Capturing Requirements by ASMs: The Light Control Case Study

Egon Börger **Elvinia Riccobene**

Joachim Schmid

J. Universal Computer Science

Special Issue on Requirements Engineering, Fall 2000

Executable version at

{http://www.uni-ulm.de/~s_jschmi/AsmGofer/light}

The Role of Ground Models

Goal: Turning informal description of requirements into a rigorous formulation: authoritative basis for design (binding contract bw customer and system designer).

Makes it necessary to offer the possibility to

- check correctness (adequacy) and completeness (no underspecification) of rigorous model wrt informal requirements (*user* constraints and scenarios)
- check internal consistency and intrinsic completeness of reqirements (in terms of the model)
- formulate test plan for acceptance of implementation
- avoid overspecification through abstraction from implementation details

Integration into Design Process

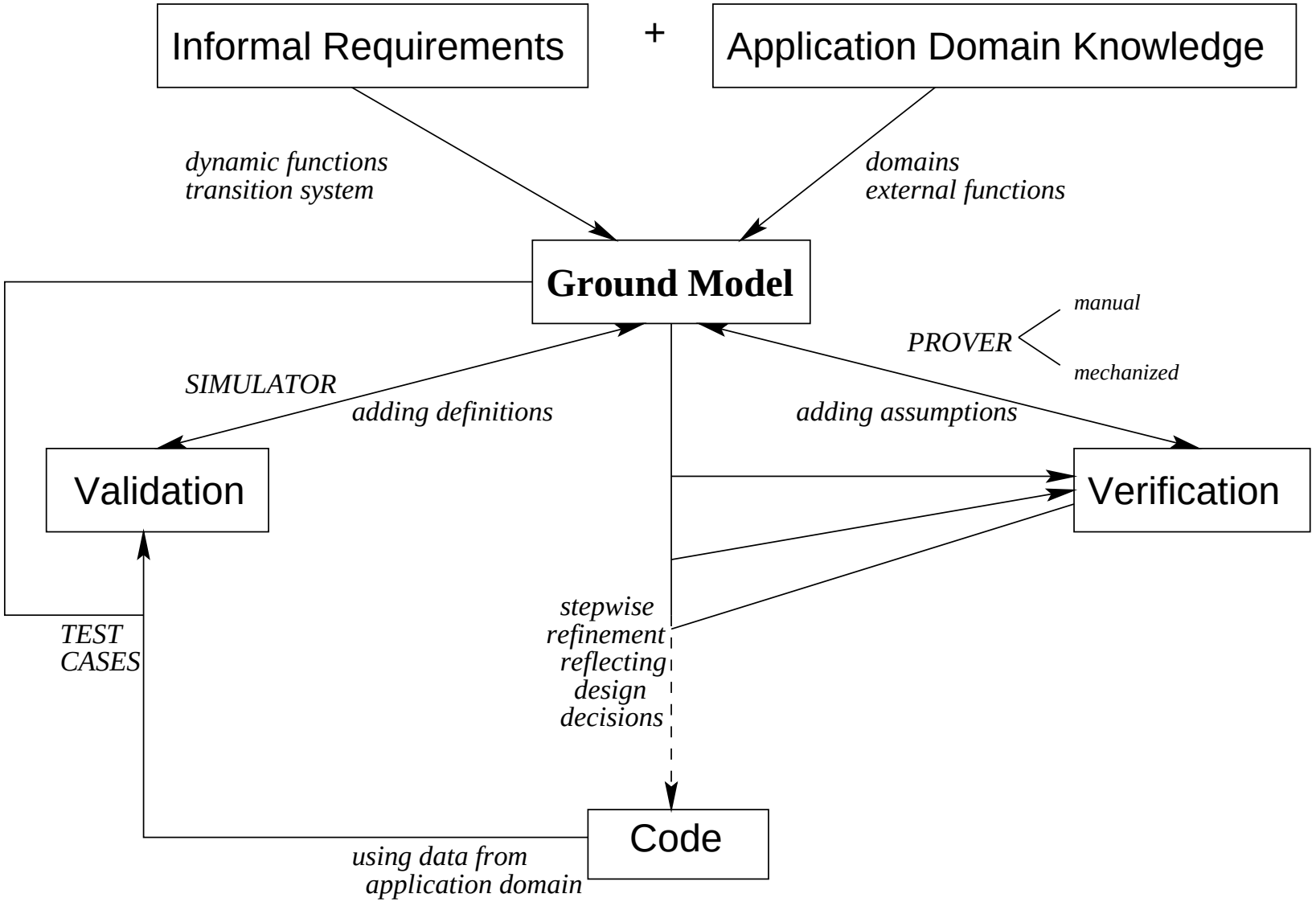

Figure 1: Integration Into Development Process

The Basic Objects

Rooms and hallway sections (Part 2, floor description) associated with

- light groups (window and wall ceiling) with actuators
  - buttons
  - dimmers
  - door closing contacts
- motion detectors (also in staircases) and light sensors
- status lines to report values of associated light groups

To structure the spec ("for change") we describe the system's actions as parameterized by these objects (oo view)

Modularization of the Requirements

Problem Description (Part 3) distinguishes user needs, facility manager needs, and fault tolerance. We group them as follows:

- reqs for *manual* interactions (Section 3.1.) of user/facility manager with control system, such as pressing buttons
- requs for *automatic* actions (triggered by the control system), such as switching off light in unoccupied rooms
- handling of *malfunctions* (Section 3.2.)

Three Submachines Forming the Ground Model

$$
\begin{aligned}
light = \; & manual\_light\_control \\
& automatic\_light\_control \\
& failure\_and\_service
\end{aligned}
$$

NB. This *parallel architectural structure* reflects that the Problem Description requires the Light Control to guarantee at each moment both forms of control together with the surveillance.

The ASM framework forces the designer to make those *assumptions* explicit which are needed to *guarantee* the *conflict-freeness* of competing actions (scheduling).

Scheduling the Ground Model Submachines

