The CIDS A330/340 Cabin Communication System – A Z Application

This article is an extended version of U. Hamer and J. Peleska: Z Applied to the A330/340 CIDS Cabin Communication System. In M. Hinchey, J. Bowen (Eds.) *Applications of Formal Methods.* Prentice-Hall International, Englewood Cliffs NJ (1995).

The CIDS A330/340 Cabin Communication System – A Z Application

Ute Hamer* Jan Peleska†

August 15, 1995

Abstract

We give a formal specification of the Airbus A330/340 cabin illumination system using the specification language Z. The cabin illumination system is an application function of the Cabin Intercommunication Data System (CIDS) that contains all cabin communication functions in the Airbus family A330/340. In order to allow the reader to re-trace our specification decisions how to transform the application's natural-language description into Z, the original system requirements specification document [1] has been included in this article with courtesy of Deutsche Aerospace Airbus GmbH. Furthermore, the transformation of the Z specification into software design implementable in the target environment is illustrated. This transformation makes use of the Z specification's mathematical rigor in combination with heuristic design techniques of conventional software engineering.

1 Introduction

The CIDS System The *Cabin Intercommunication Data System (CIDS)* is an airborne fault-tolerant dual computer system to control the cabin communication functions implemented in the Airbus family A330/340. It comprises applications such as the public address system, cabin interphone system, sign operation (e. g. fasten seatbelts, no smoking), display of sensor information for

*DST Deutsche System-Technik GmbH, Edisonstraße 3, D-24125 Kiel, Germany, FAX +49 431 7109503

†JP Software-Consulting, Goethestraße 26, D-24116 Kiel, Germany, Fax +49 431 552892, e-mail: jap@informatik.uni-kiel.d400.de

the cabin crew (e. g. door and slide sensors), emergency evacuation signalling, testing of the emergency power supply unit and control of the cabin illumination. Amongst other reasons, the necessity to implement this heterogeneous set of functions on one computer system, is given by the strict space limitations that have to be observed in the design of airborne computer systems. The CIDS system does not directly control safety-critical aircraft components, for example, the engines. However, since CIDS implements applications where a malfunction might indirectly impair human lives (e. g. emergency call function of the interphone system, emergency evacuation signalling), it has been classified according to DO178A criticality level 2b (this is the second-highest criticality level and corresponds to level B in the new revision DO178B [3]).

As a subcontractor to Deutsche Aerospace Airbus GmbH, DST have developed most of the application layer and some parts of the operating system software[1]. The Deutsche Aerospace Airbus GmbH have defined the CIDS system requirements, developed the operating system software and the computer hardware. The software development process was strictly controlled according to the requirements of DO178A. Throughout the software requirements specification phase and the software design phase, *Structured Methods* (Structured Analysis, Real-Time Analysis, Structured Design according to [4, 12] were applied with the support of a commercially available CASE tool. During the specification process, several of the application functions turned out to be surprisingly complex. The CASE methods applied completely failed to provide a sufficient means to overcome this complexity and to produce trustworthy specification results. Therefore DST decided to apply formal specification techniques using the Z notation [11, 13] as an extension of the CASE specifications for the most critical and complex application functions.

CIDS Cabin Illumination This article presents one of these formally specified functions, the *CIDS Cabin Illumination Function (CIL)*. It will be described in detail during the subsequent sections. To introduce the basic concept of the CIL functionality, consider the aircraft cabin layout shown in Figure 1. The CIL application allows separate control for the illumination of cabin zones (e. g. first class, business class), entry areas and crew rest compartments (CRC).

To change the illumination status, the cabin crew uses panels as shown in Figure 2. The commands for cabin zones allow to choose the illumination level separately for each zone with intensities *dim2, dim1, bright*. If the illumination level in zone 1 is *off* and the user presses button *dim1*, the illumination units will be switched to this intensity throughout the zone. This will be acknowledged to the user by activating a button indicator light in the *dim1*-button of zone 1. Pressing

[1] Both authors have been members of the project team, the second author in the position of the DST project manager.

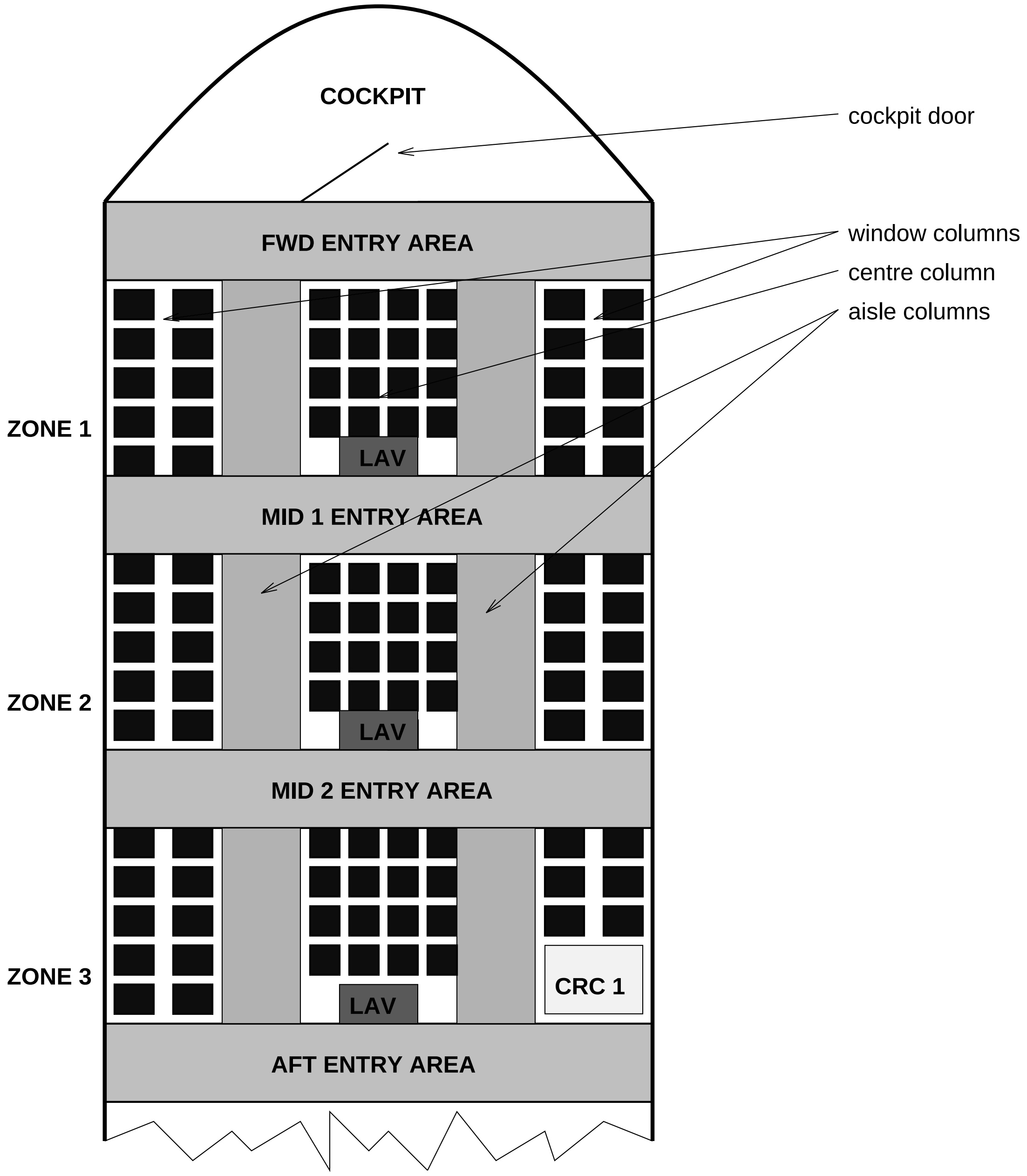

Figure 1: Aircraft cabin layout (simplified). Only three cabin zones are shown.

the *bright*-button of zone 1, will change the intensity from *dim1* to *bright*, switch off the *dim1* button indicator and turn on the *bright* button indicator. Pressing the *bright* button a second time will switch illumination and button indicators of zone 1 off. In addition to the zone commands, the *window*, *centre*, *aisles* buttons allow to switch illumination for the respective seat row columns/aisles of all zones. Commands to switch on/off night lights can be given, if the illumination of one zone has otherwise been switched off.

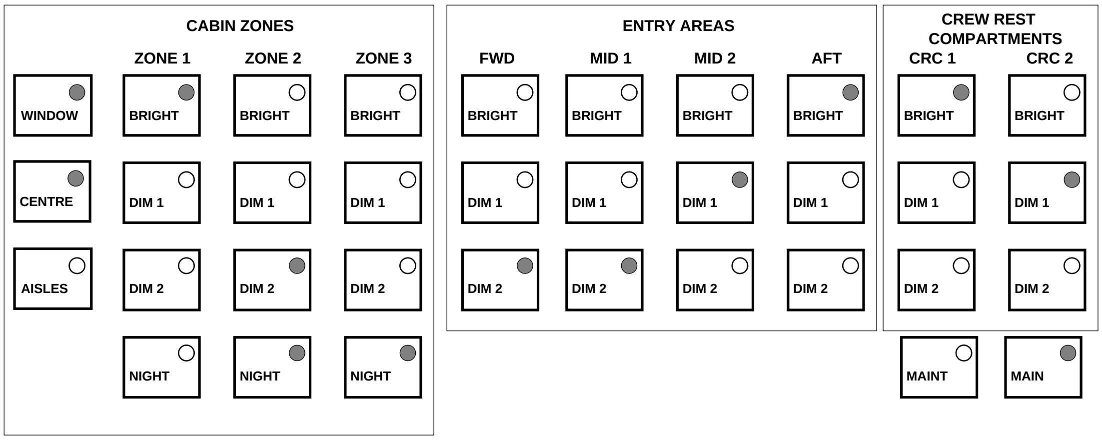

Figure 2: Forward Attendant Panel (FAP) — lights module

Similarly, the user can control the illumination status for entry areas and crew rest compartments. In addition to the user commands, certain sensors have impact on the level of illumination. For example, if the cockpit door sensor signals 'door open', the forward entry area lights must be dimmed to avoid distraction of the cockpit crew.

To allow a flexible software-controlled adaption of the functions to different cabin layouts (e. g. different numbers of seat rows in each zone), the illumination units are connected to a bus used by the CIDS system to control various peripherals in the cabin. The CIL application function switches the intensity of an illumination unit by sending a command to the unit's bus address. CIDS configuration data defines the mapping of bus addresses to cabin locations. Moreover, the functionality can be configured by means of 'software switches' that enable or disable certain CIL features.

Overview Section 2 presents the original system requirements specification for the cabin illumination functions. Section 3 justifies the suitability of Z for the application and presents the Z specification derived from system requirements. In Section 4, we will sketch the transformation process from specification to design, with a focus on the derivation of *software architecture design*. The heuristic approach presented in this section has been successfully applied at DST to several projects.

2 CIDS Airbus A330/340 Cabin Illumination – Informal Requirements Specification

In this section the requirements [1] for the CIDS cabin illumination function (CIL) are presented. Explanations of technical terms used in the requirements document and other information that might be helpful for the reader who is not acquainted with the CIDS system have been added as footnotes. Otherwise the text has been left in its original form.

2.1 General

This system function controls the dimmable illumination of different areas independently. Control information shall be received from the FAP, the MPB, AAPs[2] and some discrete inputs. The system software shall control the respective ballast units[3] in the cabin via the top lines and via DEU, type A[4] respectively.

The function of the cabin illumination shall be divided into at most 4 entry, 2 crew rest and 5 cabin zones[5], which are defined in number and configuration in the CAM[6] and which shall be controlled individually. The illumination cabin zones are split into window, centre and aisle columns. Each of those zones are controlled by three commands from the FAP, AAPs, MPB and some discrete input commands. Their implication and effects are listed below.

- BRIGHT/off : Alteration between 100% and off

[2]Forward Attendant Panel (FAP), Multi Purpose Bus (MPB) and Additional Attendant Panels (AAP) provide user interfaces to input CIL commands and to observe system response via indicators. For the CIL application, these interfaces are identical. Figure 2 sketches the design of an FAP. The MPB is a computer interface that allows to control the functions normally available on panels from a remote computer.

[3]i. e. illumination units

[4]The *top line* is a bus connected to the CIDS system. This bus runs through the whole cabin. Its purpose is to provide interfaces to peripherals (e. g. illumination units) controlled by the CIDS. To allow control of different types of peripherals, a standard bus interface, the Decoder/Encoder Unit (DEU) is used. Each illumination unit can be controlled by the CIDS computer by sending a command to the DEU address where the illumination unit is connected.

[5]ref. Figure 1

[6]The Cabin Assignment Module (CAM) stores data for the configuration of the CIDS system. These data describe the cabin layout (e. g. a mapping of DEUA addresses to entry areas, crew rest compartments and cabin zones) and enables/disables CIDS functions by means of "software switches".

- DIM I /off : Alteration between 50% and off
- DIM II/off : Alteration between 10% and off

Generally the first activation of a command switches on the illumination with respective indications on the control panels, the second activation switches then off. The illumination status shall be transmitted on the CIDS MPB[7].

All functions which are controlled from the FAP shall also be operable from the AAPs or via the MPB. The functions and the presence of these panels shall be defined in the CAM.

2.1.1 Signal/Command Sources and Sinks

Sources The different command sources are received from:

- **units interfaced directly to the Director[8]:**
  - Forward attendant panel (Lights module[9])
  - Multipurpose bus (MPB)
  - Cabin pressure controllers (CPC 1+2)[10]
  - Cockpit door switch[11]
  - Landing gear control and interface unit (LGCIU)[12]
  - Oil pressure switch (EIUs)[13]
- **units interfaced via DEU, type B, and the middle line data buses[14], respectively, to the Director**
  - Additional attendant panels

[7] The status information transmitted to the MPB is in one-to-one correspondence with the status of the button indicators on the panels.

[8] The CIDS computer is called Director

[9] The FAP also carries other buttons to control additional CIDS functions apart from cabin illumination. These buttons have not been shown in Figure 2.

[10] CPC1 and CPC2 are sensor interfaces which indicate, whether the cabin pressure is ok (i. e. high).

[11] Another sensor indicates, whether the cockpit door is open.

[12] This interface provides the actual state of the landing gears. This is evaluated to decide whether the aircraft is on ground (state *down&compressed*) or in the air (state *up&locked* or *down&locked*).

[13] The state of the oil pressure switch is evaluated to determine whether the engine is running (oil pressure = *high*).

[14] The *middle line* is a bus installed in the cabin. It is similar to the top line described above, but interfaces to other types of peripherals (e. g. AAPs).

Sinks The different command sinks are:

- **units interfaced to DEU A:**
  - Illumination ballast units

2.1.2 Control of Illumination Ballast Units

The control signals for the operation of adjustable ballast units are as follows:

	Cabin and entry lights		
	DIM I	DIM II	OFF
DEU A Conn. J2; Pins	1/4/7/10/ 13/16/19/22	2/5/8/11 14/17/20/23	3/6/9/12 15/18/21/24
MODE	Signal		
BRIGHT 100%	0	0	0
DIM I 50 %	1	0	0
DIM II 10 %	0	1	0
OFF	0	0	1

0 = open

1 = 28V

The assignments to the respective areas, zones and zone splits is CAM-related.

For lavatory lights the outputs for DIM I/BRIGHT and OFF control will be used (see MAINTENANCE command).

	Lavatory lights		
	DIM I	DIM II	OFF
DEU A Conn. J2; Pins	1/4/7/10/ 13/16/19/22	2/5/8/11 14/17/20/23	3/6/9/12 15/18/21/24
COMMAND	Signal		
MAINTENANCE	0	-	0
LAV. ON	1	-	0
LAV. OFF/ MAIN OFF	0	-	1

2.1.3 Functions

MAIN ON Command The MAIN ON function shall be active on ground only (landing gear compressed). When the MAIN ON command is given or when CIDS is switched on, the illumination of the cabin, entries and crew rests shall switch to bright with respective indications on all panels. The illumination in the lavatories shall be switched on and dimmed (DIM I Status, see MAINT.command). The night lights shall switch off. When the MAIN OFF command is given the illumination of the cabin, entries, crew rests, reading lights, attendant work lights and lavatories shall be switched off[15]. The night lights shall also switch off. A MAIN ON command shall then switch on the cabin illumination. If the system is in the MAIN OFF status and any illumination zone/area is switched on manually, the status shall change to MAIN ON without switching on other lights (except lavatories, ref. MAIN ON command).

