

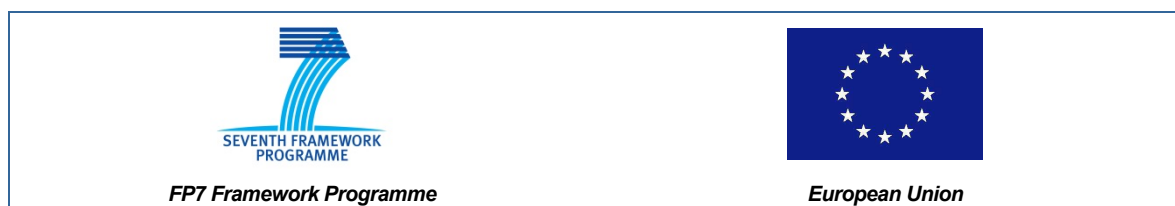
## D.2.2-PROOF OF CONCEPT APPLICATION IN SMART ENERGY DOMAIN

---

### ADVANCE

**Grant Agreement:** 287563  
**Date:** 27/09/2012  
**Pages:** 29  
**Status:** Final  
**Access:** Access List  
**Reference:** CSWT-EUADV-2012-TNR-00180  
**Issue:** 1



#### Partners / Clients:



#### Consortium Members:



# PROOF OF CONCEPT APPLICATION IN SMART ENERGY DOMAIN ADVANCE

Approval			
Name	Function	Signature	Date
Luke Walsh	Project Manager		27-09-2012
José Reis	Quality Manager		27-09-2012

Authors and Contributors			
Name	Contact	Description	Date
Brett Bicknell	<a href="mailto:bbicknell@critical-software.co.uk">bbicknell@critical-software.co.uk</a>	Author	27/09/2012
José Reis	<a href="mailto:jreis@critical-software.co.uk">jreis@critical-software.co.uk</a>	Contributor	27/09/2012
Luke Walsh	<a href="mailto:lwalsh@critical-software.co.uk">lwalsh@critical-software.co.uk</a>	Reviewer	31/08/2012
Michael Leuschel	<a href="mailto:leuschel@cs.uni-duesseldorf.de">leuschel@cs.uni-duesseldorf.de</a>	Reviewer	12/09/2012

Access List
<b>Internal Access</b>
Project Team, Engineering Department
<b>External Access</b>
Public Document
The contents of this document are under copyright of Critical Software Technologies. It is released on condition that it shall not be copied in whole, in part or otherwise reproduced (whether by photographic, or any other method) and the contents therefore shall not be divulged to any person other than that of the addressee (save to other authorized offices of his organization having need to know such contents, for the purpose for which disclosure is made) without prior written consent of submitting company.

Revision History			
Issue	Date	Description	Author
1	27/09/2012	First Issue	Brett Bicknell

<b>CRITICAL SOFTWARE TECHNOLOGIES LTD</b> 2 VENTURE ROAD SOUTHAMPTON SCIENCE PARK – CHILWORTH SOUTHAMPTON - SO16 7NP – UNITED KINGDOM	<b>CRITICAL SOFTWARE, S.A.</b> PARQUE INDUSTRIAL DE TAVEIRO, LOTE 48, 3045-504 COIMBRA PORTUGAL
--	--

# TABLE OF CONTENTS

<b>1. INTRODUCTION.....</b>	<b>3</b>
1.1 OBJECTIVE.....	3
1.2 AUDIENCE .....	3
1.3 DEFINITIONS AND ACRONYMS.....	3
1.4 REQUIREMENT CLASSIFICATION .....	3
1.5 DOCUMENT STRUCTURE .....	3
<b>2. DOCUMENTS.....</b>	<b>4</b>
2.1 APPLICABLE DOCUMENTS .....	4
<b>3. SYSTEM OVERVIEW .....</b>	<b>5</b>
3.1 REQUIREMENTS DOCUMENTATION.....	6
<b>4. OVERVIEW OF FORMAL MODELLING STRATEGY .....</b>	<b>7</b>
<b>5. DECOMPOSITION OF A DISTRIBUTED NETWORK SYSTEM.....</b>	<b>9</b>
5.1 TOP LEVEL MODEL.....	9
5.1.1 <i>Model Detail and Refinement</i> .....	9
5.1.2 <i>Shared Event Decomposition</i> .....	11
5.1.3 <i>Adherence to Previously Stated Requirements and Assumptions</i> .....	12
5.2 DATA CENTRE.....	14
5.3 SIU .....	18
5.3.1 <i>Refinement strategy and requirements</i> .....	18
5.3.2 <i>Initial Results from the Formal Modelling</i> .....	20
5.3.3 <i>Compatibility with Previously Stated Assumptions</i> .....	22
5.4 NETWORK .....	23
5.4.1 <i>Re-composing the Developed Models with the Network</i> .....	23
<b>6. CONCLUSIONS AND NEXT STEPS.....</b>	<b>26</b>
6.1 UPDATE ON SUITABILITY FOR INFLUENCING TOOL DEVELOPMENT.....	26
6.2 UPDATE ON EXPECTED OUTCOMES AND SUCCESS CRITERIA .....	27
<b>APPENDIX A. UML-B KEY .....</b>	<b>29</b>

## 1. Introduction

### 1.1 Objective

This document provides a report on the proof of concept application of ADVANCE methods to the smart energy case study, comprising of strategy descriptions, model examples and current conclusions drawn from the work. The case study is part of work package 2 of ADVANCE.

### 1.2 Audience

Those involved or interested in the case study, including the ADVANCE consortium.

### 1.3 Definitions and acronyms

Table 1 presents the list of acronyms used throughout the present document.

Acronyms	Description
SIU	Sensor Interface Unit
WP	Work Package
DC	Data Centre

Table 1: Table of acronyms

### 1.4 Requirement Classification

Table 2 presents the system of identification created for the requirements listed throughout the document.

KEY	Description
SGCS-GEN-FUN-xxx	General (functional) requirements which encompass the entire case study.
SGCS-GEN-NON-xxx	General (non-functional) requirements which encompass the entire case study.
SGCS-GEN-ASM-xxx	Assumptions associated with the general requirements.
SGCS-LVM-FUN-xxx	Functional requirements specific to the sensor interface unit.
SGCS-LVM-NON-xxx	Non-functional requirements specific to the sensor interface unit.

Table 2 : Table of requirement classification

### 1.5 Document structure

Section 1 introduces the document.

Section 2 (Documents) presents the list of applicable and reference documents as well as the documentation hierarchy this document belongs to.

Section 3 (System Overview) gives an overview of the system which is modelled during the proof of concept.

Section 4 (Overview of Formal Modelling Strategy) provides an overview of the formal modelling strategy utilised during the proof of concept.

Section 5 (Decomposition of a Distributed Network System) details and gives examples of the modelling process described in Section 4, and the application to the proof of concept.

Section 6 (Conclusions and Next Steps) lists the conclusions of the process described in the document, and presents the proposed future progression of the case study.

## 2. Documents

This section presents the list of applicable and reference documents as well as the documentation hierarchy to which this document belongs.

### 2.1 Applicable documents

Table 3 presents the list of the documents that are applicable to the present document. A document is considered applicable if it contains provisions that through reference in this document incorporate additional provisions to this document [ECSS-P-001B].