- making specific priority assumptions on possibly conflicting actions
- imposing a concrete coordination of the submachines

For executable version we assume that

- manual and automatic submachines alternate at a fixed rate, fast enough to guarantee the desired reaction time of the light system to user or environment input
- failure and service submachine is executed in between with a frequency which satisfies the time requirements for failure handling and general services

Parallelism of Light-Switching Agents

According to the Problem Description, at any time and simultaneously

- facility manager can switch off the lights in every location
- different users can push simultaneously
  - wall and window light group buttons in rooms
  - the control panel in rooms
  - buttons in hallways

Parallelism of Light-Switching Agents

$$
\begin{array}{l}
manual\_light\_control = \\
\quad \textbf{forall } location \in all\_locations \\
\quad\quad manually\_switch\_off\,(location) \\
\quad\quad \textbf{if } location\_is\_room(location) \textbf{ then} \\
\quad\quad\quad room\_wall\_button(location, LightGroupWall) \\
\quad\quad\quad room\_wall\_button(location, LightGroupWindow) \\
\quad\quad\quad control\_panel(location, switch(location)) \\
\quad\quad \textbf{if } \neg location\_is\_room(location) \textbf{ then} \\
\quad\quad\quad hallway\_button(location)
\end{array}
$$

Pushing Room Wall Buttons

Every room has two wall switches, one for the wall ceiling light group and the other for the window ceiling light group (Section 2.1-2.4, Paragraphs 7-11)

1. If the ceiling light group is completely on, it will be switched off (Section 2.1.2)
2. Otherwise it will be switched on completely

`PushButtonReq`: Event function *lightgroup_wall_button_pressed* becomes *True* when the corresponding button has been pressed and *False* when the rule fires whose guard contains this event function

Room Wall Button Rule

$room\_wall\_button(room, lightgroup) =$
if $lightgroup\_wall\_button\_pressed(room, lightgroup)$ **then**
  if $lightgroup\_is\_completely\_on(room, lightgroup)$ **then**
    $switch\_lightgroup\_off\,(room, lightgroup)$
  else
    $switch\_lightgroup\_completely\_on(room, lightgroup)$

Setting all lights in the corresponding room to *minDimValue* or *maxDimValue* resolves the apparent contradiction in U1, considering it as safe to allow a person who wants to rest in a room to choose a light scene in which all the lights are switched off and the room is dark `U1Req`.