BRIGHT/ DIM I/ DIM II Command When BRIGHT, DIM I or DIM II command is given, all lights in the respective zone shall switch to the selected intensity with respective indications on the FAP and AAPs. When above mentioned commands are given, the night lights shall switch off in the respective zone.

WINDOW/ CENTRE/ AISLE Command Only if any of the cabin zones is switched to BRIGHT, DIM I or DIM II shall the WINDOW, CENTRE and AISLE command be carried out. The WINDOW, CENTRE and AISLE command affects the illumination of all cabin zones. When WINDOW, CENTRE or AISLE command is given, all light strips in the respective column shall switch off/on. If the previously given command is repeated, the illuminations shall switch to their previously selected intensities in the respective zones. When WINDOW AND CENTRE AND AISLE 'off' command is given, the illumination in all cabin zones shall switch off. The indications on FAP and AAP shall then also go off[16].

ENTRY AREA Commands When ENTRY BRIGHT, DIM I or DIM II command is given, all lights in the respective areas shall switch to the selected

[15] Note that reading lights and attendant work lights are controlled by separate applications. Therefore they will not be mentioned in the Z specification shown in the next chapter.

[16] Note that the Window/Centre/Aisle command has 'priority' over Bright/Dim I/Dim II commands: This means, that the illumination units associated with a column in state *off* will also remain in that state, if the intensity of the associated zone is changed. The units will only be switched on by re-activating the column or performing a MAIN OFF $\longrightarrow$ MAIN ON state transition.

intensity with respective indications on the FAP and AAPs.

Automatic Dimming Function of Entry Areas This function shall be carried out only, if in the respective area the light was switched on to BRIGHT, DIM I or DIM II. The automatic dimming function of the FWD entry area shall be controlled by two discrete inputs COCKPIT DOOR OPEN and OIL PRESSURE LOW. If COCKPIT DOOR OPEN is active and OIL PRESSURE LOW is not active, the illumination in the assigned area shall switch to a CAM-related intensity. This shall only be performed for CAM assigned ballast units.

CREW REST Command When CREW REST BRIGHT, DIM I or DIM II command is given, all lights in the respective areas shall switch to the selected intensity with respective indications on the FAP and AAPs.

MAINTENANCE Command The illumination in the lavatories shall be dimmed as long as the MAINTENANCE command is not given. When the MAINTENANCE command is given, these lights shall be switched to bright (see also MAIN ON for this function).

Night Light Function The basic CIDS shall provide two different night light functions named solution one and two. With solution I, the CAM-related illumination ballast units will be used for this function. Solution II describes the use of additional 28V lights connected to DEU, type A.

If a night light command is given a second time, the illumination will be switched off completely in the respective zone(s). This function is valid for solution I and II.

Solution I When the general illumination in a cabin zone is switched off completely, the CAM-assigned ports shall automatically switch to a CAM-assigned intensity in this zone (night light auto function)[17]. When a NIGHT LIGHT command is given (e.g. from the FAP), the CAM-assigned night lights shall switch on, if the illumination in the respective zone is switched off completely[18]. It shall be possible to operate the night lights zonewise. The operation of the cabin zones with BRIGHT, DIM I or DIM II overrides the night light function in the respective cabin zone.

[17]The night light auto function must be enabled in CAM.

[18]If the illumination in the respective zone is active and the night light auto function is enabled, pressing the night light button will not have any effect. When the night light auto function is disabled pressing the night light button will result in an activation of the corresponding button indicator. This serves as a night light pre-selection: Switching off the illumination will now automatically activate the night lights, as long as the pre-selection is active.

Solution II Functions of solution II are identical to solution I, but instead of the illumination ballast units, additional 28V night lights, connected to DEU, type A, will be used.

Cabin Decompression Function In the event of cabin decompression, the illumination in the cabin, entries, lavatories and crew rest areas shall switch to bright. For that, the input signals from the CPCs shall be monitored (CPC 1+2). If input one OR two is active the above mentioned lights shall switch on. When both inputs signals are deactivated then the illumination must keep the described switch status. During all phases, the operation of the illumination system shall be possible (e.g. FAP). If assigned in the CAM, the operation shall be blocked as long as one of the input signals is active[19].

FAP Inop Reaction If the FAP fails and this failure is detected by the system's internal BITE, the illumination which is controllable from the FAP shall switch to BRIGHT. The operation of the illumination shall then not be blocked for all other command sources.

3 Z Software Requirements Specification

3.1 Implementation Considerations Influencing the Specification

When selecting an appropriate specification language, it is necessary to take certain implementation considerations into account, to make sure, that the intended system behaviour is properly reflected by the semantics of the specification. For the CIDS system, the following boundary conditions influenced our decision to choose the Z notation to model the behaviour of applications:

- The underlying operating system layer schedules each application sequentially by means of a function call inside a single task continuously 'looping' on a processor. We refer to this sequence of application function calls as the *CIDS application cycle.*
- The number of functions and their corresponding resources (e. g. memory) are statically allocated on system startup. No dynamic process or memory allocation is performed.

[19] Note that the cabin decompression function is one of the safety-critical CIDS features that motivated the 2b-classification according to DO178A.

- The CIDS director is designed as a fault-tolerant redundant master-standby system consisting of two processors with local memory, bus system, interfaces etc. However, this redundancy is 'invisible' on the application level. Each application function can be designed as if it were running on a single-processor system.

- Interface drivers operate in parallel to the CIDS application cycle, but do not write concurrently onto memory areas which are accessed by applications.

- At the beginning of each CIDS application cycle, the data which have been accumulated by the input drivers, are made available by copying the information into memory areas being readable by applications. During that time, no application function is active.

These considerations justify the decision to consider the CIDS application layer as a *sequential* system. The system's static structure implies that it is not necessary to use a specification language allowing management of dynamic objects, as it would, for example, be necessary for specifying an X-windows user interface. Therefore the Z notation is a proper candidate to specify CIDS applications.

The Z specification to be presented in the following sections was written with two main objectives in mind:

- **Easy traceability to system requirements:** It should be "sufficiently" obvious to re-trace the system's behaviour[20], as pre-planned in the system requirements, in the Z software requirements specification. This is of special importance, since the informal nature of the system requirements document does not allow any sort of proof which shows the Z specification's completeness with respect to system requirements.

- **Easy transformation from software requirements to software design:** The Z specification should also be developed in a way that does not require more than one transformation or refinement step to reach an implementable level of abstraction.

These objectives in mind, less emphasis was made to reach other quality criteria of specifications, e. g. re-usability or the elegance which can be achieved by trying to find the weakest predicates still completely describing the desired functionality.

[20]In this context behaviour means the interfaces of the application, the functions and the data structures influencing the functionality.

3.2 Basic Data Items – Types and Sets

This section formally introduces the basic data items of the Cabin Illumination (CIL) software requirements specification: These items are *addresses* of DEUs connecting the CIDS bus to illumination units, *location identifiers* denoting specific locations inside the cabin, and *state values* of peripherals connected to the CIDS director which are relevant to the CIL specification.

Let $ADDRESS$ be the set of all DEU-A addresses combined with their ports where any illumination unit is attached to. At our level of abstraction, the distinction between DEU-address and port-number as well as the specific address range is irrelevant; we only need to know that any illumination unit can be activated by means of a unique address. We use $a \in ADDRESS$ as the (unique) identifier for an illumination unit.

$[ADDRESS]$

The following items introduce location identifiers. $LOCATION$ is the free type containing *all* identifiers: $z1, \ldots, z5$ denote the *cabin zones* (like "first class", "business class" etc.); fwd, $mid1$, $mid2$, aft are the names of *cabin entries*, $crc1$, $crc2$ are the *crew rest compartments*. Since all lavatory lights are synchronously controlled by each corresponding cabin illumination command, there is only one identifier lav to denote their locations. The *window* area denotes the two window seat row columns at both sides of the cabin, the *centre* area denotes the seat row column in the middle of the cabin, the *aisle* area denotes the two aisle spaces between centre and window seat rows. Each column ranges through all cabin zones. Each of the sets $ZONES$, EA, CRC, LAV, WCA combines location identifiers of one type.

$$
\begin{aligned}
LOCATION ::= {} & z1 \mid z2 \mid z3 \mid z4 \mid z5 \mid \\
& fwd \mid mid1 \mid mid2 \mid aft \mid \\
& crc1 \mid crc2 \mid lav \mid \\
& window \mid centre \mid aisle
\end{aligned}
$$

$$
\begin{aligned}
ZONES &== \{z1, z2, z3, z4, z5\} \\
EA &== \{fwd, mid1, mid2, aft\} \\
CRC &== \{crc1, crc2\} \\
LAV &== \{lav\} \\
WCA &== \{window, centre, aisle\}
\end{aligned}
$$

The following types are used to describe state information of peripherals: DIM denotes the state of an illumination unit with increasing intensity off, $dim1$, $dim2$, $bright$.

$onNl2$ denotes the "ON-state" of *night lights solution 2*, which differs from the intensities $dim1$, $dim2$, $bright$, since these light units are of a different hardware type. $SWITCH$ is used to describe whether an *indicator* for specific buttons on a panel is illuminated ($active$) ore not ($passive$). $LGCIU$ gives state information from the *landing gear control unit*: On ground, landing gears are in state $downCompressed$, in flight they are either $downLocked$ or $upLocked$. $PRESSURE$ denotes the state of the *cabin pressure*; $high$ is the normal state, while low denotes the emergency state of insufficient cabin pressure. $DOOR$ is the state of the *cockpit door*. The values of $FEATURE$ are used as general flags to indicate whether a certain CIL feature is enabled due to the CAM configuration.

$$
\begin{array}{ll}
DIM & ::= dim1 \mid dim2 \mid bright \mid off \mid onNl2 \\
DIM0 & == \{dim1, dim2, bright, off\} \\
DIM1 & == \{dim1, dim2, bright\} \\
SWITCH & ::= passive \mid active \\
LGCIU & ::= downCompressed \mid downLocked \mid upLocked \\
PRESSURE & ::= low \mid high \\
DOOR & ::= closed \mid open \\
FEATURE & ::= disabled \mid enabled
\end{array}
$$

Relation $<_{dim}$ introduces an ordering between the dim steps in the obvious sense:

$$
\begin{array}{|l}
\_<_{dim}\_ : DIM0 \leftrightarrow DIM0 \\
\hline
off <_{dim} dim2 \\
dim2 <_{dim} dim1 \\
dim1 <_{dim} bright \\
\forall\, a, b, c : DIM \mid a <_{dim} b \land b <_{dim} c \bullet a <_{dim} c
\end{array}
$$

3.3 CIL Application Interface – Inputs

The CIDS operating system provides a service layer to each application that allows it to handle inputs in the style of an event-driven system. To find the appropriate level of abstraction for the interfaces in our Z specification, this section provides a re-specification of these system services using the Z notation.

Panel Events If any button is pressed on an AAP or FAP/MPB, a corresponding event will be visible to the application layer for one CIDS application cycle. Since the source of this panel event (i. e. an AAP or the FAP or the MPB) does not have any impact on the corresponding CIL operation, it is not necessary to include the source in the event model. Moreover, conflicting commands like pressing the *ZONE 1 BRIGHT* command at AAP No. 1 and the *ZONE 1 DIM1* command at AAP No. 2 are simply resolved by discarding the second (and any additional) conflicting command. This leads to the event model described by schema *PanelEvents*, to be interpreted as follows:

1. $zoneEvent(z) = d$ means, that the d-Button of zone z has been pressed at one of the panels. If no button for zone z has been pressed during the last CIDS application cycle, z will not be contained in the domain of *zoneEvent*.
2. $z \in nlEvent$ means, that the night light button of zone z has been pressed.
3. $c \in wcaEvent$ means that the button for column c has been pressed.
4. $eaEvent(e) = d$ means, that the d-Button of entry area e has been pressed.
5. $crcEvent(crc) = d$ means, that the d-Button of crew rest compartment crc has been pressed.
6. $maintEvent = active$ means, that the maintenance button has been pressed.
7. $mainEvent = active$ means, that the MAIN button has been pressed.

PanelEvents

$$
\begin{array}{l}
zoneEvent : ZONES \mathrel{+\!\!\!\rightarrow} DIM1 \\
nlEvent : \mathbb{P}\ ZONES \\
wcaEvent : \mathbb{P}\ WCA \\
eaEvent : EA \mathrel{+\!\!\!\rightarrow} DIM1 \\
crcEvent : CRC \mathrel{+\!\!\!\rightarrow} DIM1 \\
maintEvent : SWITCH \\
mainEvent : SWITCH
\end{array}
$$

System States and System Events The CIL application also evaluates information provided by other subsystems of the AIRBUS A330/340. This information is stored in the operating system as *state/event pairs*: It is always possible to read the actual information value. As soon as a value changes, this will be indicated by the operating system by setting a corresponding event flag for the duration of one CIDS application cycle. The state/event pairs are specified by schemas *SystemStates* and *SystemEvents*. Components $CPC1, \ldots, oilPres$

carry the actual state value of the cabin pressure controllers 1 and 2, the landing gears, the state of the FAP, the cockpit door and the oil pressure controller. $CPC1Event, \ldots, oilPresEvent$ indicate that the corresponding state value has just changed.

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\textit{SystemStates}} \\
CPC1 : PRESSURE \\
CPC2 : PRESSURE \\
LGEARst : LGCIU \\
FAP : SWITCH \\
cockDoor : DOOR \\
oilPres : PRESSURE \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\textit{SystemEvents}} \\
CPC1Event : SWITCH \\
CPC2Event : SWITCH \\
LGEARstEvent : SWITCH \\
FAPEvent : SWITCH \\
cockDoorEvent : SWITCH \\
oilPresEvent : SWITCH \\
\hline
\end{array}
$$

The above schemas are state schemas, because any update to their components is performed by the operating system layer, which is outside the scope of our specification. Moreover, during each CIL execution the event and state values remain unchanged. Therefore the application layer can regard these values as state information. Moreover, these schemas are not initialized in the CIL initialization, because the operating system takes care of that.

3.4 CIL Application Interface – Outputs

The effect of a CIL operation becomes visible in a change of state (i. e. brightness) of the illumination units connected to specific DEUA addresses and a state change for button indicators from *active* to *passive* and vice versa.

Interface to Illumination Units Changing the brightness of an illumination unit is performed by the application by writing a value associated with the brightness to a memory location associated with the corresponding DEUA address. As a consequence, the memory area reserved for control of DEUA devices serves as the interface to the illumination units and at the same time reflects the actual state of all illumination units. Schema *ILLstate* defined in section 3.6.2

below presents an abstraction of this memory area. Consistent with the implementation, each CIL function will operate on this state schema by changing the brightness values of DEUA addresses.

Interface to Indicators To switch a button indicator, a function is provided taking the button identifier and the desired *SWITCH*-value as inputs. Again, we can abstract from explicitly setting the same indicator at every panel. We will explain in Section 3.6, that the actual state of the indicators can serve as a simple description of the actual state of cabin illumination in terms of locations and corresponding brightness values. Therefore schemas *ZONEINDstate*, *EAINDstate*, *CRCINDstate*, *MAINTINDstate*, *MAININDstate* are defined below to store the actual state of indicators for cabin zones, entry areas, crew rest compartments, lavatories and the MAIN button. Since in the implementation any state change will be 'synchronously' accompanied by a corresponding function call to set the indicator, each operation's impact on the button indicators will be modelled on specification level simply by a state change for the above mentioned schemas.

3.5 Definition of CAM Tables – Axiomatic Definitions

In our context CAM tables can be viewed as *global constants* that can be accessed from anywhere in the specification. Therefore in Z, they are modelled as *axiomatic definitions*.

The following axiomatic definition gives the Z definition of the CAM tables "Assignment of DEU, type A, outputs" from the CIL system specification:

CAM_CAB The illumination units identified by a certain subset of *ADDRESS* (namely, the domain of the partial function CAM_CAB) are associated with cabin zones: If illumination unit a is contained in the domain of CAM_CAB, then CAM_CAB(a) is the cabin zone where the illumination unit is installed. Note that since CAM_CAB is a partial function, we know that each illumination unit can be associated with at most one cabin zone.