Applicable document	Document number	Issue
[AD-1] Smart Grid Case Study Definition, <i>Critical Software Technologies</i> , December 19 <sup>th</sup> 2011	CSWT-EUADV-2011-SPC-00621	1
[AD-2] Shared Event Composition/Decomposition in Event-B, <i>University of Southampton</i> , November 2010	Lecture Notes in Computer Science, 2012, Volume 6957/2012, 122-141, DOI: 10.1007/978-3-642-25271-6_7	

**Table 3: Applicable documents**

### 3. System Overview

The case study initially focuses on an example substation solution, specified in section 4.1 of [AD-1], and referred to in this document as a Sensor Interface Unit (SIU). When in operation there will be a number of SIU installations, each of which will take readings from multiple sensors, as indicated in Figure 1. The SIUs will communicate with a single data centre. The proof of concept broadens the modelling scope to include this central data centre, which provides the overall control and data acquisition for the set of SIUs.

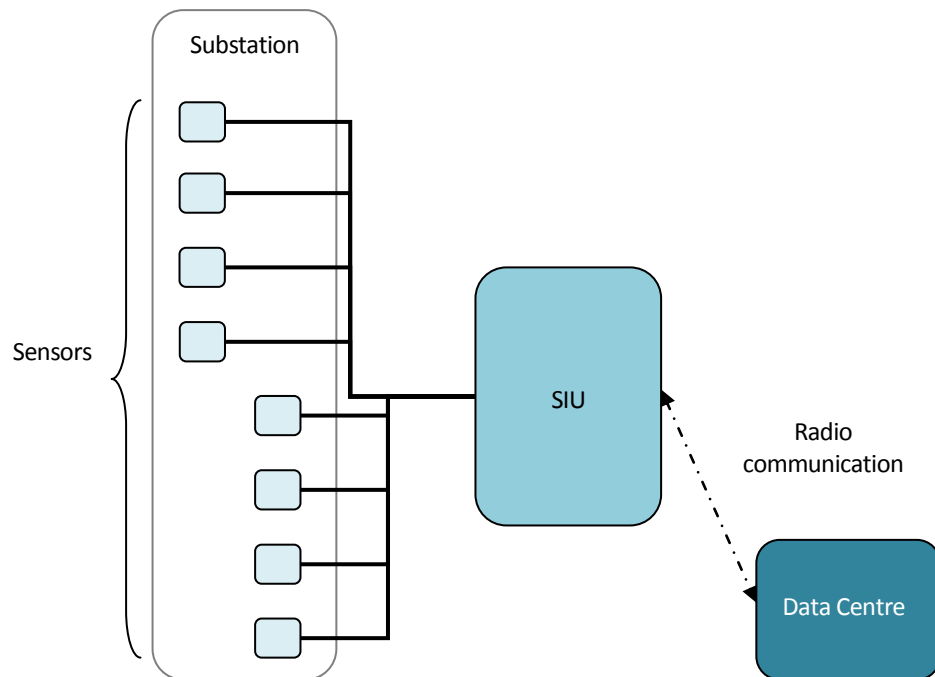


Figure 1 : A diagrammatic representation of the SIU system

The critical elements of the system to be formally modelled during the case study comprise of:

- Memory management within each SIU.
- The validation of data and the raising of alerts upon exceeding thresholds in the sensor readings.
- The management of network faults.
- The capability of the data centre to cope with a scaling number of connected SIUs.

The challenge of specifying these processes, and behaviour of the combined system, stems from remote communication. Each SIU can only communicate to the data centre via wireless methods and vice-versa, which can be susceptible to fault or delay. Commands or reports can arrive late or be lost entirely and protocols must be in place to cope with these events.

Although this system does not provide a description of the behaviour of smart meters specifically, it does provide a suitable proof of concept. The processes, protocols, and

potential issues of the distributed network system provided by this example will be representative of the same behaviour present within a smart grid structure, and thus can be reused in this context. It is also the preference of our industrial collaborative for the case study that this particular system is modelled, so that any potential issues found can be addressed before further development of the SIU.

### 3.1 Requirements Documentation

The specification for the proof of concept is provided by the requirements documentation for the SIU. The requirements text is not included in this document as it contains proprietary third party information.

## 4. Overview of Formal Modelling Strategy

The modelling strategy is based heavily upon the idea of decomposition. The combined system naturally lends itself to decomposition into separate models for the data centre, network, and monitoring/control units (in this case the SIU). Each of these models are developed independently and then composed again at later points to check that the composed system is a refinement of the original combined system. This is a suitable approach for distributed networks in general - and provides a strategy which can be reused - as the model for the combined system will be the same for any system of this type. It is only when work begins on the separate decomposed models that the attributes specific to each system are introduced. The decomposition process is displayed in Figure 2.

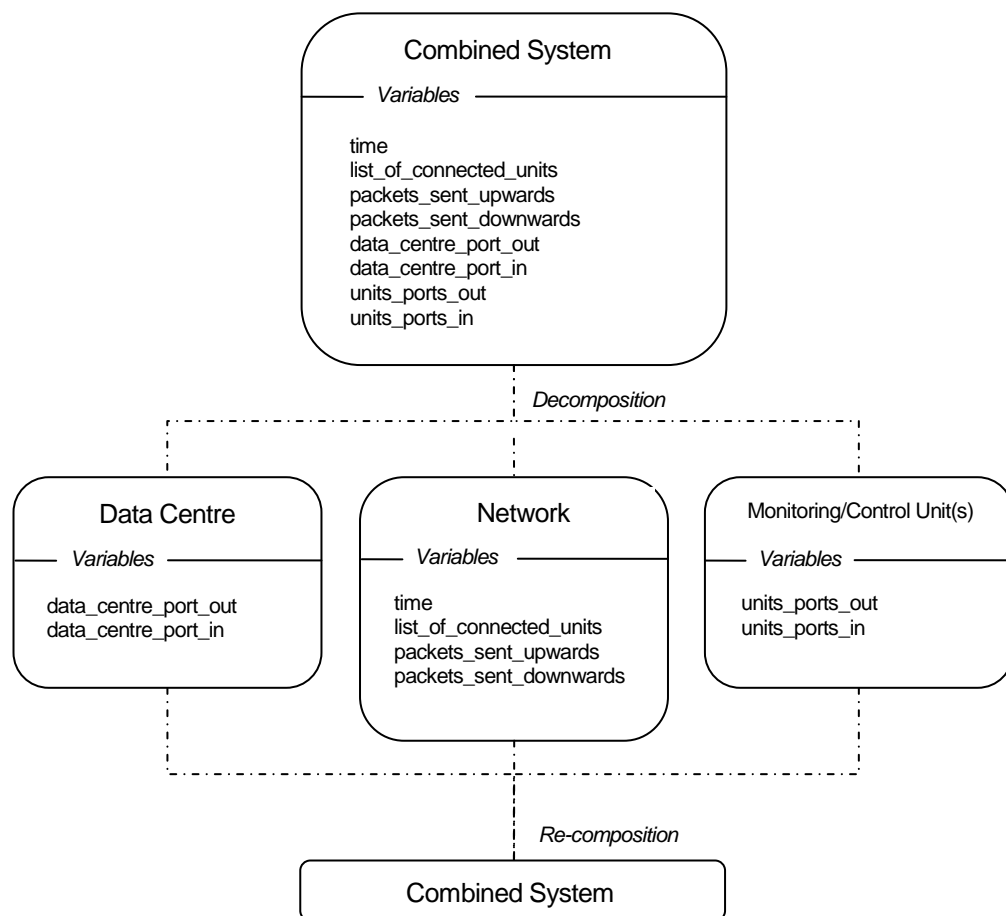


Figure 2 : Decomposition strategy

The decomposition style chosen is shared event decomposition (see [AD-2]), such that the variables in the combined system are split with each assigned to a decomposed part. As shown in Figure 2 the only elements from the combined system that the data centre and monitoring units are required to interact with are the input and output ports for that unit; the rest of the functionality is assigned to the network model. This allows for the internal processes of the SIU and data centre to be introduced and verified separately, as the behaviour of each will only depend on the current values present on the input and output ports, with no knowledge of the intricacies of the behaviour of the network. Conversely, the