Switching Lightgroups Off/Completely On

$mode : Room \rightarrow \{Manual, Ambient\}$ determines whether the room light was set by the user or by the control system

$$
\begin{array}{l}
switch\_lightgroup\_off\,(room, lightgroup) = \\
\quad mode(room) := Manual \\
\quad \textbf{forall}\ light \in lights\_in\_group(room, lightgroup) \\
\quad\quad switch\_light(room, light, minDimValue)
\end{array}
$$

$$
\begin{array}{l}
switch\_lightgroup\_completely\_on(room, lightgroup) = \\
\quad mode(room) := Manual \\
\quad \textbf{forall}\ light \in lights\_in\_group(room, lightgroup) \\
\quad\quad switch\_light(room, light, maxDimValue)
\end{array}
$$

Pushing Hallway Switch Buttons

- Hallway section buttons linked in parallel (Section 2.6.3)
- The light in the hallway section has to be on if one button is defective

`NF5aReq`: In `NF5` we interpret "not controllable manually", in view of the safety requirement `U1`, as meaning that at least one hallway button is defective (*any_hallway_button_defect*).

Pushing Hallway Buttons Rule

$$
\begin{array}{l}
hallway\_button(hallway) = \\
\quad \textbf{if } hallway\_button\_pressed(hallway) \wedge \\
\qquad \neg any\_hallway\_button\_defect(hallway) \textbf{ then} \\
\qquad \textbf{if } light\_is\_on(hallway) \textbf{ then} \\
\qquad\quad switch\_lights\_off(hallway) \\
\qquad \textbf{else} \\
\qquad\quad switch\_lights\_on(hallway)
\end{array}
$$

Switching On/Off Lights

Parameterization by locations (rooms or hallways) to reflect the similarity of the dictionary definition of "push button".

$$
\begin{array}{l}
switch\_lights\_off(location) = \\
\quad \textbf{forall } light \in lights\_at(location) \\
\quad\quad switch\_light(location, light, minDimValue) \\
\quad \textbf{if } location\_is\_room(location) \textbf{ then} \\
\quad\quad mode(location) := Manual
\end{array}
$$

$$
\begin{array}{l}
switch\_lights\_on(location) = \\
\quad \textbf{forall } light \in lights\_at(location) \\
\quad\quad switch\_light(location, light, maxDimValue)
\end{array}
$$

Manual Switching Off by Facility Manager

$$
\begin{aligned}
&\mathit{manually\_switch\_off}\,(\mathit{location}) = \\
&\quad \textbf{if } \mathit{manually\_switch\_off\_pressed}(\mathit{location}) \\
&\qquad\qquad \wedge \neg \mathit{occupied}(\mathit{location}) \textbf{ then} \\
&\qquad \mathit{switch\_lights\_off}\,(\mathit{location})
\end{aligned}
$$

- Problem 1: how to guarantee the consistency between user and facility manager light updates in *room_wall_button*, *hallway_button*, *manually_switch_off*, *control_panel*?
- Problem 2: how to determine occupation of a location (since motion sensors can sense only motion and not presence of someone quiet)?