CAM_WCA Each illumination unit a with address in the domain of CAM_WCA is contained in the cabin column CAM_WCA(a).

CAM_EA, CAM_LAV, CAM_CRC Each illumination unit with address a in the domain of CAM_EA, CAM_LAV, CAM_CRC is contained in the entry area/lavatory/crew rest compartment CAM_EA(a)/ CAM_LAV(a)/ CAM_CRC(a).

CAM_NL1/CAM_NL2 Each illumination unit a with address in the domain of CAM_NL1/CAM_NL2 is used as a night light solution 1/2 in cabin zone

CAM_NL1(a)/CAM_NL2(a).

CAM_FAP To handle the FAP INOP reaction, this CAM table defines the cabin locations being controllable by the FAP.

The interpretation of the invariants specified in the axiomatic definition's predicate part is:

1. The illumination units associated with cabin zones ($a \in \mathrm{dom}\,\mathrm{CAM\_CAB}$) are exactly the units associated with $window-$, $centre-$, $aisle-$columns ($a \in \mathrm{dom}\,\mathrm{CAM\_WCA}$).
2. If a is a night light solution 1, then it is also used as a "normal" illumination unit of the associated cabin zone.
3. Each illumination unit associated with a cabin zone, an entry area, a crew rest compartment or a lavatory cannot be associated with another area at the same time.
4. The night lights solution 2 are separate illumination units, they are not used for "normal" illumination of any cabin area.

$$\begin{array}{l}
\mathrm{CAM_CAB} : ADDRESS \twoheadrightarrow ZONES \\
\mathrm{CAM_WCA} : ADDRESS \twoheadrightarrow WCA \\
\mathrm{CAM_EA} : ADDRESS \twoheadrightarrow EA \\
\mathrm{CAM_LAV} : ADDRESS \twoheadrightarrow LAV \\
\mathrm{CAM_CRC} : ADDRESS \twoheadrightarrow CRC \\
\mathrm{CAM_NL1} : ADDRESS \twoheadrightarrow ZONES \\
\mathrm{CAM_NL2} : ADDRESS \twoheadrightarrow ZONES \\
\mathrm{CAM_FAP} : \mathbb{P}\ LOCATION \\
\hline
\mathrm{dom}\,\mathrm{CAM_CAB} = \mathrm{dom}\,\mathrm{CAM_WCA} \\
\mathrm{CAM_NL1} \subseteq \mathrm{CAM_CAB} \\
\mathrm{dom}\,\mathrm{CAM_CAB} \cap \\
\quad (\mathrm{dom}\,\mathrm{CAM_EA} \cup \mathrm{dom}\,\mathrm{CAM_LAV} \cup \mathrm{dom}\,\mathrm{CAM_CRC} \\
\qquad \cup\, \mathrm{dom}\,\mathrm{CAM_NL2}) = \emptyset \\
\mathrm{dom}\,\mathrm{CAM_EA} \cap \\
\quad (\mathrm{dom}\,\mathrm{CAM_LAV} \cup \mathrm{dom}\,\mathrm{CAM_CRC} \cup \mathrm{dom}\,\mathrm{CAM_NL2}) = \emptyset \\
\mathrm{dom}\,\mathrm{CAM_LAV} \cap \\
\quad (\mathrm{dom}\,\mathrm{CAM_CRC} \cup \mathrm{dom}\,\mathrm{CAM_NL2}) = \emptyset \\
\mathrm{dom}\,\mathrm{CAM_CRC} \cap \mathrm{dom}\,\mathrm{CAM_NL2} = \emptyset
\end{array}$$

CAM table CAM_EADIM defines a specific intensity for a certain illumination associated with the forward entry area. The illumination units a contained in the domain of CAM_EADIM are used for the *entry area dimming function* described below.

$$\begin{array}{|l} \mathrm{CAM\_EADIM} : \mathit{ADDRESS} \pfun \{\mathit{off}, \mathit{dim}1, \mathit{dim}2\} \\ \hline \mathrm{dom}\,\mathrm{CAM\_EADIM} \subseteq \mathrm{dom}(\mathrm{CAM\_EA} \rhd \{\mathit{fwd}\}) \end{array}$$

The following CAM tables are flags for the configuration of the CIL application. "*enabled*" means in each case:

CAM_DECOMP CIL will automatically switch illumination to *bright* in case of cabin decompression.

CAM_BLOCK During cabin decompression, all illumination commands will be blocked, so the illumination will stay in state *bright*.

CAM_MAINT Pressing the maintenance button will switch the lavatory illumination to state *bright*.

CAM_NLAUTO If the illumination is switched off completely in a zone due to a *dim*1, *dim*2, *bright* command or a *window*, *centre*, *aisle* command, the night lights associated with this zone are automatically switched on. (If lights are switched off by means of the *main* command, the night lights are switched off, too.)

$$\begin{array}{|l} \mathrm{CAM\_DECOMP} : \mathit{FEATURE} \\ \mathrm{CAM\_BLOCK} : \mathit{FEATURE} \\ \mathrm{CAM\_MAINT} : \mathit{FEATURE} \\ \mathrm{CAM\_NLAUTO} : \mathit{FEATURE} \end{array}$$

CAM_NL1_DIM denotes the intensity for night lights solution 1 in case of manual or automatic night light activation.

$$\begin{array}{|l} \mathrm{CAM\_NL1\_DIM} : \mathit{DIM}0 \end{array}$$

3.6 Specification of System States – State Schemas

In this section, we will provide the system's state description. It is always a good idea to try and find state variables that are directly observable or at least of "obvious significance" from the application's point of view. In our case, the idea for the description of the states was:

- The indicators on the panels' illumination command buttons reflect the state of illumination in the location associated with each button as it is intended by the user.

- The intensity of each illumination unit $a \in ADDRESS$ reflects the state of illumination as it has actually been switched by the CIL application.

3.6.1 State Description for Button Indicators

For this specification's level of abstraction, the difference between the FAP, MPB and AAPs is irrelevant, since the effect of pressing a button only depends on the button's function, not on the panel where the button has been pressed. We define certain groups of buttons with similar functions and model each group as a mapping, as defined in schema *ZONEINDstate*:

- *zoneInd* models the state of $dim1-$, $dim2-$, $bright-$buttons defined for each cabin zone: $zoneInd(z) = dim1, dim2, bright$ means, that the $dim1$, $dim2$, *bright* indicator of zone z is active. Otherwise, we have $zoneInd(z) = off$. Note that this automatically models the requirement that in each zone at most one of the buttons may be active.

- *nlInd* models the night light buttons associated with each cabin zone: $nlInd(z) = active$ means that the night light indicator for zone z is active.

- *wcaInd* models the state of the three $window-$, $centre-$, $isle-$buttons (WCA-buttons) in analogy to *nlInd*.

The meaning of the state invariants defined in the schema's predicate part is

1. If the illumination in zone z is not switched off, then the corresponding night light indicator must be *passive*, if the night light auto function is enabled in CAM_NLAUTO.

2. At least one zone has illumination state $d \neq off$ if and only if at least one of the WCA-buttons is active.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\_\textit{ZONEINDstate}\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
zoneInd : ZONES \longrightarrow DIM0 \\
nlInd : ZONES \longrightarrow SWITCH \\
wcaInd : WCA \longrightarrow SWITCH \\
\hline
\forall z : ZONES \bullet \\
\qquad nlInd(z) = active \Rightarrow (zoneInd(z) = off \lor \mathrm{CAM\_NLAUTO} = disabled) \\
(\exists z : ZONES \bullet zoneInd(z) \neq off) \Leftrightarrow \\
\qquad active \in \mathrm{ran}\ wcaInd \\
\hline
\end{array}
$$

Schema *EAINDstate* models the state of the entry area buttons. *eaInd* is defined in analogy to *zoneInd*.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\_\textit{EAINDstate}\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
eaInd : EA \longrightarrow DIM0 \\
\hline
\end{array}
$$

Schema *CRCINDstate* models the state of the crew rest compartment buttons. *crcInd* is defined in analogy to *zoneInd*.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\_\textit{CRCINDstate}\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
crcInd : CRC \longrightarrow DIM0 \\
\hline
\end{array}
$$

Schema *MAINTINDstate* models the state of the maintenance button. Note that there is no button to switch the lavatory lights off. This can only be achieved by switching off all the other illumination units, so that a MAIN OFF condition is reached.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\_\textit{MAINTINDstate}\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
maintInd : SWITCH \\
\hline
\end{array}
$$

Schema *MAININDstate* models the state of the MAIN button. The predicate part states, that the MAIN button will be passive if and only if the illumination has been switched off in every location, as far as this is under manual control.

$$
\begin{array}{l}
\textbf{MAININDstate} \\
\hline
ZONEINDstate \\
EAINDstate \\
CRCINDstate \\
MAINTINDstate \\
mainInd : SWITCH \\
\hline
mainInd = passive \Leftrightarrow \\
\quad maintInd = passive \land \\
\quad \operatorname{ran} nlInd = \{passive\} \land \operatorname{ran} zoneInd = \{off\} \land \\
\quad \operatorname{ran} eaInd = \{off\} \land \operatorname{ran} crcInd = \{off\} \land \\
\quad \operatorname{ran} wcaInd = \{passive\} \\
\hline
\end{array}
$$

3.6.2 State Description for Illumination Units

The collection of all illumination units is described as the mapping *ill* from *ADDRESS* to the set *DIM*. For example, $ill(a) = bright$ means that an illumination unit is associated with address a and this unit is switched to intensity *bright*.

$$
\begin{array}{l}
\textbf{ILLstate} \\
\hline
ill : ADDRESS \twoheadrightarrow DIM \\
\hline
\operatorname{dom} ill = \operatorname{dom} \mathrm{CAM\_CAB} \cup \operatorname{dom} \mathrm{CAM\_EA} \cup \\
\quad \operatorname{dom} \mathrm{CAM\_LAV} \cup \operatorname{dom} \mathrm{CAM\_CRC} \cup \operatorname{dom} \mathrm{CAM\_NL2} \\
\forall a : \operatorname{dom} ill \bullet (ill(a) = onNl2 \Rightarrow a \in \operatorname{dom} \mathrm{CAM\_NL2}) \\
\forall a : \operatorname{dom} \mathrm{CAM\_LAV} \bullet ill(a) \in \{dim1, bright, off\} \\
\hline
\end{array}
$$

3.6.3 Correspondence Between Button Indicators and Illumination Units

In this subsection we summarize the consistency conditions existing between the states of button indicators and the actual state of illumination.

The schema *ZONEinvariant* states consistency conditions between buttons related to cabin zones and the corresponding state of illumination:

1. If the WCA-indicator for column c is *active* and the state of illumination in zone z is $d \neq off$, then in zone z every illumination unit associated with column c is switched on with intensity d.

2. If the WCA-indicator for column c is *passive* and the state of illumination in zone z is $d \neq \mathit{off}$, then in zone z every illumination unit associated with column c is switched *off*. Note that this only holds in the case when the state of illumination in zone z is not *off*, because otherwise we have to consider the case when certain illumination units are used as night lights solution 1. This is described in the next cases.

3. If the state of illumination in zone z is *off* and the night light indicator of zone z is *active*, then every night light solution 1 associated with zone z is active with intensity according to CAM_NL1_DIM. Moreover, every night light solution 2 is switched on. Every other illumination unit associated with zone z is switched *off*.

4. If the state of illumination in zone z is *off* and the night light indicator of zone z is *passive*, then every illumination unit associated with zone z (including night lights solution 2) is switched *off*.

5. If the illumination of zone z is in state $d \neq \mathit{off}$, then every night light solution 2 associated with zone z is *off*.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{ZONEinvariant}} \\
\textit{ZONEINDstate} \\
\textit{ILLstate} \\
\hline
\forall c : \mathit{WCA}; z : \mathit{ZONES} \bullet \\
\quad \mathit{wcaInd}(c) = \mathit{active} \Rightarrow \\
\quad\quad (\forall a : \mathrm{dom}(\mathrm{CAM\_WCA} \rhd \{c\}) \cap \mathrm{dom}(\mathrm{CAM\_CAB} \rhd \{z\}) \bullet \\
\quad\quad\quad \mathit{ill}(a) = \mathit{zoneInd}(z)) \\
\\
\forall c : \mathit{WCA}; z : \mathit{ZONES} \bullet \\
\quad \mathit{wcaInd}(c) = \mathit{passive} \wedge \mathit{zoneInd}(z) \neq \mathit{off} \Rightarrow \\
\quad\quad (\forall a : \mathrm{dom}(\mathrm{CAM\_WCA} \rhd \{c\}) \cap \mathrm{dom}(\mathrm{CAM\_CAB} \rhd \{z\}) \bullet \mathit{ill}(a) = \mathit{off}) \\
\\
\forall z : \mathit{ZONES} \bullet \\
\quad \mathit{zoneInd}(z) = \mathit{off} \wedge \mathit{nlInd}(z) = \mathit{active} \Rightarrow \\
\quad\quad (\forall a : \mathrm{dom}(\mathrm{CAM\_CAB} \rhd \{z\}) \setminus \mathrm{dom}(\mathrm{CAM\_NL1} \rhd \{z\}) \bullet \mathit{ill}(a) = \mathit{off}) \wedge \\
\quad\quad (\forall a : \mathrm{dom}(\mathrm{CAM\_NL1} \rhd \{z\}) \bullet \mathit{ill}(a) = \mathrm{CAM\_NL1\_DIM}) \wedge \\
\quad\quad (\forall a : \mathrm{dom}(\mathrm{CAM\_NL2} \rhd \{z\}) \bullet \mathit{ill}(a) = \mathit{onNl2}) \\
\\
\forall z : \mathit{ZONES} \bullet \\
\quad \mathit{zoneInd}(z) = \mathit{off} \wedge \mathit{nlInd}(z) = \mathit{passive} \Rightarrow \\
\quad\quad (\forall a : \mathrm{dom}(\mathrm{CAM\_CAB} \rhd \{z\}) \bullet \mathit{ill}(a) = \mathit{off}) \wedge \\
\quad\quad (\forall a : \mathrm{dom}(\mathrm{CAM\_NL2} \rhd \{z\}) \bullet \mathit{ill}(a) = \mathit{off}) \\
\\
\forall z : \mathit{ZONES} \bullet \\
\quad \mathit{zoneInd}(z) \neq \mathit{off} \Rightarrow \\
\quad\quad (\forall a : \mathrm{dom}(\mathrm{CAM\_NL2} \rhd \{z\}) \bullet \mathit{ill}(a) = \mathit{off}) \\
\hline
\end{array}
$$

For the entry areas, the illumination units associated with their location are given by CAM_EA. In the case of a running engine ($oilPres = high$) with open cockpit door, the illumination units with address space dom CAM_EADIM must be dimmed to the CAM-defined value, if this is less than the actual brightness value indicated by the corresponding entry area button.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad EAinvariant \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}} \\
EAINDstate \\
ILLstate \\
SystemStates \\
\hline
\forall e : EA \bullet \\
\quad eaInd(e) \neq off \Rightarrow \\
\quad\quad (\forall a : \mathrm{dom}(\mathrm{CAM\_EA} \rhd \{e\}) \setminus \mathrm{dom}\,\mathrm{CAM\_EADIM} \bullet ill(a) = eaInd(e)) \land \\
\quad\quad (\forall a : \mathrm{dom}(\mathrm{CAM\_EA} \rhd \{e\}) \cap \mathrm{dom}\,\mathrm{CAM\_EADIM} \bullet \\
\quad\quad\quad (cockDoor = closed \lor oilPres = low \lor eaInd(e) <_{dim} \mathrm{CAM\_EADIM}(a) \\
\quad\quad\quad\quad \Rightarrow ill(a) = eaInd(e)) \land \\
\quad\quad\quad (cockDoor = open \land oilPres = high \land \mathrm{CAM\_EADIM}(a) <_{dim} eaInd(e) \\
\quad\quad\quad\quad \Rightarrow ill(a) = \mathrm{CAM\_EADIM}(a))) \\
\\
\forall e : EA \bullet \\
\quad eaInd(e) = off \Rightarrow (\forall a : \mathrm{dom}(\mathrm{CAM\_EA} \rhd \{e\}) \bullet ill(a) = off) \\
\hline
\end{array}
$$