fact that the network only sees an input and output port for each unit and the data centre, without any knowledge of internal workings, means that multiple versions and models of the data centre and monitoring/control units can be composed with the same network model. This again lends the process to reusability, but also allows for the comparison of the behaviour of different monitoring and control units, and which requirements they satisfy.

## 5. Decomposition of a Distributed Network System

### 5.1 Top Level Model

#### 5.1.1 Model Detail and Refinement

A simplified, natural language version of the combined system model is presented in Figure 3.

<b>Variables</b>	
time	(the current time of the system)
connected_units	(the list of connected monitoring/control units)
packets_up	(the set of packets moving through the network towards the data centre)
packets_down	(the set of packets moving through the network towards a unit)
DC_port_out	DC_port_in (the output/input ports of the data centre)
UNIT_ports_out	UNIT_ports_in (the set of all output/input ports of the units)
<b>Events</b>	
<b>Tick</b>	
<b>Where</b>	
- All data due to arrive has been copied to UNIT_ports_in and DC_port_in	
<b>Then</b>	
- Increase time by one tick (time = time + 1)	
- For each piece of data in DC_port_out, copy this to packets_down to arrive after some delay	
- For each piece of data in UNIT_ports_out, copy this to packets_up to arrive after some delay	
- Clear UNIT_ports_out and DC_port_out	
<b>UNIT_ReceiveData</b>	
<b>Where</b>	
- There are packets due to arrive at some unit(s) at the current time in packets_down	
<b>Then</b>	
- Copy the data for time from packets_down to UNIT_ports_in	
<b>DC_ReceiveData</b>	
<b>Where</b>	
- There are packets due to arrive at the data centre at the current time in packets_up	
<b>Then</b>	
- Copy the data for time from packets_up to DC_port_in	
<b>UNIT_SendData</b>	
<b>Where</b>	
- There is not already data on the port(s) (UNIT_ports_out is empty for the selected unit(s))	
<b>Then</b>	
- Add the selected message to UNIT_ports_out	
<b>DC_SendData</b>	
<b>Where</b>	
- There is not already data on the port (DC_port_out is empty)	
<b>Then</b>	
- Add the selected message to DC_port_out	
<b>AddUnit</b>	
<b>Then</b>	
- Add the selected unit to connected_units (if not already connected)	
<b>RemoveUnit</b>	
<b>Then</b>	
- Remove the selected unit from connected_units (as long as it is connected)	

Figure 3 : Natural language pseudo code of the Event-B for the combined system model

For the development presented in Section 5.3, the ‘units’ referred to in Figure 3 will be a set of SIUs, although it can be seen that the model is generic before decomposition and could apply to any communicating unit. At this abstract level the majority of the operation of the network is represented through the variables *packets\_up* and *packets\_down*. Each represents a set of packets related to a nominal delay which dictates their arrival time at the data centre and a connected unit respectively. Each connected unit and the data centre can send data by adding packets to these sets through the events *UNIT\_SendData* and *DC\_SendData*. Time will advance discretely through the *Tick* event until the current time matches that of an arrival time of a packet in *packets\_up* or *packets\_down*, at which time the *UNIT\_ReceiveData* or *DC\_ReceiveData* event will have to occur before time can advance further.

As mentioned in the previous section, the decomposition is set up so that the only variables assigned to the monitoring/control units and data centre are the respective input and output ports. Therefore, as can be inferred from Figure 3, the main events to be refined in the monitoring/control units and data centre models will be the send and receive events in each case. The time is mainly an artefact of the network; each unit and the data centre respond according to the data that is received on the input port (and, in the case of the data centre, commands received from an operator), and the network model dictates the progress of all the communications. Along with this the network model will describe the creation and removal of connections between the data centre and monitoring/control units. Therefore the main events that will be refined in the network model are those which advance the time (*tick*), and those which add and remove monitoring/control units (*AddUnit/RemoveUnit*).

The combined system goes through multiple refinement levels to reach the end result represented in Figure 3. Although these are not strictly necessary, they help to introduce the detail in steps and separate out the complexity and number of proof obligations. The refinement strategy is shown in Table 4, which refers to the variable and event identifiers in Figure 3. The pseudo code in Figure 3 is representative of the final refinement level in Table 4, M6.

Machine	Introduces	Included variables							Included events							
		<i>time</i>	<i>connected_units</i>	<i>UNIT_ports_out</i>	<i>DC_port_out</i>	<i>packets_down</i>	<i>packets_up</i>	<i>UNIT_ports_in</i>	<i>DC_port_in</i>	<i>Tick</i>	<i>AddUnit</i>	<i>RemoveUnit</i>	<i>UNIT_SendData</i>	<i>DC_SendData</i>	<i>UNIT_ReceiveData</i>	<i>DC_ReceiveData</i>
<b>M0</b>	<i>Tick event and global time</i>	█								█						
<b>M1</b>	<i>Adding and removing units</i>	█	█							█	█	█				
<b>M2</b>	<i>Data sent out from a unit</i>	█	█	█					█	█	█	█				
<b>M3</b>	<i>Data sent out from the data centre</i>	█	█	█	█				█	█	█	█	█			
<b>M4</b>	<i>Mechanism by which data is sent through the network</i>	█	█	█	█	█			█	█	█	█	█			
<b>M5</b>	<i>Receiving data at a unit</i>	█	█	█	█	█	█		█	█	█	█	█	█		

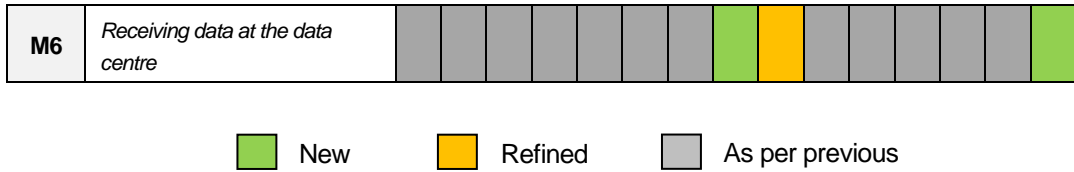


Table 4 : Refinement strategy for top-level model

5.1.2 Shared Event Decomposition

The allocation of variables to each of the decomposed parts, as indicated in Figure 2, result in the event distribution shown in Table 5 below.

	tick	addUnit	removeUnit	UNIT_SendData	UNIT_ReceiveData	DC_SendData	DC_ReceiveData
Data Centre							
Network							
Monitoring/Control Units							

■ Represents an included event

Table 5 : Distribution of events within decomposed parts

As mentioned in the previous section, the send and receive events will be extended through the modelling of the internal processes of the data centre and monitoring/control units. However, the *tick* event is also prevalent in each so that any recurring processes can be added by refining this event. The *removeUnit* event is also included in both the models for the data centre and monitoring/control units so that any elements in Event-B relations or functions that rely on the list of connected units can be removed at the same time the unit is removed. No real functionality will be included in each refinement of the event as this will be implemented in the network refinement.