Defining Room Occupation

- `RoomOccupationReq`: location not occupied if there has been no motion for a period of max_quiet_time
- `NF5bReq` (guaranteeing `NF4`): location occupied if at least one of its motion detectors does not work correctly
- `MotionDetectorReq`: motion sensor detects when users push buttons

$$occupied(location) =$$
$$current\_time - last\_motion(location) \leq max\_quiet\_time$$

$$observe\_motion\_detector(location) =$$
$$\textbf{if } somebody\_is\_moving(location) \textbf{ then}$$
$$last\_motion(location) := current\_time$$

User Action at Control Panel

Control panel (`U5,U6,U9`): activate ambient light scene and switch on/off ceiling lights

`PushButtonReq`: to guarantee that simultaneous pushing on wall buttons (by the user or the facility manager) and on the control panel does not produce effects which exclude each other, we assume that the hardware solves this conflict.

In a software solution:

- a fixed priority scheme results in additional mutually exclusive rule guards
- unspecified scheduling can be handled by selection functions

Control Panel Rule

$$
\begin{array}{l}
control\_panel(room, switch) = \\
\textbf{if } switch\_pressed(room, switch) \textbf{ then case } switch \textbf{ of} \\
AmbientSelection \rightarrow \\
\quad activate\_light\_scene(room, last\_light\_scene(room)) \\
LightGroup(lg) \rightarrow \\
\quad \textbf{case } switch\_value(room, switch) \textbf{ of} \\
\quad\quad On \rightarrow switch\_lightgroup\_completely\_on(room, lg) \\
\quad\quad Off \rightarrow switch\_lightgroup\_off(room, lg) \\
SceneSelection \rightarrow \\
\quad \textbf{case } switch\_value(room, switch) \textbf{ of} \\
\quad\quad Scene(s) \rightarrow set\_light\_scene(room, s)
\end{array}
$$

Ambient Light Scene Activation Requirements

- Requirement 2.10, Paragraph 19: light scene contains an *ambient light level* and an ordered list of lights together with a dim value for each light
- Dictionary: control system has to switch on the lights in the given order with the corresponding dim value
- `FM1`: take into account the ambient light from outside

LightSceneReq: Let *lights_to_turn_on* compute an ordered set containing all lights that should be switched on in this order, together with their dim values, taking into account malfunctioning lights (makes defn of light scene uniform and easily changable)

Ambient Light Scene Activation Rule

$$
\begin{array}{l}
activate\_light\_scene(room, scene) = \\
\quad mode(room) := Ambient \\
\quad last\_light\_scene(room) := scene \\
\quad \textbf{if } scene = default\_light\_scene(room) \wedge \\
\qquad outdoor\_light\_sensor\_defect(room) \textbf{ then} \\
\qquad\quad switch\_lights\_on(room) \textbf{ else} \\
\qquad \textbf{let } lights\_on = \\
\qquad\quad lights\_to\_turn\_on(room, outdoor\_sensor(room), scene) \\
\qquad \textbf{forall } (light, value) \in lights\_on \\
\qquad\quad switch\_light(room, light, value) \\
\qquad \textbf{forall } light \in lights\_at(room)\{l \mid (l, v) \in lights\_on\} \\
\qquad\quad switch\_light(room, light, minDimValue)
\end{array}
$$

Ambient Light Scene Activation Satisfies Requirements

`NF2`: If any outdoor light sensor does not work correctly, the default light scene for all rooms is that both ceiling light groups are on.

OutdoorSensorReq: value of $outdoor\_sensor(room)$ remains constant if the sensor does not work correctly.

U10Req: we interprete *to maintain the ceiling light group on a given light scene* to mean that the ceiling lights are set to *minDimValue* if they do not enter explicitly the lights to be turned on for the given light scene.

Setting Light Scene

$$
\begin{array}{l}
set\_light\_scene(room, scene) = \\
\quad \textbf{if } mode(room) = Ambient \textbf{ then} \\
\quad\quad activate\_light\_scene(room, scene) \\
\quad \textbf{else} \\
\quad\quad last\_light\_scene(room) := scene
\end{array}
$$

