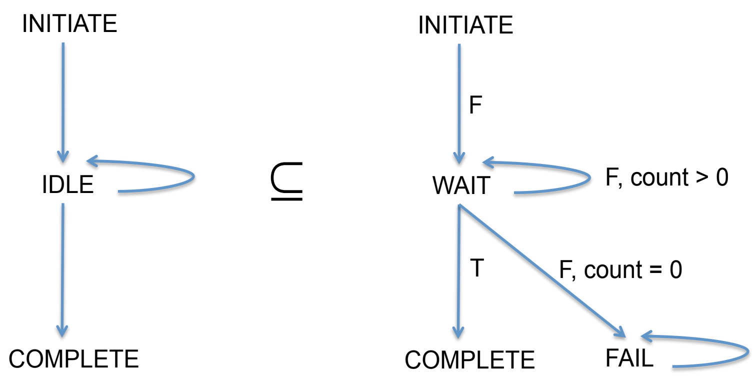

Figure 7: Introducing the Deadlines

We write the theorem as an implication, with the current state on the left hand side and the *disjunction of the guards* of the two enabled events on the right hand side as shown below in invariant *inv19*.

$$\begin{aligned}
\texttt{inv19} : (gearstate = locked\_down \wedge doorstate = locked\_open) \quad \Rightarrow \\
&(gearstate \in \{extending, locked\_down\} \wedge \\
&\ doorstate = locked\_open \wedge handle = UP) \quad \vee \\
&(gearstate = locked\_up \wedge \\
&\ doorstate \in \{opening, locked\_open\} \wedge handle = UP) \quad \vee \\
&(gearstate = locked\_down \wedge \\
&\ doorstate \in \{opening, locked\_open\} \wedge \\
&\ handle = DOWN)
\end{aligned}$$

Ten theorems describe the liveness properties of the system and are proved automatically by the Rodin theorem provers.

4 Measuring Coverage

The Modified Condition/Decision Coverage (MC/DC) measure introduced in ADVANCE can be used to verify, at every refinement level, that the guards of each event can be independently set to *FALSE*. The facility is also used at every refinement stage to ensure that any *vacuous* guards are eliminated.

5 Model-based Test Generation

We are now in a position to generate tests from the model that can be used to verify the implementation. Because, the pilot is free to move the handle backwards and forwards with complete freedom, we now need to constrain the model behaviour to ensure that we can generate a test suite of tractable size. We introduce, in a refinement a *ghost variable*, *handle_toggle_count* which limits the number of times the handle can change position. We can then restrict the value of this count in the ADVANCE test generation tool to limit the *scope* of the model search space.

For instance, restricting the count to two, the pilot can only move the handle twice and a total of 48 tests are generated. If the count is restricted to three, 852 tests are generated. In a structured regression suite mechanism, these tests sets can be run first to detect any gross errors. For more thorough testing, the tests generated for higher handle counts can be run - setting the count to 5, for instance, results in 15400 tests. The coverage obtained by each set of tests is measured by using the ProB simulator to run the generated tests against the model.

6 Decomposition

Formal, *shared event* decomposition is now used to separate the Controller model from its environment for further refinement towards implementation. The decomposed Controller model can also be converted into a Functional Mockup Unit (FMU) for simulation in a continuous representation of the landing gear environment.

References

[BW14] Frédéric Boniol and Virginie Wiels. The landing gear system case study. In *ABZ 2014: The Landing Gear Case Study*, pages 1–18. Springer, 2014.

[Lev12] N.G. Leveson. *Engineering a safer world: Systems thinking applied to safety.* MIT Press (MA), 2012.