The network requires all of the events in Table 5, as it is responsible for passing the data between one send event and another receive event. The *addUnit* and *removeUnit* events will be refined in the network model for two main reasons:

1. The detail of the communication structure and hierarchy will be introduced in the refinement of the network. This will manifest mainly as extra guards in the *addUnit* event, due to increased limitations with regard to when and how a unit can be connected. These will be introduced in order to create a more realistic network model.
2. Failures of the network or connectivity will be modelled via refinement of the *removeUnit* event. The refinement of this event will also include the 'safe' removal

of a unit which, similar to the *addUnit* event, will inherit extra guards due to restrictions on when and how a unit can be removed.

### 5.1.3 Adherence to Previously Stated Requirements and Assumptions

In [AD-1] several high level requirements were stated which the case study will aim to verify. Those of particular relevance to the top-level model described are:

	Description
SGCS-GEN-FUN-002	It shall be possible to include the addition and removal of meters and stations to and from the network without compromising the other requirements.
Notes:	This shall be possible without compromising the security or reliability of the network.

	Description
SGCS-GEN-NON-004	The model shall be verified regardless of how the smart grid scales.
Notes:	

These are integral to the top-level model as events which add and remove monitoring/control units are introduced from the very initial stages. Therefore any further functionality and requirements added to the model have to be compatible with these events. This addresses both the verification of the addition and removal of meters, and the scalability of the network. The latter is achieved due to the fact that the model does not identify a constant number of connected units, but instead a dynamic set. The expansion of this set can represent the scaling of the grid.

The remaining requirements, SGCS-GEN-FUN-001 and SGCS-GEN-NON-003, are more specific and are therefore not addressed at this level; the associated functionality will be implemented later. Although, it is still true that none of the current events invalid these requirements. These requirements are listed below for reference:

	Description
SGCS-GEN-FUN-001	There shall be a set of devices which are guaranteed to supply constant power at all times.
Notes:	This is necessary in a safety-critical environment.

	Description
SGCS-GEN-NON-003	There shall be levels of authorisation within the system.
Notes:	Only with certain authorisation level(s) can a device change the power supplied to a meter, read the data from a meter, and so on.

All of the general assumptions listed in Section 3.1.1 within [AD-1] remain valid with the model, with the exception of the below assumption which will be implemented in the network model:

	Description
SGCS-GEN-ASM-002	Any damage to the physical network is ignored.
Notes:	An example of damage to the physical network is a broken power line. Later on it may be possible for faults of this manner to be injected into the system during simulation.

In particular, the following assumption is kept as each of the connected units is part of a single set, where the operations on each element of the set are identical:

	Description
SGCS-GEN-ASM-001	Each meter and each station is identical.
Notes:	

The remaining assumptions are also listed below for reference:

	Description
SGCS-GEN-ASM-003	When verifying properties (such as the requirement that devices in safety-critical environments cannot be cut off or have their supply inadvertently changed) only the potential faults or limitations within the software system itself are considered, and not any changes to the physical devices and network or the overall power supply in the network (power blackouts, etc).
Notes:	Later this will be an important point to consider in the context of a cyber-physical system.

	Description
SGCS-GEN-ASM-004	The security of the network will be validated assuming that the devices already present will not be modified or reprogrammed in any way.
Notes:	

	Description
SGCS-GEN-ASM-005	The software and network protocols are well enough protected or encrypted on each device that they cannot be replicated by outside agencies.
Notes:	

## 5.2 Data Centre

The requirements document for the SIU provides enough detail about the supporting data centre so that a model can be constructed, provided some assumptions are made. This is a valuable exercise as it is only with a suitably detailed model of the data centre that the interaction of the SIU with the network can be properly examined and verified. In this case the main purpose of modelling the data centre is to aid the verification of the SIU, rather than verify properties of the data centre itself. Regardless, some requirements specific to the data centre are modelled through this exercise, and there is no reason that in a different application of this methodology, properties of the data centre cannot also be verified.

The model for the data centre was created using the UML-B and Statemachine plug-in tools for Rodin, which provide the automatic translation of UML and state machine diagrams to Event-B. This was deemed an advantageous approach as the internal processes in the data centre are easiest represented using a process flow. The refinement strategy is shown in the package diagram, Figure 4. Each Event-B machine, indicated in blue, has a single state machine which refines the abstract state machine from the previous step, as well as a group of applicable global variables and invariants - also specified in UML-B - needed for the functionality at that level. For reference to the elements used within the figures in this section, see Appendix A.