Automatic Light Control

Observing motion detectors and switching off/on the light in rooms (using also daylight) and hallways

$$
\begin{array}{l}
automatic\_light\_control = \\
\quad \textbf{forall } location \in all\_locations \\
\quad\quad observe\_motion\_detector(location) \\
\quad\quad auto\_switch\_off(location) \\
\quad\quad \textbf{if } location\_is\_room(location) \textbf{ then} \\
\quad\quad\quad auto\_switch\_on\_in\_room(location) \\
\quad\quad\quad use\_daylight(location) \\
\quad\quad \textbf{if } \neg location\_is\_room(location) \textbf{ then} \\
\quad\quad\quad auto\_switch\_on\_in\_hallway(location)
\end{array}
$$

Automatic Switching Off Requirements

- `FM2`, `FM3`: switch off the light in a location if it is not occupied for T3 or T2 minutes
- malfunction condition `NF5`: do not switch off the light in a hallway section if one of its buttons is defective
- malfunction condition `NF4`: occurrence of a malfunction for a motion sensor is interpreted as presence of motion so that the location appears as occupied

Automatic Switching Off Rule

$$
\begin{array}{l}
auto\_switch\_off\,(location) = \\
\quad \textbf{if } location\_is\_hallway(location) \wedge \\
\qquad any\_hallway\_button\_defect(location) \\
\qquad \textbf{then } skip \\
\qquad \textbf{else} \\
\qquad\quad \textbf{if } \neg occupied(location) \wedge \\
\qquad\qquad no\_motion\_for\_long\_time(location) \wedge \\
\qquad\qquad \neg some\_door\_is\_open(location) \wedge \\
\qquad\qquad \neg light\_is\_off\,(location) \\
\qquad\quad \textbf{then } switch\_lights\_off\,(location)
\end{array}
$$

Defining "No Motion for Long Time"

$$
\begin{aligned}
&no\_motion\_for\_long\_time(location) = \\
&\quad \textbf{if } location\_is\_room(location) \textbf{ then} \\
&\qquad current\_time - last\_motion(location) > t_3(location) \\
&\quad \textbf{else} \\
&\qquad current\_time - last\_motion(location) > t_2(location)
\end{aligned}
$$

Automatic Switching On in Hallways

Hallway lights are not dimmable so that switching on can be done only completely. Switching on triggered by motion or open doors.

$$
\begin{aligned}
&\mathit{auto\_switch\_on\_in\_hallway}(\mathit{hallway}) = \\
&\quad \textbf{if } (\mathit{somebody\_is\_moving}(\mathit{hallway}) \vee \\
&\qquad \mathit{some\_door\_is\_open}(\mathit{hallway})) \wedge \\
&\qquad \mathit{light\_is\_off}\,(\mathit{hallway}) \\
&\quad \textbf{then} \\
&\qquad \mathit{switch\_lights\_on}(\mathit{hallway})
\end{aligned}
$$

Automatic Switching On in Rooms: Requirements

`U3` If the room is reoccupied within T1 minutes after the last person has left the room, the *chosen light scene* has to be reestablished.

`U4` If the room is reoccupied after more than T1 minutes since the last person has left the room, the *default light scene* has to be established.

U3Req: instead of establishing the *chosen light scene* we use the *last light scene* to avoid incoherence of the requirements (see counterexample).

Automatic Switching On in Rooms: Counterexample

1. Person A enters the room and selects scene s.
2. Person A leaves the room for more than T1 minutes.
3. Person B enters the room. According to `U4`, the *default light scene* should be established.
4. Person B does not change the light scene and leaves the room.
5. Person B enters within T1 minutes. According to `U3`, we should establish the *chosen light scene* s.

Person B, upon entering the room for the first time, gets the *default light scene* and, upon reentering, gets the *chosen light scene* of Person A.