The addresses associated with crew rest compartments are defined by CAM_CRC and shall always have the state defined by the CRC button indicators.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad CRCinvariant \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}} \\
CRCINDstate \\
ILLstate \\
\hline
\forall c : CRC \bullet (\forall a : \mathrm{dom}(\mathrm{CAM\_CRC} \rhd \{c\}) \bullet ill(a) = crcInd(c)) \\
\hline
\end{array}
$$

Schema *LAVinvariant* states that the lavatory illumination shall always be active if this holds for the MAIN-button indicator. If in addition, the maintenance indicator is active, the illumination will be *bright*.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad LAVinvariant \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}} \\
MAININDstate \\
ILLstate \\
\hline
mainInd = passive \Leftrightarrow \\
\quad (\forall a : \mathrm{dom}\ \mathrm{CAM\_LAV} \bullet ill(a) = off) \\
\\
mainInd = active \land maintInd = passive \Leftrightarrow \\
\quad (\forall a : \mathrm{dom}\ \mathrm{CAM\_LAV} \bullet ill(a) = dim1) \\
\\
mainInd = active \land maintInd = active \Leftrightarrow \\
\quad (\forall a : \mathrm{dom}\ \mathrm{CAM\_LAV} \bullet ill(a) = bright) \\
\hline
\end{array}
$$

3.7 Description of CIL Operations – Operational Schemas

3.7.1 Auxiliary Schemas

Schema *NOop* is used to indicate that the CIL state remains unchanged in situations where an operation does not have any effect (e.g. because it is disabled in CAM).

$$
\begin{array}{|l}
\hline
\textit{NOop} \\
\hline
\Xi ZONEINDstate \\
\Xi EAINDstate \\
\Xi CRCINDstate \\
\Xi MAINTINDstate \\
\Xi MAININDstate \\
\Xi ILLstate \\
\hline
\end{array}
$$

The following schema describes the control of the MAIN-button indicator which depends on the state of the other indicators. The schema is only used in sequential composition with other operational schemas: The other schemas provide the pre-state, *MAINLAVadjust* describes how the main-button indicator is then adjusted to the new CIL state. Furthermore, the illumination units of the lavatories are switched on or off, if the MAIN indicator changes its state accordingly.

$$
\begin{array}{|l}
\hline
\textit{MAINLAVadjust} \\
\hline
\Delta MAININDstate \\
\Delta ILLstate \\
\Xi ZONEINDstate \\
\Xi EAINDstate \\
\Xi CRCINDstate \\
\Xi MAINTINDstate \\
\hline
mainInd' = \\
\quad \mathbf{if}\ (maintInd = active \lor \\
\qquad active \in \operatorname{ran} nlInd \lor \operatorname{ran} zoneInd \neq \{off\} \lor \\
\qquad \operatorname{ran} eaInd \neq \{off\} \lor \operatorname{ran} crcInd \neq \{off\} \lor \\
\qquad active \in \operatorname{ran} wcaInd) \\
\quad \mathbf{then}\ active\ \mathbf{else}\ mainInd \\
ill' = \mathbf{if}\ mainInd' = passive \lor maintInd = active\ \mathbf{then}\ ill\ \mathbf{else} \\
\qquad ill \oplus \{a : \operatorname{dom} \mathrm{CAM\_LAV} \bullet a \mapsto dim1\} \\
\hline
\end{array}
$$

3.7.2 Initialization Operation

On system initialization, the following state of illumination is adopted:

1. For cabin zones, entry areas and crew rest compartments, the associated illumination units are switched *bright* and the corresponding indicators (including WCA-buttons) are switched *active*. Night lights solution 2 and all other indicators related to these locations are switched off.
2. In the lavatories, lights are switched to intensity $dim1$. The *maintenance button indicator* is *passive*.
3. The *main button indicator* is switched *active*.

$$
\begin{array}{|l}
\hline
\textit{ZONEINDINITop} \\
\textit{ZONEINDstate}' \\
\hline
\textit{zoneInd}' = \{ z : \textit{ZONES} \bullet z \mapsto \textit{bright} \} \\
\textit{nlInd}' = \{ z : \textit{ZONES} \bullet z \mapsto \textit{passive} \} \\
\textit{wcaInd}' = \{ w : \textit{WCA} \bullet w \mapsto \textit{active} \} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{EAINDINITop} \\
\textit{EAINDstate}' \\
\hline
\textit{eaInd}' = \{ z : \textit{EA} \bullet z \mapsto \textit{bright} \} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{CRCINDINITop} \\
\textit{CRCINDstate}' \\
\hline
\textit{crcInd}' = \{ z : \textit{CRC} \bullet z \mapsto \textit{bright} \} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{MAINTINDINITop} \\
\textit{MAINTINDstate}' \\
\hline
\textit{maintInd}' = \textit{passive} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{MAININDINITop} \\
\textit{MAININDstate}' \\
\hline
\textit{mainInd}' = \textit{active} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{ILLINITop} \\
\textit{ILLstate}' \\
\hline
\textit{ill}' = \{a : (\mathrm{dom}\,\mathrm{CAM\_CAB} \cup \mathrm{dom}\,\mathrm{CAM\_EA} \cup \mathrm{dom}\,\mathrm{CAM\_CRC}) \bullet \\
\qquad\qquad a \mapsto \textit{bright}\} \\
\qquad \cup \{a : \mathrm{dom}\,\mathrm{CAM\_LAV} \bullet a \mapsto \textit{dim}1\} \\
\qquad \cup \{a : \mathrm{dom}\,\mathrm{CAM\_NL2} \bullet a \mapsto \textit{off}\} \\
\hline
\end{array}
$$

$$
\begin{array}{l}
\textit{INITop} \mathrel{\widehat=} \\
\quad \textit{ZONEINDINITop} \land \textit{EAINDINITop} \land \textit{CRCINDINITop} \land \\
\qquad \textit{MAINTINDINITop} \land \textit{MAININDINITop} \land \textit{ILLINITop}
\end{array}
$$

3.7.3 Operations Related to the Main Button

The *main ON/OFF operation* only changes the system state if the landing gears are in state *downCompressed* as it is indicated by schema *MAINisBlocked*. In the non-blocked state, the *main command* will switch the illumination completely *off*, if the previous state of the main button was *active*. (As a consequence, all CIL indicators will also be switched off.) If the previous state was *passive*, the illumination will be switched on with exactly the same effect as on system initialization.

$$
\begin{array}{|l}
\hline
\textit{MAININDopPassive} \\
\Delta \textit{MAININDstate} \\
\hline
\textit{mainInd} = \textit{active} \\
\textit{mainInd}' = \textit{passive} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{MAINILLopPassive} \\
\Delta \textit{ILLstate} \\
\hline
\textit{ill}' = \{a : \mathrm{dom}\ \textit{ill} \bullet a \mapsto \textit{off}\} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{MAINisBlocked} \\
\textit{SystemStates} \\
\hline
\textit{LGEARst} \in \{\textit{downLocked}, \textit{upLocked}\} \\
\hline
\end{array}
$$

$$
\begin{array}{l}
\textit{MAINop} \mathrel{\widehat=} \\
\quad (\lnot\ \textit{MAINisBlocked} \land (\textit{MAINILLopPassive} \land \textit{MAININDopPassive} \lor \\
\qquad [\textit{MAININDstate} \mid \textit{mainInd} = \textit{passive}] \land \textit{INITop})) \lor \\
\quad (\textit{MAINisBlocked} \land \textit{NOop})
\end{array}
$$

3.7.4 Switching Operation for Cabin Zones

In this section the functions to control cabin illumination by means of the buttons *dim1, dim2, bright* for each zone are specified. The total operation is divided into the cases *switch passive* $\longrightarrow$ *active* and *switch active* $\longrightarrow$ *passive*. Moreover, the control of the button indicators is separated from the control of the illumination units.

Schema *ZONEINDopActive* specifies the control of the button indicators for case *switch passive* $\longrightarrow$ *active*. Inputs $(z?, d?)$ specify that the button for brightness $d?$ has been pressed in zone $z?$. The pre-condition to perform the transition into a new illumination state with brightness $d?$ is, that the button $(z?, d?)$ is not already active. According to our definition of *zoneInd*, this is expressed by condition $zoneInd(z?) \neq d?$. In this case, the button indicator will be activated and any zone button previously active for zone $z?$ will be switched off. Note that both switching operations are specified by the expression $\{z? \mapsto d?\}$, since only one zone button is allowed to be active at a time. Otherwise the zone buttons are left unchanged.

If the night light auto function is enabled (CAM_NLAUTO = *enabled*), the night light button indicator of zone $z?$ must automatically be switched off, as soon as one of the illumination states $dim1$, $dim2$, *bright* is taken. If the night light auto function is disabled, the night light indicators will remain unchanged.

If the cabin lights had previously been switched off (and as a consequence the WCA indicators are switched off, too), they will now all be re-activated. Otherwise the WCA indicators remain unchanged.

Note that we have taken care to specify that *ZONEINDopActive* does not alter any of the button indicators described in schemas *CRCINDstate*, *EAINDstate*, *MAINTINDstate*. This is necessary, because later on the schema will be used in $\wedge$-combination with schemas *EAinvariant*, *CRCinvariant* and *LAVinvariant*. Therefore it operates in fact on the complete button indicator state space, though only changes of *ZONEINDstate* are visible in the schema. This situation will be clarified in all schemas by using the $\Xi \ldots INDstate$-expressions.

$$
\begin{array}{|l}
\hline
\textit{ZONEINDopActive} \\
\Delta ZONEINDstate \\
\Xi CRCINDstate \\
\Xi EAINDstate \\
\Xi MAINTINDstate \\
z? : ZONES \\
d? : DIM1 \\
\hline
zoneInd(z?) \neq d? \\
zoneInd' = zoneInd \oplus \{z? \mapsto d?\} \\
nlInd' = \mathbf{if}\ \mathrm{CAM\_NLAUTO} = enabled\ \mathbf{then}\ nlInd \oplus \{z? \mapsto passive\}\ \mathbf{else}\ nlInd \\
wcaInd' = \mathbf{if}\ active \notin \operatorname{ran} wcaInd\ \mathbf{then}\ \{x : WCA \bullet x \mapsto active\}\ \mathbf{else}\ wcaInd \\
\hline
\end{array}
$$

Schema *ZONEINDopPassive* handles the case *switch active* $\longrightarrow$ *passive*. The previously active zone indicator $d?$ of zone $z?$ is switched off. The night light indicators will automatically be activated, if the night light auto function is enabled in CAM. If $z?$ was the last zone to be switched off, the WCA indicators will also be switched off.

$$
\begin{array}{|l}
\hline
\textit{ZONEINDopPassive} \\
\Delta ZONEINDstate \\
\Xi CRCINDstate \\
\Xi EAINDstate \\
\Xi MAINTINDstate \\
z? : ZONES \\
d? : DIM1 \\
\hline
zoneInd(z?) = d? \\
zoneInd' = zoneInd \oplus \{z? \mapsto off\} \\
nlInd' = \mathbf{if}\ \mathrm{CAM\_NLAUTO} = enabled\ \mathbf{then}\ nlInd \oplus \{z? \mapsto active\}\ \mathbf{else}\ nlInd \\
wcaInd' = \\
\qquad \mathbf{if}\ (\forall z : ZONES \mid z \neq z? \bullet zoneInd(z) = off) \\
\qquad \mathbf{then}\ \{x : WCA \bullet x \mapsto passive\} \\
\qquad \mathbf{else}\ wcaInd \\
\hline
\end{array}
$$

Schema *ZONEILLopActive* handles the case *switch passive* $\longrightarrow$ *active* for the illumination units. Night lights that might have been previously active are switched off. If at least one of the WCA columns is active, the illumination units x of zone $z?$ will only be switched to intensity $d?$ in active columns ($wcaInd(\mathrm{CAM\_WCA}(x)) = active$). If the WCA columns have been passive, the illumination units of all columns in zone $z?$ will be switched to $d?$. Note that in this case we do not have to switch off night lights solution I, since they are a subset of the zone $z?$ illumination units.

$$
\begin{array}{|l}
\hline
\textit{ZONEILLopActive} \\
\Delta \textit{ILLstate} \\
\textit{ZONEINDstate} \\
z? : \textit{ZONES} \\
d? : \textit{DIM}1 \\
\hline
\textit{zoneInd}(z?) \neq d? \land \textit{active} \in \operatorname{ran} \textit{wcaInd} \Rightarrow \\
\textit{ill}' = \textit{ill} \oplus \\
\qquad \{x : \operatorname{dom}(\mathrm{CAM\_NL1} \rhd \{z?\}) \bullet x \mapsto \textit{off}\} \oplus \\
\qquad \{x : \operatorname{dom}(\mathrm{CAM\_NL2} \rhd \{z?\}) \bullet x \mapsto \textit{off}\} \oplus \\
\qquad \{x : \operatorname{dom}(\mathrm{CAM\_CAB} \rhd \{z?\}) \mid \\
\qquad\qquad \textit{wcaInd}(\mathrm{CAM\_WCA}(x)) = \textit{active} \bullet x \mapsto d?\} \\
\\
\textit{zoneInd}(z?) \neq d? \land \textit{active} \notin \operatorname{ran} \textit{wcaInd} \Rightarrow \\
\textit{ill}' = \textit{ill} \oplus \\
\qquad \{x : \operatorname{dom}(\mathrm{CAM\_NL2} \rhd \{z?\}) \bullet x \mapsto \textit{off}\} \oplus \\
\qquad \{x : \operatorname{dom}(\mathrm{CAM\_CAB} \rhd \{z?\}) \bullet x \mapsto d?\} \\
\hline
\end{array}
$$

Schema *ZONEILLopPassive* handles the case *switch active* $\longrightarrow$ *passive* for the illumination units. If the night light auto function is enabled or if the user has pre-selected the night light function by pressing the night light button of zone $z?$, night lights solution I and II will be turned on in zone $z?$. The solution I night lights are switched to their CAM-defined intensity. Otherwise the illumination units of zone $z?$ are switched off.

$$
\begin{array}{|l}
\hline
\textit{ZONEILLopPassive} \\
\Delta \textit{ILLstate} \\
\textit{ZONEINDstate} \\
z? : \textit{ZONES} \\
d? : \textit{DIM}1 \\
\hline
\textit{zoneInd}(z?) = d? \\
\\
\textit{ill}' = \textit{ill} \oplus \\
\qquad \{x : \operatorname{dom}(\mathrm{CAM\_CAB} \rhd \{z?\}) \bullet x \mapsto \textit{off}\} \oplus \\
\qquad \{x : \operatorname{dom}(\mathrm{CAM\_NL1} \rhd \{z?\}) \mid \\
\qquad\qquad (\textit{nlInd}(z?) = \textit{active} \lor \mathrm{CAM\_NLAUTO} = \textit{enabled}) \\
\qquad\qquad \bullet\, x \mapsto \mathrm{CAM\_NL1\_DIM}\} \oplus \\
\qquad \{x : \operatorname{dom}(\mathrm{CAM\_NL2} \rhd \{z?\}) \mid \\
\qquad\qquad (\textit{nlInd}(z?) = \textit{active} \lor \mathrm{CAM\_NLAUTO} = \textit{enabled}) \\
\qquad\qquad \bullet\, x \mapsto \textit{onNl}2\} \\
\hline
\end{array}
$$

Schema equation *ZONEop* combines the above schemas controlling cabin illumination by means of zone buttons, into a total function.

$$
\begin{array}{l}
\mathit{ZONEop} \mathrel{\widehat=} \\
\quad \mathit{ZONEILLopActive} \land \mathit{ZONEINDopActive} \lor \\
\qquad \mathit{ZONEILLopPassive} \land \mathit{ZONEINDopPassive}
\end{array}
$$

3.7.5 Switching Operation for Window/Centre/Aisles

This section specifies the operation of window-, centre- and aisle (WCA)- buttons. This operation will be inactive, if the illumination in the zones has been switched off (ran $zoneInd = \{off\}$), so that only night lights might be active.

$$
\begin{array}{|l}
\hline
\textit{WCAINDopActive} \\
\Delta \mathit{ZONEINDstate} \\
\Xi \mathit{CRCINDstate} \\
\Xi \mathit{EAINDstate} \\
\Xi \mathit{MAINTINDstate} \\
c? : \mathit{WCA} \\
\hline
\mathit{wcaInd}(c?) = \mathit{passive} \land \operatorname{ran} \mathit{zoneInd} \neq \{\mathit{off}\} \\
\mathit{zoneInd}' = \mathit{zoneInd} \\
\mathit{nlInd}' = \mathit{nlInd} \\
\mathit{wcaInd}' = \mathit{wcaInd} \oplus \{c? \mapsto \mathit{active}\} \\
\hline
\end{array}
$$