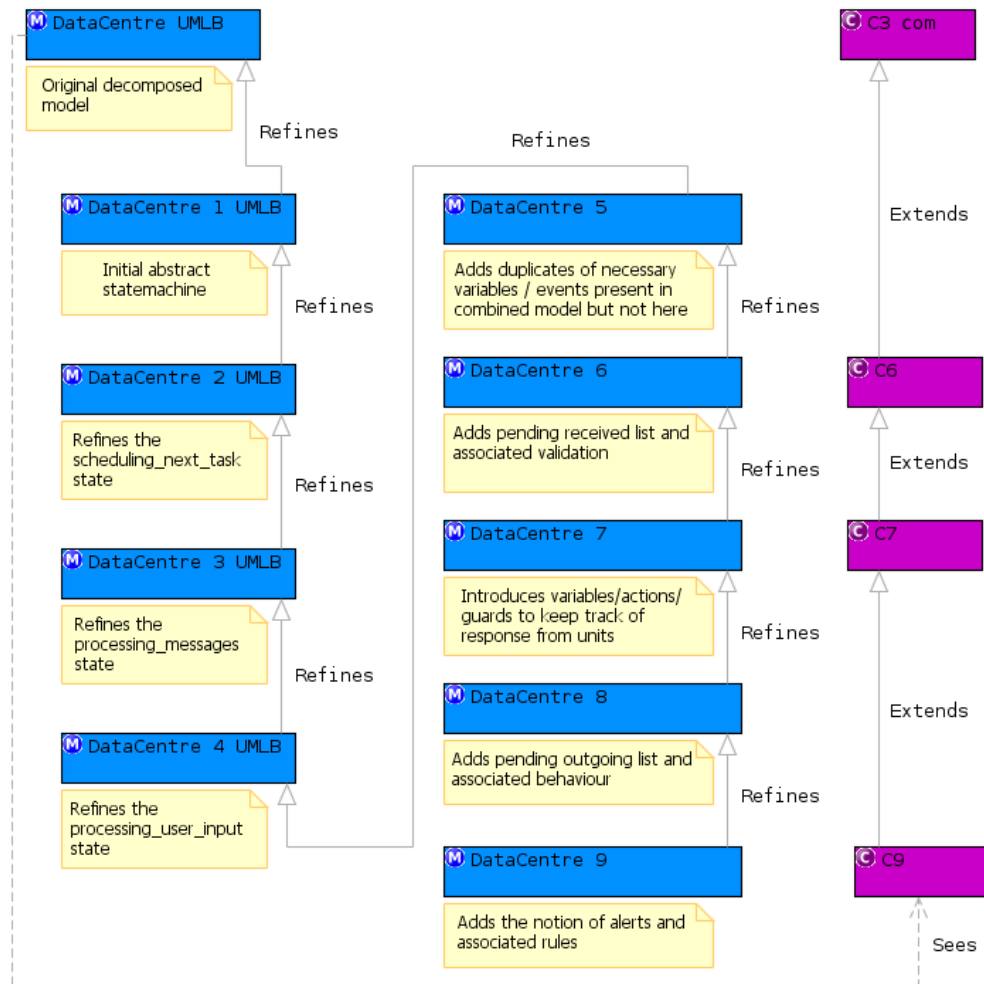


Figure 4 : Refinement strategy for the data centre model

The UML-B and Statemachine tools also encompass the notion of refinement. This was utilised by first specifying a top-level state machine representing the entire process list of the data centre in an abstract form, and then refining this in several stages by introducing nested state machines to elaborate abstract states. The contents of the final refined machine in Figure 4, *DataCentre\_9*, are displayed in Figure 5.

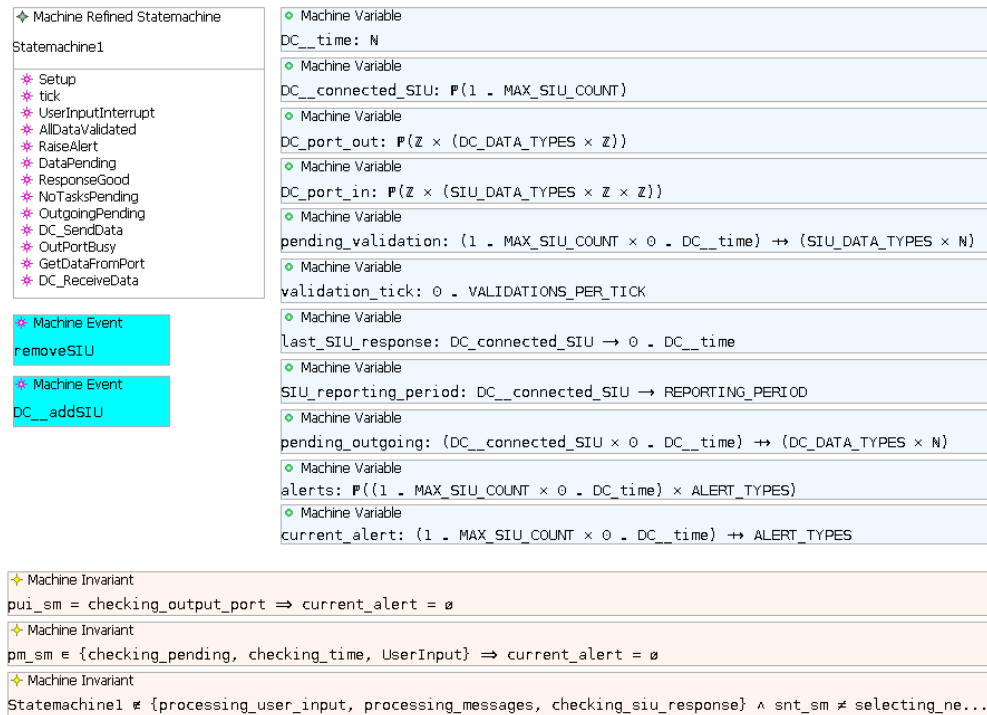


Figure 5 : UML-B diagram for DataCentre\_9

Introducing a state machine after decomposition is more difficult compared to using state machines from the initial stages. This is due to the fact that the UML-B model has to be a valid refinement of the original decomposed model if it is to be used in the eventual composition. This was solved by creating an exact copy of the original decomposed model in UML-B form (*DataCentre\_UMLB* in Figure 4). Once this is translated to Event-B it can be proven that it refines the original decomposed part; and therefore that all of the other machines in Figure 4 also refine the decomposed part.

The final refinement of the state machine from *DataCentre\_9* is shown in Figure 6. It should be noted that each of the states *processing\_messages*, *scheduling\_next\_task* and *processing\_user\_input* contain nested state machines. These elaborate the initial states in *DataCentre\_1\_UMLB* and correspond to the first four levels of refinement in Figure 4.



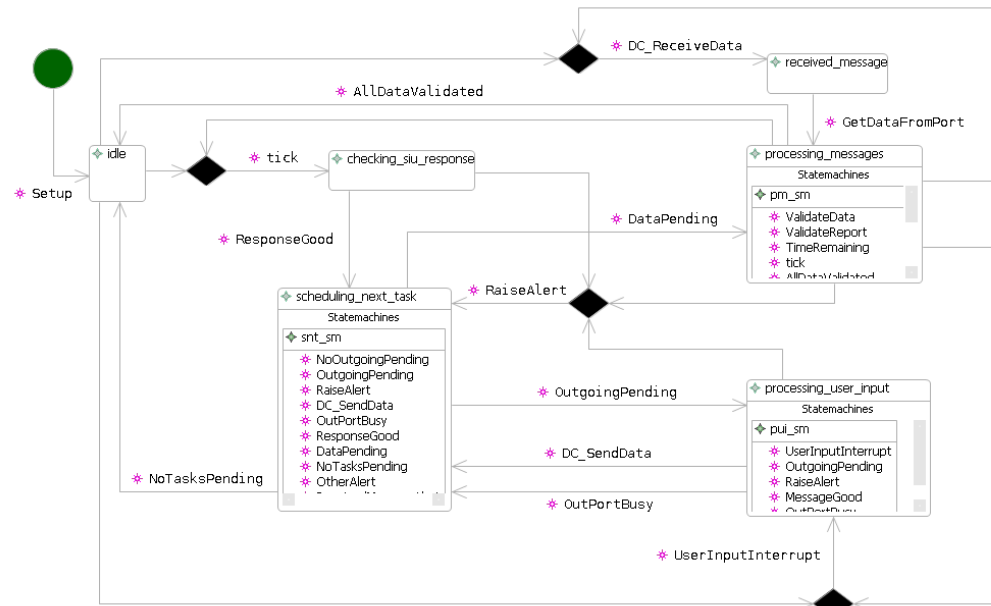


Figure 6: the final refinement of the state machine from DataCentre\_9

During the translation to Event-B, each state in the state machine is translated into a Boolean argument and each transition is translated into an event; this is demonstrated in Figure 7. It is possible to add any guards, actions or parameters for these events to the transitions using the UML-B tool.

```

invariants
@Statemachine1.type Statemachine1 ∈ Statemachine1_STATES

events
event INITIALISATION
  then
  @Statemachine1.init Statemachine1 = idle
  end

event ResponseGood
  where
  @Statemachine1.isin_checking_siu_response Statemachine1 = checking_siu_response
  then
  @Statemachine1.enterState_scheduling_next_task Statemachine1 = scheduling_next_task
  end

event AllDataValidated
  where
  @Statemachine1.isin_processing_messages Statemachine1 = processing_messages
  then
  @Statemachine1.enterState_idle Statemachine1 = idle
  end

event UserInputInterrupt
  where
  @UserInputInterrupt_disjunct (Statemachine1 = processing_messages) ∨ (Statemachine1 = idle)
  then
  @Statemachine1.enterState_processing_user_input Statemachine1 = processing_user_input
  end
    
```

Figure 7 : A section of the automatically translated Event-B from the state machine

As can be seen in Figure 6, the events present in the original decomposed data centre machine (*tick*, *DC\_SendData*, *DC\_ReceiveData*, and *removeUnit*: see Table 5) are refined directly through the state machine, with the exception of the *removeUnit* event (which is labelled as *removeSIU*). The *removeSIU* event is not refined via the state machine - although, as shown in Figure 5, it is refined through the UML-B – due to the fact that the event could occur at any time and is not necessarily anticipated. For example, when there is a connection failure in the network, the event will not be considered part of the process flow.