Automatic Switching On in Rooms: Rule

$$
\begin{aligned}
&\mathit{auto\_switch\_on\_in\_room}(\mathit{room}) = \\
&\quad \textbf{if } (\mathit{somebody\_is\_moving}(\mathit{room}) \vee \\
&\qquad \mathit{some\_door\_is\_open}(\mathit{room})) \wedge \mathit{light\_is\_off}(\mathit{room}) \\
&\quad \textbf{then} \\
&\qquad \mathit{activate\_light\_scene}(\mathit{room}, \mathit{scene}) \\
&\quad \textbf{where } \mathit{scene} = \textbf{if } \mathit{recently\_occupied}(\mathit{room}) \textbf{ then} \\
&\qquad\qquad\qquad\qquad \mathit{last\_light\_scene}(\mathit{room}) \\
&\qquad\qquad\qquad\quad \textbf{else } \mathit{default\_light\_scene}(\mathit{room}) \\
&\mathit{recently\_occupied}(\mathit{room}) = \\
&\quad \mathit{current\_time} - \mathit{last\_motion}(\mathit{room}) \leq t_1(\mathit{room})
\end{aligned}
$$

We leave defn of *default_light_scene* open

Using Daylight for Ambient Light Level

FM1 requires to use daylight to achieve *ambient light level*. We reactivate the current light scene if the room is in ambient mode and there is no request for the ceiling lights.

$$
\begin{array}{l}
use\_daylight(room) = \\
\quad \textbf{if } no\_event\_for\_ceiling\_light(room) \wedge \\
\qquad mode(room) = Ambient \textbf{ then} \\
\qquad activate\_light\_scene(room, last\_light\_scene(room))
\end{array}
$$

The rule also reflects the informal need U10.

Failure and Service Machine

Handling of malfunctions (identifying and handling),
detecting unreasonable inputs, reporting energy consumption,
setting parameters (reflecting `FM11`, `FM4`, `FM5`, `U7`)

$$
\begin{aligned}
&\mathit{failure\_and\_service} = \\
&\quad \mathit{malfunction} \\
&\quad \mathit{detect\_unresonable\_input} \\
&\quad \mathit{report\_energy\_consumption} \\
&\quad \mathit{set\_parameters}
\end{aligned}
$$

Identifying Malfunctions and Unreasonable Input

The Problem Description does not provide any further details.

malfunction_occurs and *detect_unresonable_input* machine have to be defined by application domain experts.

Due to requirement `FM1` *malfunction_occurs* should be updatable manually.

$$
\begin{array}{l}
\mathit{malfunction} = \\
\quad \textbf{forall } \mathit{component} \in \mathit{all\_components} \\
\quad\quad \textbf{if } \mathit{malfunction\_occurs}(\mathit{component}) \textbf{ then} \\
\quad\quad\quad \mathit{handle\_malfunction}(\mathit{component})
\end{array}
$$

Components are hallway buttons, light sensors, motion sensors or any light.

Handling Malfunctions

- According to `U8`, `FM7`, and `FM10`, the handling of malfunction logs the corresponding information
- By `NF5`, when a hallway button is defective we switch on the lights in that hallway
- When hallway motion detector is defective, by assumption (NF5bReq) *somebody_is_moving* is true and we therefore switch on the lights by rule *auto_switch_on_in_hallway*

Handling Malfunctions Rule

$$
\begin{array}{l}
\mathit{handle\_malfunction}(\mathit{component}) = \textbf{case } \mathit{component} \textbf{ of} \\
\mathit{OutdoorLightSensor}(i) \rightarrow \\
\quad \textbf{forall } \mathit{room} \in \mathit{rooms\_under\_lightsensor}(i) \\
\quad\quad \mathit{inform\_user}(\mathit{room}, \mathit{LightSensorDefect}(i)) \\
\quad\quad \mathit{inform\_facility\_manager}(\mathit{LightSensorDefect}(i)) \\
\quad\quad \mathit{write\_log\_in\_database}(\mathit{LightSensorDefect}(i)) \\
\mathit{MotionSensor}(\mathit{location}, i) \rightarrow \\
\quad \textbf{if } \mathit{location\_is\_room}(\mathit{location}) \textbf{ then} \\
\quad\quad \mathit{inform\_user}(\mathit{location}, \mathit{MotionSensorDefect}(\mathit{location}, i)) \\
\quad \mathit{inform\_facility\_manager}(\mathit{MotionSensorDefect}(\mathit{location}, i)) \\
\quad \mathit{write\_log\_in\_database}(\mathit{MotionSensorDefect}(\mathit{location}, i))
\end{array}
$$