If column $c?$ is activated, the illumination units of this column ($x : \operatorname{dom}(\mathrm{CAM\_WCA} \rhd \{c?\})$) will be switched to the intensity $zoneInd(\mathrm{CAM\_CAB}(x))$ being valid for the associated zone.

$$
\begin{array}{|l}
\hline
\textit{WCAILLopActive} \\
\Delta \mathit{ILLstate} \\
\mathit{ZONEINDstate} \\
c? : \mathit{WCA} \\
\hline
\mathit{wcaInd}(c?) = \mathit{passive} \land \operatorname{ran} \mathit{zoneInd} \neq \{\mathit{off}\} \\
\mathit{ill}' = \mathit{ill} \oplus \\
\quad \{x : \operatorname{dom}(\mathrm{CAM\_WCA} \rhd \{c?\}) \bullet x \mapsto \mathit{zoneInd}(\mathrm{CAM\_CAB}(x))\} \\
\hline
\end{array}
$$

If $c?$ is the last WCA button indicator to be turned off, this will also switch off the zone buttons. If in this case the night light auto function is enabled, the night light button indicators will be turned on in every zone.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\_\_ \; \mathit{WCAINDopPassive} \; \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
\Delta \mathit{ZONEINDstate} \\
\Xi \mathit{CRCINDstate} \\
\Xi \mathit{EAINDstate} \\
\Xi \mathit{MAINTINDstate} \\
c? : \mathit{WCA} \\
\hline
\mathit{wcaInd}(c?) = \mathit{active} \\
\\
\mathit{zoneInd}' = \\
\qquad\qquad \textbf{if } (\forall w : \mathit{WCA} \setminus \{c?\} \bullet \mathit{wcaInd}(w) = \mathit{passive}) \\
\qquad\qquad \textbf{then } \{z : \mathit{ZONES} \bullet z \mapsto \mathit{off}\} \\
\qquad\qquad \textbf{else } \mathit{zoneInd} \\
\\
\mathit{nlInd}' = \\
\qquad\qquad \textbf{if } (\forall w : \mathit{WCA} \setminus \{c?\} \bullet \mathit{wcaInd}(w) = \mathit{passive}) \wedge \\
\qquad\qquad\qquad \mathrm{CAM\_NLAUTO} = \mathit{enabled} \\
\qquad\qquad \textbf{then } \{z : \mathit{ZONES} \bullet z \mapsto \mathit{active}\} \\
\qquad\qquad \textbf{else } \mathit{nlInd} \\
\\
\mathit{wcaInd}' = \mathit{wcaInd} \oplus \{c? \mapsto \mathit{passive}\} \\
\hline
\end{array}
$$

If $c?$ is the last column to be switched off, the night lights of each zone will be switched on, if either the night light auto function is enabled or the night light function had been pre-selected ($\mathit{nlInd}(\mathrm{CAM\_NL1}(x)) = \mathit{active}$). If still other active columns exist, only the illumination units associated with column $c?$ will be turned off.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\_\ \mathit{WCAILLopPassive}\ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
\Delta \mathit{ILLstate} \\
\mathit{ZONEINDstate} \\
c? : \mathit{WCA} \\
\hline
\mathit{wcaInd}(c?) = \mathit{active} \\
\mathit{ill}' = \\
\quad \textbf{if } (\forall w : \mathit{WCA} \setminus \{c?\} \bullet \mathit{wcaInd}(w) = \mathit{passive}) \\
\quad \textbf{then } \mathit{ill} \oplus \\
\quad\quad \{x : \mathrm{dom}\,(\mathrm{CAM\_WCA} \rhd \{c?\}) \bullet x \mapsto \mathit{off}\} \oplus \\
\quad\quad \{x : \mathrm{dom}\ \mathrm{CAM\_NL1} \mid \\
\quad\quad\quad \mathit{nlInd}(\mathrm{CAM\_NL1}(x)) = \mathit{active} \lor \mathrm{CAM\_NLAUTO} = \mathit{enabled} \\
\quad\quad\quad \bullet\ x \mapsto \mathrm{CAM\_NL1\_DIM}\} \oplus \\
\quad\quad \{x : \mathrm{dom}\ \mathrm{CAM\_NL2} \mid \\
\quad\quad\quad \mathit{nlInd}(\mathrm{CAM\_NL2}(x)) = \mathit{active} \lor \mathrm{CAM\_NLAUTO} = \mathit{enabled} \\
\quad\quad\quad \bullet\ x \mapsto \mathit{onNl2}\} \\
\quad \textbf{else } \mathit{ill} \oplus \{x : \mathrm{dom}\,(\mathrm{CAM\_WCA} \rhd \{c?\}) \bullet x \mapsto \mathit{off}\} \\
\hline
\end{array}
$$

Similar to the schema equation of the previous section, *WCAop* defines a total operation for window- centre- and aisle-buttons. If the illumination has been turned off in all zones, pressing a WCA button will have no effect.

$$
\begin{array}{l}
\mathit{WCAop} \mathrel{\widehat=} \\
\qquad (\mathit{WCAILLopActive} \land \mathit{WCAINDopActive} \lor \\
\qquad \mathit{WCAILLopPassive} \land \mathit{WCAINDopPassive}) \lor \\
\qquad [\mathit{ZONEINDstate};\ c? : \mathit{WCA} \mid \mathrm{ran}\ \mathit{zoneInd} = \{\mathit{off}\}] \land \mathit{NOop}
\end{array}
$$

3.7.6 Entry Area Commands

Schemas *EAINDop, EAILLopPassive, EAILLopActive* specify the behaviour of the entry area illumination commands. Here we have to take into account a special situation: If the cockpit door is open and the oil pressure is high (this reflects the fact that the aircraft has a running engine), certain illumination units in the *fwd* entry area must not be switched to a state that is brighter than the one indicated in table CAM_EADIM. This mechanism is implemented in order to avoid blinding the cockpit personnel. It is reflected by schema *EAILLopActive*.

$$
\begin{array}{|l}
\hline
\textit{EAINDop} \\
\hline
\Delta EAINDstate \\
\Xi CRCINDstate \\
\Xi ZONEINDstate \\
\Xi EAINDstate \\
\Xi MAINTINDstate \\
ea? : EA \\
dim? : DIM1 \\
\hline
eaInd(ea?) = dim? \Rightarrow \\
\quad eaInd' = eaInd \oplus \{ea? \mapsto off\} \\
eaInd(ea?) \neq dim? \Rightarrow \\
\quad eaInd' = eaInd \oplus \{ea? \mapsto dim?\} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{EAILLopPassive} \\
\hline
\Delta ILLstate \\
EAINDstate \\
ea? : EA \\
dim? : DIM1 \\
\hline
eaInd(ea?) = dim? \\
\quad ill' = ill \oplus \\
\qquad \{x : \mathrm{dom}(\mathrm{CAM\_EA} \rhd \{ea?\}) \bullet x \mapsto off\} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{EAILLopActive} \\
\hline
\Delta ILLstate \\
EAINDstate \\
\Xi SystemStates \\
ea? : EA \\
dim? : DIM1 \\
\hline
eaInd(ea?) \neq dim? \\
\quad ill' = \\
\qquad \mathbf{if}\ ea? = fwd \land cockDoor = open \land oilPres = high \\
\qquad \mathbf{then}\ ill \oplus \\
\qquad\quad \{x : \mathrm{dom}(\mathrm{CAM\_EA} \rhd \{ea?\}) \bullet x \mapsto dim?\} \oplus \\
\qquad\quad \{x : \mathrm{dom}\ \mathrm{CAM\_EADIM} \mid \\
\qquad\qquad \mathrm{CAM\_EADIM}(x) <_{dim} dim? \bullet x \mapsto \mathrm{CAM\_EADIM}(x)\} \\
\qquad \mathbf{else}\ ill \oplus \{x : \mathrm{dom}(\mathrm{CAM\_EA} \rhd \{ea?\}) \bullet x \mapsto dim?\} \\
\hline
\end{array}
$$

The complete operation for entry areas is described by equation

$EAop \mathrel{\widehat=} EAINDop \land (EAILLopActive \lor EAILLopPassive)$

3.7.7 Automatic Dimming Operation for Forward Entry Area

If the oil pressure controller signals "oil pressure high" (i.e. engine running) and the cockpit door sensor signals "door open", then the CAM-assigned illumination units located in the forward entry area will be switched to the CAM-defined intensity I if the actual intensity is not already lower than I. As soon as the door closes again or the oil pressure becomes low, these illumination units resume their previous state.

$$
\begin{array}{l}
\textit{AUTODIMop} \\
\hline
\Xi \textit{EAINDstate} \\
\Xi \textit{SystemStates} \\
\Delta \textit{ILLstate} \\
\hline
\textit{cockDoor} = \textit{open} \land \textit{oilPres} = \textit{high} \Rightarrow \\
\quad \textit{ill}' = \textit{ill} \oplus \\
\qquad \{a : \mathrm{dom}\,\mathrm{CAM\_EADIM} \mid \\
\qquad\quad \mathrm{CAM\_EADIM}(a) <_{dim} \textit{eaInd}(\textit{fwd}) \bullet a \mapsto \mathrm{CAM\_EADIM}(a)\} \\
\textit{cockDoor} = \textit{closed} \lor \textit{oilPres} = \textit{low} \Rightarrow \\
\quad \textit{ill}' = \textit{ill} \oplus \\
\qquad \{a : \mathrm{dom}\,\mathrm{CAM\_EADIM} \bullet a \mapsto \textit{eaInd}(\textit{fwd})\}
\end{array}
$$

3.7.8 Switching Operation for CRC Illumination Units

Control of the crew rest compartment illumination is performed in analogy to the control of the entry areas. However, here no special cases like automatic dimming have to be considered.

$$
\begin{array}{l}
\textit{CRCINDop} \\
\hline
\Delta \textit{CRCINDstate} \\
\Xi \textit{ZONEINDstate} \\
\Xi \textit{EAINDstate} \\
\Xi \textit{MAINTINDstate} \\
\textit{crc}? : \textit{CRC} \\
\textit{dim}? : \textit{DIM}1 \\
\hline
\textit{crcInd}(\textit{crc}?) = \textit{dim}? \Rightarrow \\
\quad \textit{crcInd}' = \textit{crcInd} \oplus \{\textit{crc}? \mapsto \textit{off}\} \\
\textit{crcInd}(\textit{crc}?) \neq \textit{dim}? \Rightarrow \\
\quad \textit{crcInd}' = \textit{crcInd} \oplus \{\textit{crc}? \mapsto \textit{dim}?\}
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{CRCILLop} \\
\Delta \textit{ILLstate} \\
\textit{CRCINDstate} \\
crc? : CRC \\
dim? : DIM1 \\
\hline
crcInd(crc?) \neq dim? \Rightarrow \\
\quad ill' = ill \oplus \\
\qquad \{x : \mathrm{dom}(\mathrm{CAM\_CRC} \rhd \{crc?\}) \bullet x \mapsto dim?\} \\
crcInd(crc?) = dim? \Rightarrow \\
\quad ill' = ill \oplus \\
\qquad \{x : \mathrm{dom}(\mathrm{CAM\_CRC} \rhd \{crc?\}) \bullet x \mapsto off\} \\
\hline
\end{array}
$$

$CRCop \mathrel{\widehat{=}} CRCINDop \land CRCILLop$

3.7.9 Maintenance Command

If the maintenance function is enabled according to CAM_MAINT, the maintenance button can be used to toggle the illumination intensity in the lavatories from *dim*1 to *bright* and vice versa.

$$
\begin{array}{|l}
\hline
\textit{MAINTINDop} \\
\Delta \textit{MAINTINDstate} \\
\hline
maintInd' = \\
\qquad \mathbf{if}\ maintInd = passive \land \mathrm{CAM\_MAINT} = enabled \\
\qquad \mathbf{then}\ active\ \mathbf{else}\ passive \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{MAINTILLop} \\
\Delta \textit{ILLstate} \\
\textit{MAINTINDstate} \\
\hline
ill' = \\
\quad \mathbf{if}\ maintInd = passive \land \mathrm{CAM\_MAINT} = enabled \\
\quad \mathbf{then}\ ill \oplus \{a : \mathrm{dom}\ \mathrm{CAM\_LAV} \bullet a \mapsto bright\} \\
\quad \mathbf{else}\ ill \oplus \{a : \mathrm{dom}\ \mathrm{CAM\_LAV} \bullet a \mapsto dim1\} \\
\hline
\end{array}
$$

$MAINTop \mathrel{\widehat{=}} MAINTINDop \land MAINTILLop$

3.7.10 Night Light Operation

The following schemas describe the operation of the night light buttons. If the night light button indicator of zone $n?$ is passive, it will be turned on, if the night light auto function is disabled or the zone illumination has previously been turned off. In the former case this will serve as a pre-selection, so that night lights will automatically be turned on, as soon as the zone illumination has been switched off. If the night light auto function is enabled and the illumination is active, pressing a night light button will have no effect, because when switching off the zone illumination the night lights will automatically be turned on in any case.

$$
\begin{array}{|l}
\hline
\textit{NLINDop} \\
\Delta \textit{ZONEINDstate} \\
\Xi \textit{CRCINDstate} \\
\Xi \textit{EAINDstate} \\
\Xi \textit{MAINTINDstate} \\
n? : \textit{ZONES} \\
\hline
\textit{nlInd}(n?) = \textit{passive} \Rightarrow \\
\qquad \textit{nlInd}' = \\
\qquad\qquad \textbf{if } \mathrm{CAM\_NLAUTO} = \textit{disabled} \lor \textit{zoneInd}(n?) = \textit{off} \\
\qquad\qquad \textbf{then } \textit{nlInd} \oplus \{n? \mapsto \textit{active}\} \\
\qquad\qquad \textbf{else } \textit{nlInd} \\
\textit{nlInd}(n?) = \textit{active} \Rightarrow \\
\qquad\qquad \textit{nlInd}' = \textit{nlInd} \oplus \{n? \mapsto \textit{passive}\} \\
\textit{zoneInd}' = \textit{zoneInd} \\
\textit{wcaInd}' = \textit{wcaInd} \\
\hline
\end{array}
$$

Toggling the night light button will only affect the illumination units assigned as night lights solution I or II, if the cabin illumination of zone $n?$ has been turned off.

$$
\begin{array}{l}
\textit{NLILLop} \\
\hline
\textit{ZONEINDstate} \\
\Delta \textit{ILLstate} \\
n? : \textit{ZONES} \\
\hline
\textit{nlInd}(n?) = \textit{passive} \Rightarrow \\
\quad \textit{ill}' = \\
\qquad \textbf{if } \textit{zoneInd}(n?) = \textit{off} \\
\qquad \textbf{then } \textit{ill} \oplus \\
\qquad\quad \{x : \mathrm{dom}(\mathrm{CAM\_NL2} \rhd \{n?\}) \bullet x \mapsto \textit{onNl2}\} \oplus \\
\qquad\quad \{x : \mathrm{dom}(\mathrm{CAM\_NL1} \rhd \{n?\}) \bullet x \mapsto \mathrm{CAM\_NL1\_DIM}\} \\
\qquad \textbf{else } \textit{ill} \\
\\
\textit{nlInd}(n?) = \textit{active} \Rightarrow \\
\quad \textit{ill}' = \\
\qquad \textbf{if } \textit{zoneInd}(n?) = \textit{off} \\
\qquad \textbf{then } \textit{ill} \oplus \\
\qquad\quad \{x : \mathrm{dom}(\mathrm{CAM\_NL2} \rhd \{n?\}) \bullet x \mapsto \textit{off}\} \oplus \\
\qquad\quad \{x : \mathrm{dom}(\mathrm{CAM\_NL1} \rhd \{n?\}) \bullet x \mapsto \textit{off}\} \\
\qquad \textbf{else } \textit{ill} \\
\hline
\end{array}
$$

$\textit{NLop} \mathrel{\widehat{=}} \textit{NLILLop} \land \textit{NLINDop}$

3.7.11 Operations Related to Cabin Decompression

In case of cabin decompression, the indicators will be set so that they reflect the 'all bright' state.