The nested state machines which form the first stages of the refinement are shown in Figure 8. Additional functionality is added in these steps, with the only restriction being that the nested state machines must elaborate the same input and output transitions as the original state. This use of refinement allows for each of the nested state machines to be validated with the abstract model separately, so that it is clear where any issues are present.

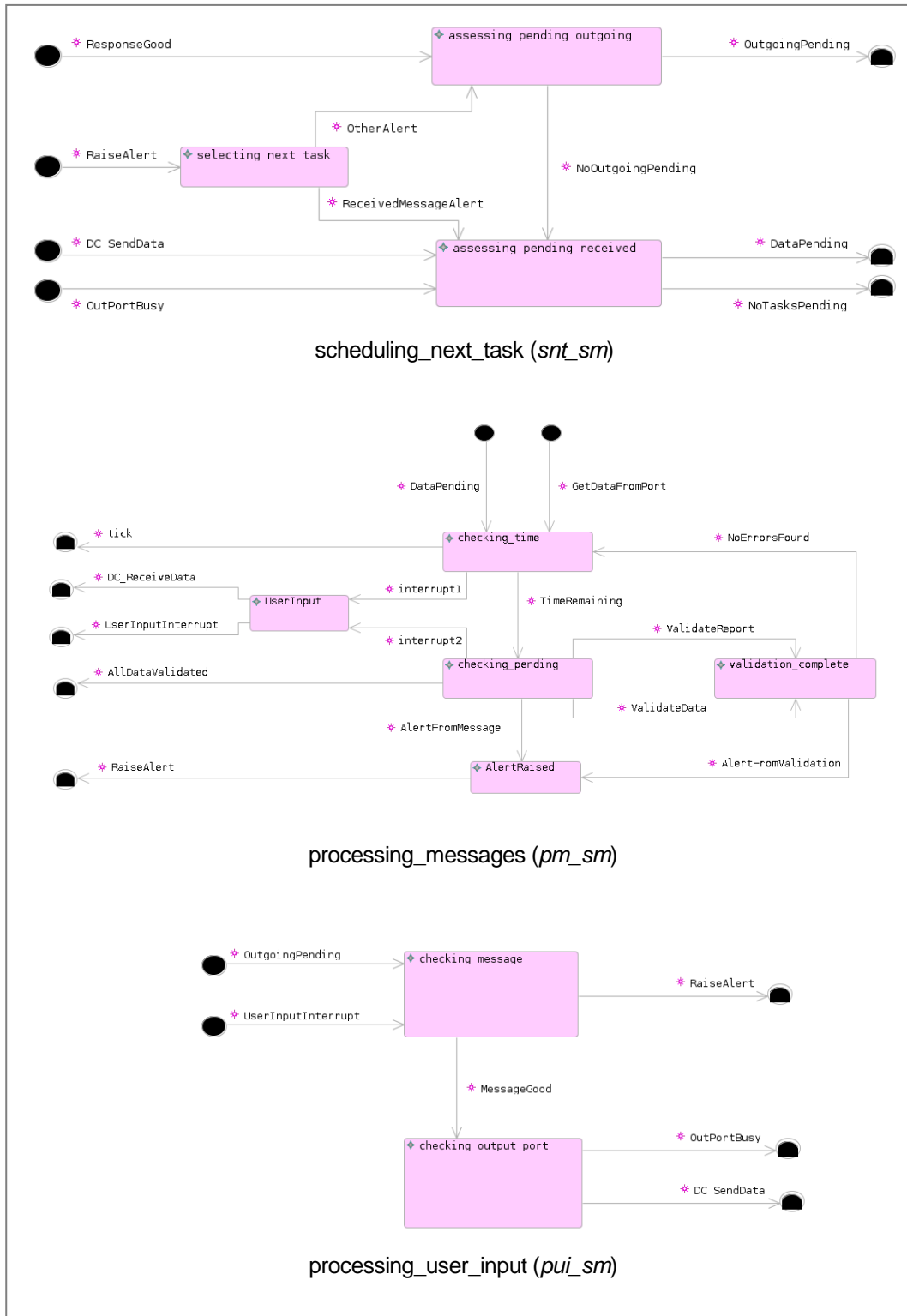


Figure 8 : Each of the nested state machines with the elaborated state listed below

## 5.3 SIU

### 5.3.1 Refinement strategy and requirements

The majority of the model for the SIU focuses on the issue of memory management, and how the external commands from the data centre affect this. The initial refinement strategy is shown in Table 6 below.

Machine	Introduces
SIU_1	Adds events for storing and removing measurement data and data reports. Adds the notion of the total memory occupied.
SIU_2	Adds variables which represent the current measurement and reporting periods for each SIU, and the actions to change these if a message is received with a command.
SIU_3	Refines the existing events so that at each reporting and measurement interval, data or a report is stored in the memory.
SIU_4	Adds a timestamp to each report and data item.
SIU_5	Adds the reaction of the SIU to commands to enable or disable the automatic transmission of reports.
SIU_6	Adds the reaction of the SIU to commands to change the alert threshold or toggle alerts on or off.
SIU_7	Introduces the capability of the SIU to send messages to the data centre. This includes: <ul style="list-style-type: none"> <li>- The ability to issue alerts to the data centre upon exceeding alert thresholds.</li> <li>- The ability to send stored reports and data back to the data centre upon request.</li> <li>- The periodic transmission of reports, if this option is enabled.</li> </ul>

Table 6 : Refinement strategy for the SIU model

The first four levels of refinement build up the memory management processes. These correspond to processes the SIU will perform automatically without the need for external input. The latter stages of refinement detail the reaction of the SIU to external commands from the data centre. This order is chosen because the function of the external commands is to alter the behaviour of the automatic internal processes. An example of specific Event-B constructs which capture some of the requirements describing the behaviour implemented in Table 6 are shown below. It should be noted that not all of the functionality for each event is displayed.

```

event RemoveData extends RemoveData
  any id_no
  where
    @grd6 (SIU_time - data_timestamp(id_no ↦ SIU_id)) ≥ DATA_RETAIN_TIME
end

```

A guard in refinement of the *RemoveData* event; *data\_timestamp* is a function mapping each data item (identified by an id number and the SIU) to the time it was created. *SIU\_time* is the current time in the system. The guard asserts that the data cannot be removed until the time it has been stored is at least equal to a minimum value.

```

invariants
@inv2  $\forall m \cdot m \in \text{SIU\_set} \Rightarrow \text{memory\_occupied}(m) \leq \text{TOTAL\_MEMORY}$ 