Handling Malfunctions Rule (Cont'd)

$handle\_malfunction(component) = \textbf{case}\ component\ \textbf{of}$
$HallwayButton(hallway, i) \rightarrow$
  $switch\_lights\_on(hallway)$
  $inform\_facility\_manager(HallwayButtonDefect(hallway, i))$
  $write\_log\_in\_database(HallwayButtonDefect(hallway, i))$
$Luminaire(location, light) \rightarrow$
  $inform\_facility\_manager(LightDefect(location, light))$
  $write\_log\_in\_database(LightDefect(location, light))$

Reqmt FM9: Dim Values for Power Consumption Report

dim_value stores the current dim value of a light. Switching the light is defined as setting a dim value.

If the dim value is less than 10 then the light is switched off (see Problem Description, Table 2):

$$
\begin{array}{l}
switch\_light(location, light, value) = \\
\quad dim\_value(location, light) := value \\
\quad \textbf{if } value < 10 \textbf{ then} \\
\quad\quad status\_of\_light(location, light) := Light\_Off \\
\quad \textbf{else} \\
\quad\quad status\_of\_light(location, light) := Light\_On
\end{array}
$$

Reporting Power Consumption

The energy consumption is the integral of the power consumption over the time. We store the power consumption in each step and define the energy consumption as the product of the interval t_e with the sum of the power consumptions.

The following rule is assumed to be executed every t_e minutes.

$$
\begin{aligned}
&\mathit{report\_energy\_consumption} = \\
&\mathit{consumption}(\mathit{current\_time}) && := \mathit{power\_consumption} \\
&\mathit{energy\_consumption}(\mathit{current\_time}) && := \\
& && t_e * \sum_t \mathit{consumption}(t)
\end{aligned}
$$

Computing Power Consumption

$$power\_consumption = \sum [p(l, dim\_value(l)) \mid l \in dom(dim\_value)]$$

$$\textbf{where } p(l, v) = \textbf{case } light\_defect(l) \textbf{ of}$$

$$NotDefect \rightarrow c * v$$

$$DefectOn \rightarrow c * maxDimValue$$

$$DefectOff \rightarrow c * minDimValue$$

Ground Model Refinement for ASMGofer Execution

Standard data structures to encode locations, lights, light groups, actuators, light sensor, etc.

$$inform\_user(room, malfunction) =$$
$$user\_information(room, current\_time, malfunction) := True$$

$$inform\_facility\_manager(malfunction) =$$
$$facility\_manager(malfunction, current\_time) := True$$

$$write\_log\_in\_database(malfunction) =$$
$$database(current\_time, malfunction) := True$$

Examples of derived functions in ASMGofer

$$
\begin{aligned}
&lightgroup\_is\_completely\_on(room, lg) = \\
&\quad \forall l \in lights\_in\_group(room, lg) \\
&\qquad dim\_value(room, l) = maxDimValue
\end{aligned}
$$

$$
\begin{aligned}
&light\_is\_on(hallway) = \\
&\quad \forall l \in lights\_at(hallway) \\
&\qquad status\_of\_light(hallway, l) = Light\_On
\end{aligned}
$$

$$
\begin{aligned}
&light\_is\_off(location) = \\
&\quad \forall l \in lights\_at(location) \\
&\qquad status\_of\_light(location, l) = Light\_Off
\end{aligned}
$$

Executable version available at

{http://www.uni-ulm.de/~s_jschmi/AsmGofer/light}

Questionnaire:

- all requirements (including real-time aspects) elicitated, specified, implemented in ASMGofer
- inconsistencies removed and implicit assumptions made explicit
- spec is reusable (via parameterization/modularization) and requs are traceable in spec
- ASMGofer code is compilable to C++ code
- effort: 14 days (à 8 hrs)