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\mathit{DECOMPINDopActive}} \\
\Delta \mathit{ZONEINDstate} \\
\Delta \mathit{EAINDstate} \\
\Delta \mathit{CRCINDstate} \\
\Delta \mathit{MAINTINDstate} \\
\Delta \mathit{MAININDstate} \\
\hline
\mathit{mainInd}' = \mathit{active} \\
\mathit{maintInd}' = \mathit{active} \\
\mathit{zoneInd}' = \{ z : \mathit{ZONES} \bullet z \mapsto \mathit{bright} \} \\
\mathit{nlInd}' = \mathit{nlInd} \oplus \\
\qquad \{ z : \mathit{ZONES} \mid \mathrm{CAM\_NLAUTO} = \mathit{enabled} \bullet z \mapsto \mathit{passive} \} \\
\mathit{wcaInd}' = \{ x : \mathit{WCA} \bullet x \mapsto \mathit{active} \} \\
\mathit{eaInd}' = \{ e : \mathit{EA} \bullet e \mapsto \mathit{bright} \} \\
\mathit{crcInd}' = \{ c : \mathit{CRC} \bullet c \mapsto \mathit{bright} \} \\
\hline
\end{array}
$$

The cabin decompression function will only become active, if it is enabled in CAM and at least one of the cabin pressure controllers $CPC1$, $CPC2$ indicate *'pressure low'*. In this case the illumination units will be switched to bright. Only if the condition for *fwd* entry area dimming holds, will the associated illumination units be dimmed (ref. 3.7.7).

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\mathit{DECOMPILLopActive}} \\
\Delta \mathit{ILLstate} \\
\Xi \mathit{SystemStates} \\
\hline
\mathit{ill}' = \{ a : \mathrm{dom}\,\mathrm{CAM\_NL2} \bullet a \mapsto \mathit{off} \} \oplus \\
\qquad \{ a : (\mathrm{dom}\,\mathrm{CAM\_CAB} \cup \mathrm{dom}\,\mathrm{CAM\_EA} \cup \\
\qquad\qquad \mathrm{dom}\,\mathrm{CAM\_LAV} \cup \mathrm{dom}\,\mathrm{CAM\_CRC}) \bullet a \mapsto \mathit{bright} \} \oplus \\
\qquad \{ a : \mathrm{dom}\ \mathrm{CAM\_EADIM} \mid \\
\qquad\qquad \mathit{cockDoor} = \mathit{open} \land \mathit{oilPres} = \mathit{high} \bullet a \mapsto \mathrm{CAM\_EADIM}(a) \} \\
\hline
\end{array}
$$

$$
\begin{array}{l}
\mathit{DECOMPop} \mathrel{\widehat=} \\
\quad [\mathit{SystemStates} \mid \mathrm{CAM\_DECOMP} = \mathit{enabled} \land (\mathit{CPC}1 = \mathit{low} \lor \mathit{CPC}2 = \mathit{low})] \\
\qquad \land\ \mathit{DECOMPINDopActive} \land \mathit{DECOMPILLopActive} \\
\quad \lor\ [\mathit{SystemStates} \mid \mathrm{CAM\_DECOMP} = \mathit{disabled} \lor \mathit{CPC}1 = \mathit{high} \land \mathit{CPC}2 = \mathit{high}] \\
\qquad \land\ \mathit{NOop}
\end{array}
$$

3.7.12 FAP Inop Reaction

If an FAP failure occurs, the illumination in every cabin location controlled by the FAP must be switched to *bright*. These locations are given by CAM table CAM_FAP. Though schema equation *FAPinop* does not look very appealing, its structure is quite simple and re-uses the existing CIL operations: For each location – cabin zones, entry areas, crew rest compartments and lavatories – it is first checked whether the location is controlled by the FAP and whether its actual illumination state is not already *bright*. If both conditions are true, the corresponding operation will be activated, if necessary by piping the location and the desired intensity into the function.

$$
\begin{array}{l}
FAPinop \mathrel{\widehat=} \\
((([ZONEINDstate; z! : ZONES; d! : DIM1 \mid \\
\qquad zoneInd(z1) \neq bright \land z1 \in \mathrm{CAM\_FAP} \land z! = z1 \land d! = bright] \gg ZONEop) \lor \\
\quad [ZONEINDstate; z : ZONES \mid zoneInd(z1) = bright \lor z1 \notin \mathrm{CAM\_FAP}] \land NOop) \mathbin{;} \\
((([ZONEINDstate; z! : ZONES; d! : DIM1 \mid \\
\qquad zoneInd(z2) \neq bright \land z2 \in \mathrm{CAM\_FAP} \land z! = z2 \land d! = bright] \gg ZONEop) \lor \\
\quad [ZONEINDstate; z : ZONES \mid zoneInd(z2) = bright \lor z2 \notin \mathrm{CAM\_FAP}] \land NOop) \mathbin{;} \\
((([ZONEINDstate; z! : ZONES; d! : DIM1 \mid \\
\qquad zoneInd(z3) \neq bright \land z3 \in \mathrm{CAM\_FAP} \land z! = z3 \land d! = bright] \gg ZONEop) \lor \\
\quad [ZONEINDstate; z : ZONES \mid zoneInd(z3) = bright \lor z3 \notin \mathrm{CAM\_FAP}] \land NOop) \mathbin{;} \\
((([ZONEINDstate; z! : ZONES; d! : DIM1 \mid \\
\qquad zoneInd(z4) \neq bright \land z4 \in \mathrm{CAM\_FAP} \land z! = z4 \land d! = bright] \gg ZONEop) \lor \\
\quad [ZONEINDstate; z : ZONES \mid zoneInd(z4) = bright \lor z4 \notin \mathrm{CAM\_FAP}] \land NOop) \mathbin{;} \\
((([ZONEINDstate; z! : ZONES; d! : DIM1 \mid \\
\qquad zoneInd(z5) \neq bright \land z5 \in \mathrm{CAM\_FAP} \land z! = z5 \land d! = bright] \gg ZONEop) \lor \\
\quad [ZONEINDstate; z : ZONES \mid zoneInd(z5) = bright \lor z5 \notin \mathrm{CAM\_FAP}] \land NOop) \mathbin{;} \\
((([EAINDstate; ea! : EA; dim! : DIM1 \mid \\
\qquad eaInd(fwd) \neq bright \land fwd \in \mathrm{CAM\_FAP} \land ea! = fwd \land dim! = bright] \gg EAop) \lor \\
\quad [EAINDstate; ea : EA \mid eaInd(fwd) = bright \lor fwd \notin \mathrm{CAM\_FAP}] \land NOop) \mathbin{;} \\
((([EAINDstate; ea! : EA; dim! : DIM1 \mid \\
\qquad eaInd(mid1) \neq bright \land mid1 \in \mathrm{CAM\_FAP} \land ea! = mid1 \land dim! = bright] \gg EAop) \lor \\
\quad [EAINDstate; ea : EA \mid eaInd(mid1) = bright \lor mid1 \notin \mathrm{CAM\_FAP}] \land NOop) \mathbin{;} \\
((([EAINDstate; ea! : EA; dim! : DIM1 \mid \\
\qquad eaInd(mid2) \neq bright \land mid2 \in \mathrm{CAM\_FAP} \land ea! = mid2 \land dim! = bright] \gg EAop) \lor \\
\quad [EAINDstate; ea : EA \mid eaInd(mid2) = bright \lor mid2 \notin \mathrm{CAM\_FAP}] \land NOop) \mathbin{;} \\
((([EAINDstate; ea! : EA; dim! : DIM1 \mid \\
\qquad eaInd(aft) \neq bright \land aft \in \mathrm{CAM\_FAP} \land ea! = aft \land dim! = bright] \gg EAop) \lor \\
\quad [EAINDstate; ea : EA \mid eaInd(aft) = bright \lor aft \notin \mathrm{CAM\_FAP}] \land NOop) \mathbin{;} \\
((([CRCINDstate; crc! : CRC; dim! : DIM1 \mid \\
\qquad crcInd(crc1) \neq bright \land crc1 \in \mathrm{CAM\_FAP} \land crc! = crc1 \land dim! = bright] \gg CRCop) \lor \\
\quad [CRCINDstate; crc : CRC \mid crcInd(crc1) = bright \lor crc1 \notin \mathrm{CAM\_FAP}] \land NOop) \mathbin{;} \\
((([CRCINDstate; crc! : CRC; dim! : DIM1 \mid \\
\qquad crcInd(crc2) \neq bright \land crc2 \in \mathrm{CAM\_FAP} \land crc! = crc2 \land dim! = bright] \gg CRCop) \lor \\
\quad [CRCINDstate; crc : CRC \mid crcInd(crc2) = bright \lor crc2 \notin \mathrm{CAM\_FAP}] \land NOop) \mathbin{;} \\
((([MAINTINDstate; l : LOCATION \mid maintInd = passive \land lav \in \mathrm{CAM\_FAP}] \land MAINTop) \lor \\
\quad [MAINTINDstate; l : LOCATION \mid maintInd = active \lor lav \notin \mathrm{CAM\_FAP}] \land NOop)) \mathbin{;} \\
MAINLAVadjust
\end{array}
$$

The FAP inop reaction is triggered, when the FAP system status is passive:

$$
\begin{array}{l}
FAPctrl \mathrel{\widehat=} \\
\quad [SystemStates \mid FAP = passive] \land FAPinop \lor \\
\qquad [SystemStates \mid FAP = active] \land NOop
\end{array}
$$

3.7.13 Top-Level Structure of the Specification

The CIL application provides two top-level entries: The initialization – as specified by schema *INITop* in 3.7.2 – is activated only once on CIDS system startup. In this section the second top-level entry is described: This is the schema equation *CIL* representing the control structure which specifies how each of the CIL functions is activated during one CIDS application cycle.

Since each event only lives for one cycle, it is important to activate the associated functions at once. Moreover, we wish to separate the examination of a function's trigger conditions from the operation itself. Therefore *CIL* specifies the control structure of a sequential system checking for each function whether it should be activated and – if the relevant activation conditions hold – it will trigger the operation, at the same time providing its required inputs $x?, \ldots$. With respect to trigger conditions, we make the following distinctions:

- Event expressions describing conditions for function activation are handled by the top-level control structure.
- Data conditions (e. g. values of input parameters and of CAM tables) peventing or otherwise influencing the function execution are processed within each function.

Schema equation *UserOp* below specifies the trigger conditions for the CIL operations accessible to the user, i. e. zone operations, WCA operations, entry area operations, operations for the crew rest compartments, reaction to the maintenance button, night light operation and reaction to the MAIN button. The control structure has been designed such that first the existence of corresponding events is checked. Then the function is triggered. If the function has input parameters, they will be provided by the control structure as a result of the event evaluation. The reason for not letting the user functions themselves evaluate their associated events is, that these operations may be re-used in a context, where the events are not active, e. g. for the FAP inop reaction described in 3.7.12. An operation (e. g. ZONEop) may be triggered more then once in this sequence, if several active events (e. g. dimming commands for several zones) require its execution.

Schema equation *UserOpZones* defines the part of *UserOp* handling events for cabin zone commands.

$$
\begin{array}{l}
\mathit{UserOpZones} \mathrel{\widehat=} \\
\quad (([\mathit{PanelEvents};\ z! : \mathit{ZONES};\ d! : \mathit{DIM1} \mid \\
\qquad z1 \in \mathrm{dom}\ \mathit{zoneEvent} \land z! = z1 \land d! = \mathit{zoneEvent}(z1)] \gg \mathit{ZONEop}) \lor \\
\qquad [\mathit{PanelEvents} \mid z1 \notin \mathrm{dom}\ \mathit{zoneEvent}] \land \mathit{NOop}) \fatsemi \\
\quad (([\mathit{PanelEvents};\ z! : \mathit{ZONES};\ d! : \mathit{DIM1} \mid \\
\qquad z2 \in \mathrm{dom}\ \mathit{zoneEvent} \land z! = z2 \land d! = \mathit{zoneEvent}(z2)] \gg \mathit{ZONEop}) \lor \\
\qquad [\mathit{PanelEvents} \mid z2 \notin \mathrm{dom}\ \mathit{zoneEvent}] \land \mathit{NOop}) \fatsemi \\
\quad (([\mathit{PanelEvents};\ z! : \mathit{ZONES};\ d! : \mathit{DIM1} \mid \\
\qquad z3 \in \mathrm{dom}\ \mathit{zoneEvent} \land z! = z3 \land d! = \mathit{zoneEvent}(z3)] \gg \mathit{ZONEop}) \lor \\
\qquad [\mathit{PanelEvents} \mid z3 \notin \mathrm{dom}\ \mathit{zoneEvent}] \land \mathit{NOop}) \fatsemi \\
\quad (([\mathit{PanelEvents};\ z! : \mathit{ZONES};\ d! : \mathit{DIM1} \mid \\
\qquad z4 \in \mathrm{dom}\ \mathit{zoneEvent} \land z! = z4 \land d! = \mathit{zoneEvent}(z4)] \gg \mathit{ZONEop}) \lor \\
\qquad [\mathit{PanelEvents} \mid z4 \notin \mathrm{dom}\ \mathit{zoneEvent}] \land \mathit{NOop}) \fatsemi \\
\quad (([\mathit{PanelEvents};\ z! : \mathit{ZONES};\ d! : \mathit{DIM1} \mid \\
\qquad z5 \in \mathrm{dom}\ \mathit{zoneEvent} \land z! = z5 \land d! = \mathit{zoneEvent}(z5)] \gg \mathit{ZONEop}) \lor \\
\qquad [\mathit{PanelEvents} \mid z5 \notin \mathrm{dom}\ \mathit{zoneEvent}] \land \mathit{NOop})
\end{array}
$$

Schema equation *UserOpWCA* defines the part of *UserOp* handling events for window/centre/aisles columns.

$$
\begin{array}{l}
\mathit{UserOpWCA} \mathrel{\widehat=} \\
\quad (([\mathit{PanelEvents};\ c! : \mathit{WCA} \mid \\
\qquad \mathit{window} \in \mathit{wcaEvent} \land c! = \mathit{window}] \gg \mathit{WCAop}) \lor \\
\qquad [\mathit{PanelEvents} \mid \mathit{window} \notin \mathit{wcaEvent}] \land \mathit{NOop}) \fatsemi \\
\quad (([\mathit{PanelEvents};\ c! : \mathit{WCA} \mid \\
\qquad \mathit{centre} \in \mathit{wcaEvent} \land c! = \mathit{centre}] \gg \mathit{WCAop}) \lor \\
\qquad [\mathit{PanelEvents} \mid \mathit{centre} \notin \mathit{wcaEvent}] \land \mathit{NOop}) \fatsemi \\
\quad (([\mathit{PanelEvents};\ c! : \mathit{WCA} \mid \\
\qquad \mathit{aisle} \in \mathit{wcaEvent} \land c! = \mathit{aisle}] \gg \mathit{WCAop}) \lor \\
\qquad [\mathit{PanelEvents} \mid \mathit{aisle} \notin \mathit{wcaEvent}] \land \mathit{NOop})
\end{array}
$$

Schema equation *UserOpEA* defines the part of *UserOp* which handles events for entry area commands.

$$
\begin{array}{l}
UserOpEA \mathrel{\widehat=} \\
\quad (([PanelEvents; ea! : EA; dim! : DIM1 \mid \\
\qquad fwd \in \mathrm{dom}\ eaEvent \land ea! = fwd \land dim! = eaEvent(fwd)] \gg EAop) \lor \\
\qquad [PanelEvents \mid fwd \notin \mathrm{dom}\ eaEvent] \land NOop) \mathbin{;} \\
\quad (([PanelEvents; ea! : EA; dim! : DIM1 \mid \\
\qquad mid1 \in \mathrm{dom}\ eaEvent \land ea! = mid1 \land dim! = eaEvent(mid1)] \gg EAop) \lor \\
\qquad [PanelEvents \mid mid1 \notin \mathrm{dom}\ eaEvent] \land NOop) \mathbin{;} \\
\quad (([PanelEvents; ea! : EA; dim! : DIM1 \mid \\
\qquad mid2 \in \mathrm{dom}\ eaEvent \land ea! = mid2 \land dim! = eaEvent(mid2)] \gg EAop) \lor \\
\qquad [PanelEvents \mid mid2 \notin \mathrm{dom}\ eaEvent] \land NOop) \mathbin{;} \\
\quad (([PanelEvents; ea! : EA; dim! : DIM1 \mid \\
\qquad aft \in \mathrm{dom}\ eaEvent \land ea! = aft \land dim! = eaEvent(aft)] \gg EAop) \lor \\
\qquad [PanelEvents \mid aft \notin \mathrm{dom}\ eaEvent] \land NOop)
\end{array}
$$