```

An invariant which specifies that for each of the connected SIUs within *SIU\_set*, the memory occupied should never exceed the limit. The limit is represented by the constant *TOTAL\_MEMORY*.

```

invariants
@inv1  $\text{pending\_messages\_out} \subseteq \text{SIU\_set} \times (\mathbb{0} \dots \text{SIU\_time}) \times \text{SIU\_DATA\_TYPES} \times \mathbb{N}$ 

events
event SIU_SendData extends SIU_SendData
  any time_issued
  where
    @grd_SIU_3  $\text{time\_issued} = \min(\{x \mid \exists m, y, z \cdot (m \mapsto x \mapsto y \mapsto z) \in \text{pending\_messages\_out}\})$ 
    @grd_SIU_4  $(\text{SIU\_id} \mapsto \text{time\_issued} \mapsto \text{data\_type} \mapsto \text{data\_value}) \in \text{pending\_messages\_out}$ 
  end

event StoreData extends StoreData
  then
    @act7  $\text{pending\_messages\_out} = \text{pending\_messages\_out} \cup \{m \cdot m = \text{SIU\_id} \wedge$ 
       $\text{value} > \text{alert\_threshold}(\text{SIU\_id}) \wedge$ 
       $\text{alert\_enabled}(\text{SIU\_id}) = \text{TRUE}$ 
       $\mid m \mapsto \text{SIU\_time} \mapsto \text{ALARM} \mapsto \text{value}\}$ 
  end

```

*pending\_messages\_out* is a set representing the list of messages due to be sent to the data centre. *inv1* specifies that each message has to be of the form [*sending SIU*, *time sent*, *data type*, *data value*].

*SIU\_SendData* refines the original event in the top-level model (see Section 5.1) by insuring that the *SIU\_id*, *data\_type* and *data\_value* variables, all of which are defined in the top-level model, are associated with a message in the pending list (*grd\_SIU\_4*). *grd\_SIU\_3* ensures that the next message sent is the one which has been pending the longest.

The action in *StoreData* will add a message to the pending list if an alert is raised on the data (represented by *value*) that is in the process of being stored, providing that the alert is enabled (dictated by *alert\_enabled*).

The requirements in the documentation for the SIU are traced directly to the model using the ProR tool; this is demonstrated in Figure 9, although the actual requirements have been blurred as they contain proprietary information. The *Link* column on the right hand side allows for each requirement to be linked to a number of Event-B components, such as invariants and events, so that directly traceability to the elements in the model can be achieved. Some warning symbols can also be seen in the *Source Changed* and *Target Changed* columns. In the case of the *Source Changed* column these are shown when the requirement description changes. For *Target Changed*, these are shown when the elements in the model - to which the requirements are linked - are modified.

ID	Description	Source Changed	Target Changed	Link
1	SIU_1			
1.1	5GCS-LVM-14 The SIU shall provide a minimum of 100000 of non-volatile data storage.			0 ▷ 0 ▷ 1
	▷			inv2 (SIU_1)
1.2	5GCS-LVM-15 Up to 4% of the non-volatile memory shall be allocated to storage of measurement data and data reports.			0 ▷ 0 ▷ 1
	▷			inv2 (SIU_1)
2	SIU_2			
2.1	5GCS-LVM-16 The reporting period shall be able to be modified at any time while the system is in operation.			0 ▷ 0 ▷ 1
	▷			ReadPortData (SIU_2)
2.2	5GCS-LVM-17 It shall be possible to set a separate reporting period for each substation controller.			0 ▷ 0 ▷ 1
	▷			inv2 (SIU_2)
2.3	5GCS-LVM-18 The measurement rate shall be able to be modified at any time while the system is in operation.			0 ▷ 0 ▷ 1
	▷			ReadPortData (SIU_2)
2.4	5GCS-LVM-19 Each substation controller may have its own database.			0 ▷ 0 ▷ 1
	▷			inv1 (SIU_2)
3	SIU_3			
4	SIU_4			
4.1	5GCS-LVM-20 Each SIU shall maintain a lock for the purposes of data stripping data.			0 ▷ 0 ▷ 3
	▷			inv3 (SIU_4)
	▷			inv4 (SIU_4)
	▷			SIU_time (SIU_0)
4.2	5GCS-LVM-21 The reports shall be timestamped.			0 ▷ 0 ▷ 1
	▷			inv4 (SIU_4)
4.3	5GCS-LVM-22 Measurement data shall be retained in non-volatile memory for at least 1 day.			0 ▷ 0 ▷ 2
	▷			RemoveData (SIU_4)
	▷			asm2 (C4)

Figure 9 : Traceability of the SIU requirements in ProR

### 5.3.2 Initial Results from the Formal Modelling

From modelling the requirements for the SIU, it immediately becomes apparent that there are several missing requirements. This is evident due to the fact that certain invariants could not be discharged without adding extra elements to the Event-B model:

- There is no requirement stating that data has to be time stamped; only reports. However without this timestamp there is no way to discharge one of the proof obligations. This proof obligation represents a requirement which indicates that the measurement data should be retained in the memory for a minimum period of time.
- The proof obligation representing another requirement - which indicates that the data centre should be able to request reports from the SIU up to a specific time in the past - could not be discharged. Although there are requirements stating the length of time that data and periodic reports should be stored in the memory, there is no equivalent for alert reports. This needs to be specified to ensure that the requirement is met, assuming that the “reports” mentioned in the requirement refers to both periodic and alert reports.

Once the proof obligations for the model were successfully discharged, further issues were found through simulation of the model with ProB. The problems encountered included:

- Deadlock was found in certain scenarios, due to conflicting requirements. This occurs when the memory is full but no data or reports can be removed, as the time they have been stored is less than a minimum stated in the requirements. The requirements document for the SIU also specifies a set of reporting and measurement rates which can be selected. By running a simulation with each

combination of rates it is possible to show which combinations produce deadlock and which do not. A screenshot of an example with deadlock is shown in Figure 10. As can be seen none of the events are enabled on the left hand side. *tick* cannot occur as data needs to be stored for the current time (*data\_pending* is true for SIU 1). However, *memory\_occupied* for SIU 1 is too large to save this data, and no data or reports can be removed as all values for SIU 1 in *data\_timestamp* and *report\_timestamp* are less than *DATA\_RETAIN\_TIME* and *REPORT\_RETAIN\_TIME* respectively.