Schema equation *UserOpCRC* defines the part of *UserOp* handling events for crew rest compartment commands.

$$
\begin{array}{l}
UserOpCRC \mathrel{\widehat=} \\
\quad (([PanelEvents; crc! : CRC; dim! : DIM1 \mid \\
\qquad crc1 \in \mathrm{dom}\ crcEvent \land crc! = crc1 \land dim! = crcEvent(crc1)] \gg CRCop) \lor \\
\qquad [PanelEvents \mid crc1 \notin \mathrm{dom}\ crcEvent] \land NOop) \mathbin{;} \\
\quad (([PanelEvents; crc! : CRC; dim! : DIM1 \mid \\
\qquad crc2 \in \mathrm{dom}\ crcEvent \land crc! = crc2 \land dim! = crcEvent(crc2)] \gg CRCop) \lor \\
\qquad [PanelEvents \mid crc2 \notin \mathrm{dom}\ crcEvent] \land NOop)
\end{array}
$$

Schema equation *UserOpMAINT* defines the part of *UserOp* handling the maintenance button event.

$$
\begin{array}{l}
UserOpMAINT \mathrel{\widehat=} \\
\quad ([PanelEvents \mid maintEvent = active] \land MAINTop \lor \\
\qquad [PanelEvents \mid maintEvent = passive] \land NOop)
\end{array}
$$

Schema equation *UserOpNightLights* defines the part of *UserOp* handling the events for the night light function.

$$
\begin{array}{l}
UserOpNightLights \mathrel{\widehat=} \\
\quad ((([PanelEvents;\ n! : ZONES \mid \\
\qquad z1 \in nlEvent \land n! = z1] \gg NLop) \lor \\
\qquad [PanelEvents \mid z1 \notin nlEvent] \land NOop) \mathbin{;} \\
\quad ((([PanelEvents;\ n! : ZONES \mid \\
\qquad z2 \in nlEvent \land n! = z2] \gg NLop) \lor \\
\qquad [PanelEvents \mid z2 \notin nlEvent] \land NOop) \mathbin{;} \\
\quad ((([PanelEvents;\ n! : ZONES \mid \\
\qquad z3 \in nlEvent \land n! = z3] \gg NLop) \lor \\
\qquad [PanelEvents \mid z3 \notin nlEvent] \land NOop) \mathbin{;} \\
\quad ((([PanelEvents;\ n! : ZONES \mid \\
\qquad z4 \in nlEvent \land n! = z4] \gg NLop) \lor \\
\qquad [PanelEvents \mid z4 \notin nlEvent] \land NOop) \mathbin{;} \\
\quad ((([PanelEvents;\ n! : ZONES \mid \\
\qquad z5 \in nlEvent \land n! = z5] \gg NLop) \lor \\
\qquad [PanelEvents \mid z5 \notin nlEvent] \land NOop)
\end{array}
$$

Schema equation *UserOpMAIN* defines the part of *UserOp* which handles the MAIN button event.

$$
\begin{array}{l}
UserOpMAIN \mathrel{\widehat=} \\
\quad ([PanelEvents \mid mainEvent = active] \land MAINop \lor \\
\qquad [PanelEvents \mid mainEvent = passive] \land NOop)
\end{array}
$$

The user operations are summarized by

$$
\begin{array}{l}
UserOp \mathrel{\widehat=} \\
\quad UserOpZones \mathbin{;} UserOpWCA \mathbin{;} UserOpEA \mathbin{;} \\
\quad UserOpCRC \mathbin{;} UserOpMAINT \mathbin{;} UserOpNightLights \mathbin{;} \\
\quad UserOpMAIN
\end{array}
$$

We allow one exception to the two rules given above: The predicate schema *UserIsBlocked* is evaluated on the top-level control structure, although it evaluates state information and CAM data. The reason for this is that none of the operations summarized in *UserOp* will be triggered if the condition *UserIsBlocked* holds. Therefore the condition is evaluated only once to determine whether any *UserOp*-function may be triggered at all.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\text{\_\_ } UserIsBlocked \text{ \_\_\_\_\_\_\_\_}} \\
SystemStates \\
\hline
CPC1 = low \lor CPC2 = low \\
\mathrm{CAM\_DECOMP} = enabled \\
\mathrm{CAM\_BLOCK} = enabled \\
\hline
\end{array}
$$

Schema equation CIL now represents the top-level specification for the sequential composition of isolated CIL operations. If the corresponding events are active, the operations to react on cabin decompression, failure of the FAP or on events possibly requiring an execution of the automatic dimming function are triggered. Then condition *UserIsBlocked* is checked. If it does not hold, the user operations summarized in *UserOp* will be triggered, if corresponding events are active. The user operations may have an impact on the MAIN button indicator and on the state of the lavatory lights. This is covered by triggering *MAINLAVadjust* after having executed the *UserOp*-operations. Furthermore, the schema equation states that the global invariants defined in 3.6.3 must be respected by each operation. Since the functions have already been specified completely, introduction of the invariants results in an 'over specified' description of the CIL functionality. This was intended to produce two different 'views' on the application system: The operational schemas provide the *functional view*, the invariants the *data view* with its associated *safety conditions*. If the precondition theorems [13] for each operation prove to be correct, each operation is also consistent with the invariants. Therefore the two views, combined with the pre-condition calculation, provide a means to strengthen the specification's trustworthiness.

$$
\begin{array}{l}
CIL \mathrel{\widehat=} \\
\quad (([SystemEvents \mid CPC1Event = active \lor CPC2Event = active] \land DECOMPop \lor \\
\qquad [SystemEvents \mid CPC1Event = passive \land CPC2Event = passive] \land NOop) \, ; \\
\quad ([SystemEvents \mid FAPEvent = active] \land FAPctrl \lor \\
\qquad [SystemEvents \mid FAPEvent = passive] \land NOop) \, ; \\
\quad ([SystemEvents \mid cockDoorEvent = active \lor oilPresEvent = active] \land AUTODIMop \lor \\
\qquad [SystemEvents \mid cockDoorEvent = passive \land oilPresEvent = passive] \land NOop) \, ; \\
\quad (\lnot\ UserIsBlocked \land (UserOp \, ; MAINLAVadjust) \lor \\
\qquad UserIsBlocked \land NOop)) \land \\
\quad ZONEinvariant \land ZONEinvariant' \land \\
\quad EAinvariant \land EAinvariant' \land \\
\quad CRCinvariant \land CRCinvariant' \land \\
\quad LAVinvariant \land LAVinvariant'
\end{array}
$$

4 Transformation of Software Requirements into Software Design

In this section we will sketch how to transform the software requirements specification written in Z into a software design possessing the attributes of a "good" design according to design heuristics of software engineering. The transformation of isolated Z schemas into executable programs by means of data refinement and function refinement has been thoroughly investigated [7, 8, 11, 13]. Therefore we will focus on another issue, namely the derivation of a suitable *architectural design* from the overall structure of the Z software specification. Recall that the

CIL application will be developed as a sequential function, so we do not consider any aspects of concurrent process design. Therefore, in our context, architectural design means

- decomposition of the application into sequential modules
- definition of module interfaces
- design of the sequential *call structure* ("function A first calls function B, then function C")

4.1 Quality Criteria for Software Design

Observation of the following criteria usually helps to produce a clear and efficient software design. These criteria mostly apply to sequential systems implemented with conventional programming languages.

Module Cohesion The *Cohesion* of a module denotes the degree of functional homogeneity of the activities performed by the module. It will be regarded as good software engineering practice, if each module only implements (a part of) functionality being defined as a homogeneous application function in the software requirements specification. In general, this will result in easy traceability from design to specification and in better maintainability in case of specification changes and error localization. An important aspect of module cohesion is the module's interface design: It should always be the case, that a module execution makes use of every input parameter. Otherwise the module will probably combine different types of functionality, with each "sub-function" requiring a different subset of inputs.

Data Hiding Packages Complex data structures should not directly be accessed by application functions. Instead, the algorithms retrieving and manipulating the data should be encapsulated in *Data Hiding Packages* implementing the data structure and exporting a set of access functions for the application[21]. This will avoid the repeated implementation of algorithms in several application functions and facilitate changes of the data structure and the algorithms.

Module Re-Use Modules should be designed to be re-usable whenever this is possible without affecting their cohesion.

[21]We prefer not to call the packages 'abstract data types', since they do not necessarily provide a set of access functions, that completely characterize the underlying data type.

Separation of Control and Data Transformation To conquer the complexity of software design, it is useful to separate complex decision making from algorithmic data processing. The following rules are helpful to distribute the application's overall complexity in a suitable way:

- *System control signals* (e. g. the components of schema *SystemEvents*) should be separately evaluated in *transaction centers*. These functions evaluate control structures and activate the corresponding data processing functions.
- *Data conditions* (i. e. control structures evaluated according to the actual state of application data) should be evaluated in separate transaction centers, if the conditions are complex. In time-critical applications, this concept can be applied to avoid unnecessary function calls if the corresponding pre-condition for the data processing is false[22].
- Functions performing data transformations should not implement decision structures which evaluate system control signals.
- The call structure should be designed such that the transaction centers performing system control are implemented in the top-level of the call tree, the transaction centers evaluating data conditions in the tree's middle layer and the data transformation functions in its leaves.

4.2 Software Design Activities

For the development of software design, we recommend a mixed top-down and bottom-up approach: While the architecture is developed in top-down fashion, basic interface and data manipulation functions are designed in bottom-up manner. These basic functions represent the leaves of the function call tree. Therefore it is necessary to keep these 'targets of decomposition' in mind, while the top-down design is developed. For this reason, the design activities described below are executed rather in a "recursive" manner than in the exact sequence as we have chosen to present them.

Collection of System Interfaces Schemas *PanelEvents*, *SystemStates*, *SystemEvents* of section 3.3 indicate the system inputs required by the application. We can assume the following operating system services to be available:

[22]Note that this concept is often unsuitable for library functions being called by many different applications and should therefore be sufficiently robust. In this case, pre-conditions and other data conditions should be evaluated by the data processing function itself.

For each zone $z?$, function getZoneEvent will return the event $dim2$, $dim1$, $bright$, if the corresponding button has been pressed during the previous CIDS application cycle. Otherwise getZoneEvent returns off. Similarly, functions are provided to retrieve the other events summarized in $PanelEvents$.

To retrieve the actual value or event of system states, functions getSystemStates and getSystemEvents are defined. They input an identifier for $CPC1, \ldots, oilPres$ and return the corresponding state or event value.

As an output interface to control the button indicators, the operating system provides a function putButtonIndicator, which takes a button identifier and the desired $SWITCH$-state as input.

To control the illumination units, a pre-defined memory area is given, where the commands for illumination units can be stored. The operating system provides a function illumination to perform the mapping of DEUA addresses to corresponding memory locations and writes each command to the required location.

Designing the State Space Design of the state space is driven by the specification of state schemas in Section 3.6. Each of these schemas give rise to a data hiding module. This is illustrated by Figures 3 and 4 for the sub-space manipulated by zone operations.

The basic approach is to model the data structures together with a set of functions to access them. However, the access functions should not only be designed according to the data structure's characteristics, but rather according to the access operations performed in the specification's operational schemas. In case of $ZONEINDstate$ this is rather straightforward: The functions putZoneInd, putWCAInd, putNLInd take a pair $argument \mapsto image$ as input and implement functional overriding $f' = f \oplus \{x \mapsto y\}$. The state information is always changed together with the corresponding call to putButtonIndicator. It only has to be observed, that function putZoneInd may require two calls to putButtonIndicator, namely switching off the indicator previously active and activating the new one.

The package to access the $ILLstate$ is more complex to design. The pre-defined operating system service illumination serves as a data hiding access function. Inspection of the operational schemas shows, that $ILLstate$ is never retrieved, because all relevant state information is included in $ZONEINDstate$, $EAINDstate, \ldots$. As a consequence, we do not need any "getILLstate" functions. In case of the operations required for zone commands, the following functions are suggested: Schemas $ZONEILLopActive$ requires manipulation of illumination units cabin zones according to expression

$$\{x : \mathrm{dom}(\mathrm{CAM\_CAB} \rhd \{z?\}) \mid$$
$$\quad wcaInd(\mathrm{CAM\_WCA}(x)) = active \bullet x \mapsto d?\}$$

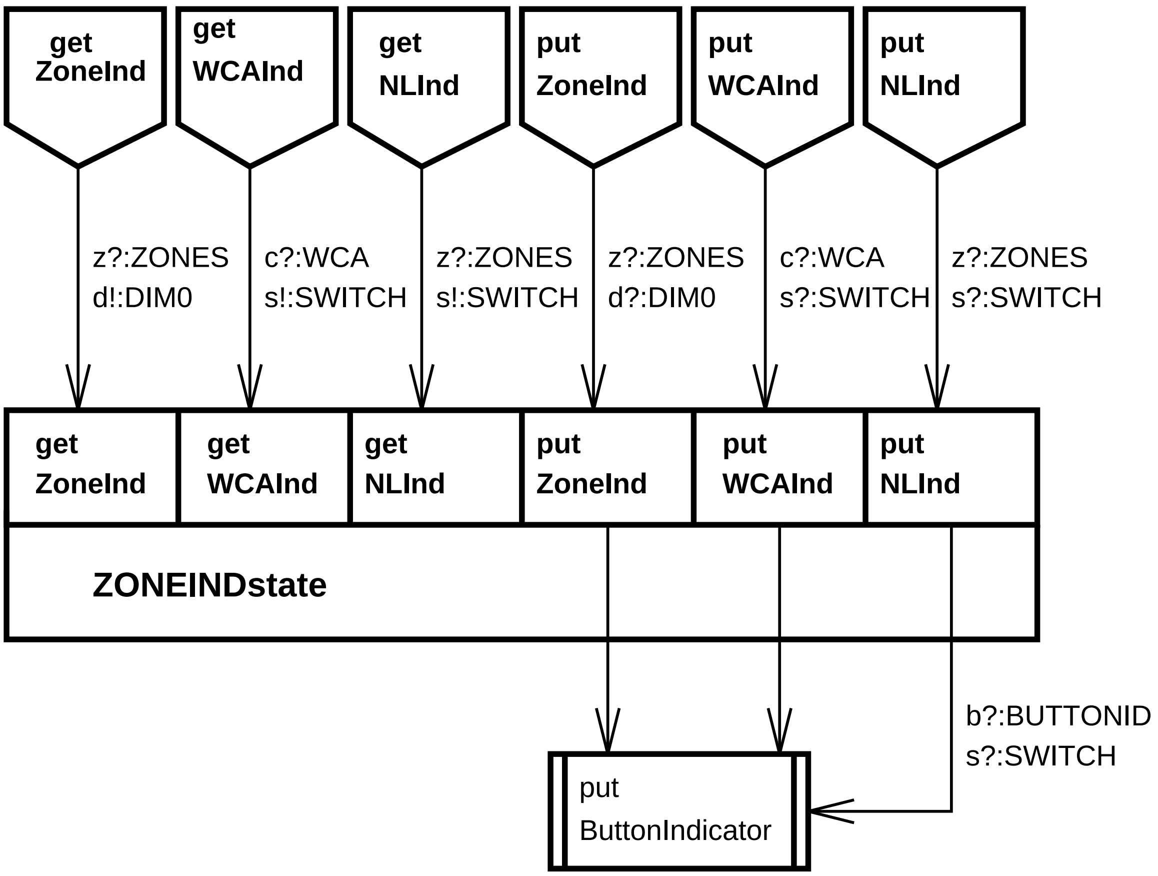

Figure 3: Data hiding module for the control of zone indicators and storage of indicator states.

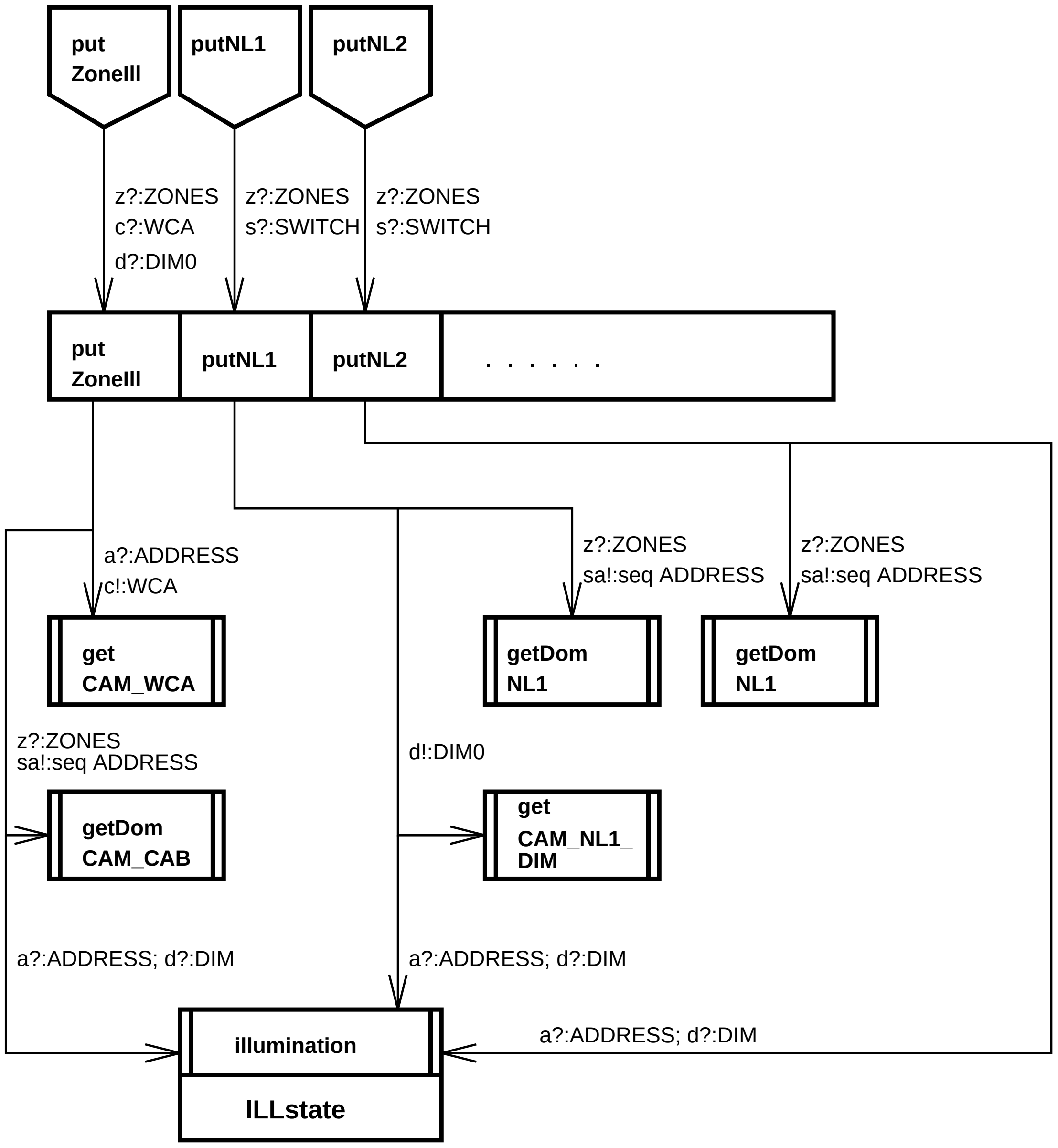

Figure 4: Data hiding module for the control of illumination units.

This gives rise to function putZoneIll in Figure 4, which takes $z?$, $c?$, $d?$ as input parameters, to switch all illumination units of zone $z?$ in column $c?$ to intensity $d?$. Instead of checking *wcaInd* for each address, the calling function rather checks for the active WCA columns using getWCAInd and then calls putZoneIll for each active column. Inspection of the other schemas shows that this definition of putZoneIll is appropriate:

$\{x : \mathrm{dom}(\mathrm{CAM\_CAB} \rhd \{z?\}) \bullet x \mapsto \mathit{off}\}$

in schema *ZONEILLopPassive* can be implemented by calling putZoneIll for each WCA-column and fixed $z?$. Expression

$\{x : \mathrm{dom}\,(\mathrm{CAM\_WCA} \rhd \{c?\}) \bullet x \mapsto \mathit{zoneInd}(\mathrm{CAM\_CAB}(x))\}$

in schema *WCAILLopActive* can be implemented by calling putZoneIll for each cabin zone with fixed column $c?$ and intensity d? = getZoneInd(z?).

Further inspection of the above mentioned operational schemas shows, that CAM tables are accessed in two ways, as expressions $x : \mathrm{dom}(\mathrm{CAM\_CAB} \rhd \{z?\})$ or as $\mathrm{CAM\_WCA}(x)$. This suggests, that two types of access functions should be provided for time-critical operations: Function getCAM_WCA inputs a DEUA address and returns the associated column value. For fast access this could be implemented by an array of the form $\{a? \mapsto c!\}$. Function getDomCAM_CAB inputs a zone $z?$ and returns the domain of the corresponding range restriction. This could be effectively implemented by storing a second CAM representation which keeps addresses of one zone in a specific array.

Similarly, functions putNL1, putNL2 are designed to implement expressions like

$\{x : \mathrm{dom}(\mathrm{CAM\_NL1} \rhd \{z?\}) \bullet x \mapsto \mathit{off}\}$

Note that it is not suitable to control both night light solutions by a single function, because schema *ZONEILLopActive* shows that they are not always switched simultaneously.

Top-Level Architecture Figure 5 shows the top-level architecture, which has been derived from schema equation *CIL* in Section 3.7.13. Since the separation of system control events from application data processing has already been performed on specification level, the transformation into the call tree's root is straight forward. Function CIL checks system events to decide whether DECOMPop, FAPctrl, AUTODIMop should be activated. If condition *UserIsBlocked* does not hold according to the system status, UserOp will be called to process the user-controllable functions. At the end of a CIL activation, the MAIN button indicator and the lavatory illumination are adapted to the actual state by means of MAINLAVadjust.

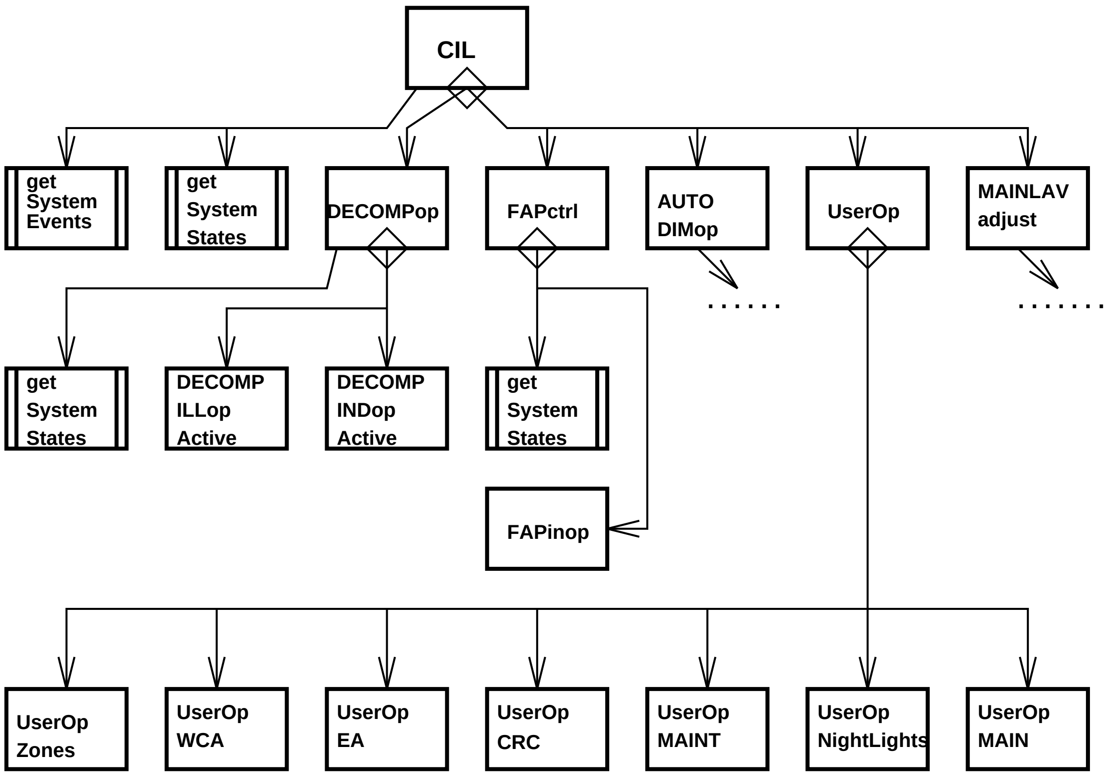

Figure 5: Top-level architecture for CIL software design.

Lower-Level Architecture Figure 6 shows transaction centers processing data conditions in the case of zone commands. UserOpZones checks for zone button events and calls ZONEop for each event, providing the associated zone and required intensity as input parameters. The structure of ZONEop has been directly derived from schema equation *ZONEop* in Section 3.7.4. Functions ZONEILLopActive, ZONEINDopActive, ... represent the data transformation functions which implement the corresponding schemas, making use of the basic packages for data manipulation.

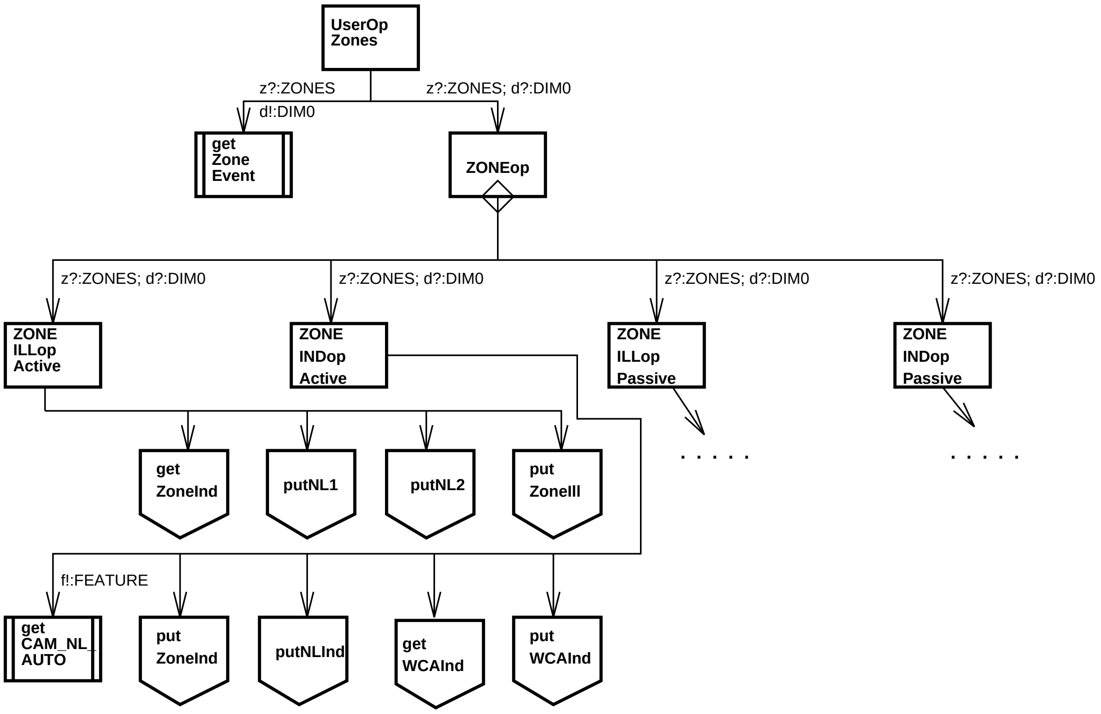

Figure 6: Zone operations and data manipulation functions.

Since all modules have been directly derived from schemas in the specification, the desired cohesion is automatically enforced by the design transformation.

5 Conclusion

In this case study the Z specification of the Airbus A330/340 cabin illumination function, together with the associated transformation process *system requirements* $\longrightarrow$ *software requirements specification* $\longrightarrow$ *software design* has been presented. To summarize the most important benefits derived from the application of Z, it can be said that

- The Z notation's powerful means to describe state spaces using mathematical data types provided the basis to develop a specification which reduced the overall complexity and could easily be verified.

- The understandability of the specification and its acceptable size was made possible by the Z constructs to produce modular and re-usable specifications.

- While the checking facilities of the CASE tool provided little help or no help at all to find logical errors in the specification, the Z type checker applied [2] uncovered a considerable amount of logical errors in the early phases of the specifications.

- The techniques applied to derive design from the Z specification considerably reduced the effort during the design and implementation phases. The development of a design architecture and of implementable data structures and algorithms from the specification turned out to be rather straight forward.

- The Z specification could easily be applied to define trustworthy and significant test cases for integration testing. The concept to derive test cases from Z specifications in a systematic way has been described in [5]. It also allows to automize a considerable amount of the test evaluation effort. To this end DST have developed a prototype tool in cooperation with Christian Albrechts Universität Kiel, which is evaluated at the moment [5, 6].

Conventional CASE methods turned out to be insufficient to tackle the more complex functions of a safety-critical application system. This problem can be overcome by combining CASE methods with formal specification and verification methods. This approach has been described in [9, 10]. At present it is further extended in cooperation between the Christian Albrechts Universität Kiel, DST and the second author.

We regard the application of Z as an important factor for the successful completion of the CIDS applications developed by DST. It did not only provide a technique to develop trustworthy specifications, but also helped to reduce the costs for implementation and integration, because the specification allowed to derive a software design with minimum complexity.

Acknowledgements We would like to thank Manfred Endreß from Deutsche Aerospace Airbus GmbH for the excellent cooperation during all development phases of the CIDS project.

The final version of this article has been produced, while the second author was visiting lecturer at the University of Capetown. We wish to express our gratitude to Chris Brink and his group who provided both the opportunity and the stimulus to finish this work.

References

[1] DA Airbus Industrie No 2370M1F000101: *Cabin Intercommunication Data System System Specification, A330/340, CIDS* Volume I, Issue 5 (1993)

[2] Using Z and DST-fuzz, an Introduction. DST Deutsche System-Technik GmbH, Kiel (1993).

[3] RTCA DO178B: *Development considerations in airborne computer systems.* (1993).

[4] D. J. Hatley, I. A. Pirbhai: *Strategies for Real-Time System Specification.* New York, Dorset House (1987).

[5] H.-M. Hörcher, J. Peleska: The Role of Formal Specifications in Software Test. Tutorial, held at the *FME'94 Symposium* Barcelona (1994).

[6] E. Mikk: Translation of Z specifications into executable code for automatic evaluation of test results. Technical Report, Christian Albrechts Universität zu Kiel (submitted to TAPSOFT '94).

[7] C. C. Morgan: *Programming from Specifications.* Prentice-Hall, Englewood Cliffs NJ (1990).

[8] C. C. Morgan, K. A. Robinson, P. H. B. Gardiner: On the refinement calculus. *Oxford University Computing Laboratory.* PRG-70 (1989).

[9] J. Peleska: *Formales Software-Engineering mit Strukturierten Methoden und Z: nachweisbare Korrektheit fuer sicherheitsrelevante Anwendungen.* in H.-J. Scheibl (ed.): Software-Entwicklung - Methoden, Werkzeuge, Erfahrungen '93. Technischen Akademie Esslingen (1993). (English version available as *Formal Software Engineering, Z and Structured Methods – provable correctness for safety-critical systems.* Technical Report, DST Deutsche System-Technik GmbH, Kiel (1992).)

[10] J. Peleska: CSP, Formal Software Engineering and the Development of Fault-Tolerant Systems. In J. Vytopil (ed.): *Formal Techniques in Real-Time and Fault-Tolerant Systems.* Kluwer Academic Publishers (1993).

[11] J. M. Spivey: *The Z Notation: A Reference Manual*, Prentice-Hall International, 1989

[12] P. T. Ward, S. . J. Mellor: *Structured Development for Real-Time Systems.* (3 vols), Yourdon Press Computing Series, Prentice-Hall, Englewood Cliffs NJ (1985).

[13] Wordsworth J. B.: *Software Development with Z.* Addison-Wesley,Wokingham (U.K.) (1992).