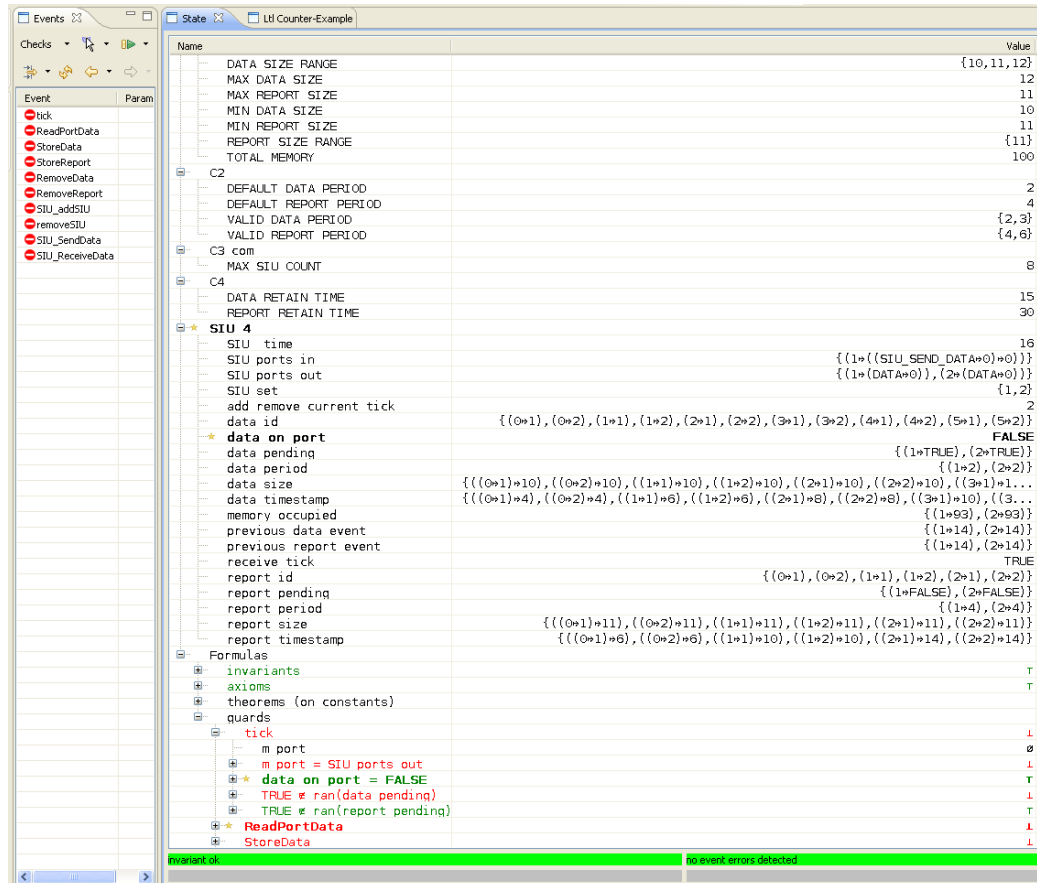


Figure 10 : Example of deadlock found through ProB simulation

- The requirements state that it should be possible to change the reporting or measurement period at any time. An assumption had to be made with regards to this for the model to behave correctly during simulation. A rule has to be set in place as to whether the new interval is either: initiated from the time the message is received, or initiated at the next periodic interval. In the first instance it cannot be reliably determined if the SIU is reporting regularly or responding to commands correctly. This is because the exact time at which the command to change the period is received by the SIU is not known to the data centre, due to the varying time taken for a packet to travel through the network. The data centre needs enough knowledge of the current behaviour of the SIU to determine if the future behaviour is correct, and in this case it does not. This indicates a missing requirement specifying the time at which the new reporting or measurement period is initiated after receipt of the command. If the new period is instead initiated at the next periodic interval, the data centre can predict the times at which it should

receive future messages from the SIU, and also the interval within which the message was received by the SIU.

### 5.3.3 Compatibility with Previously Stated Assumptions

A number of assumed requirements regarding the SIU were made in [AD-1]. Since this point, it has been possible to review the requirements document. It has become apparent that some of the assumptions are no longer valid, and that others need adjusting. A summary of these findings is presented in Table 7 below, where green represents a requirement which is still valid and red a requirement which has to be changed or discarded:

Original Requirement		Current Status
ID	Description	
SGCS-LVM-FUN-001	<i>It shall be possible for new sensors to be added to the system after initial setup without disrupting the normal operation of the system.</i>	Not relevant to the model as it is indicated in the requirements that sensors can only be added during the setup state.
SGCS-LVM-FUN-002	<i>It shall be possible for existing sensors to be removed or replaced without disrupting the normal operation of the system.</i>	Not possible for the reasoning above.
SGCS-LVM-NON-003	<i>There shall be a secure identification system for adding new sensors such that only sensors approved by the supplier can be added.</i>	Sensor addition has not yet been modelled, although there is no mention of this in the requirements document.
SGCS-LVM-NON-004	<i>There shall be a maximum number of sensors that system can use; after this has been reached no more sensors should be added.</i>	Still valid.
SGCS-LVM-NON-005	<i>The limits should not be changed except by an outside control system.</i>	Still valid as the limits associated with alerts can only be changed by the data centre.
SGCS-LVM-NON-006	<i>The warning shall be issued immediately after the readings which exceed the limit are received by the SIU – this shall still be true when multiple limits are exceeded at the same time.</i>	“Immediately” needs to be changed so instead the warning should be issued within a certain period; one of the requirements specifies this time period. It is also the case that the process of issuing the alert is more complex, as it has to be sent to the data centre and this is not necessarily reliable.
SGCS-LVM-NON-007	<i>The warning shall be retained (until input by an outside control system or operator is provided) if the system subsequently drops back below the limit. However in this interim period – if the type of warning is not detrimental to the health of the device - the system should not prevented from functioning as normal</i>	Still valid.

<b>SGCS-LVM-FUN-008</b>	<i>It shall be possible for the reporting interval and reporting mode to be changed at any time.</i>	Still valid, although require additional requirement as described in Section 5.3.2.
<b>SGCS-LVM-NON-009</b>	<i>Changing the reporting mode or reporting interval should not cause any loss of records.</i>	Still valid.
<b>SGCS-LVM-NON-010</b>	<i>The reporting interval and reporting mode should only be changed by an outside control system and not internally.</i>	Still valid, as these can only be changed due to commands from the data centre.
<b>SGCS-LVM-NON-011</b>	<i>There shall be limited memory in the system.</i>	Still valid, and specified by requirements in the document.
<b>SGCS-LVM-NON-012</b>	<i>If the memory is full then the most historical record shall be deleted on arrival of a new record.</i>	No memory management system is specified in the requirements document - so it is not clear if this is the process used - although there have been problems shown with deadlock of the model (see Section 5.3.2)
<b>SGCS-LVM-FUN-013</b>	<i>Only if the memory is full or there is direct intervention by an outside control system should records be removed.</i>	Not necessarily required, as the requirements state a minimum period of time each report and data has to be kept in the memory. After this there will be no conflict if it is removed.

Table 7 : Validity of previously assumed requirements for the SIU

## 5.4 Network

The model for the network differs to that of the SIU and data centre as there are no strict requirements regarding how it has to behave. Instead its role in the verification is more apparent when it is composed with the models for the data centre and SIU. By introducing events that represent, for instance, a connection failure or lost message, it can be verified that the data centre and SIU react to these scenarios in a suitable fashion. Even without introducing new events, the automatically generated network model from the decomposition provides the binding needed for the data centre and monitoring/control units to communicate, and thus allows for the verification of the established properties of the data centre and monitoring/control units when this communication is taken into account.

Furthermore, it is from the results of the composition that the requirements relevant to the network can be inferred, where before they were ambiguous. This will allow for a specification of the network – which has been shown to be required for the monitoring/control units to function correctly - to be drawn up.

### 5.4.1 Re-composing the Developed Models with the Network

The models are composed together using the Parallel Composition tool; a sample of the composition file used to create the composed machine is shown in Figure 11.



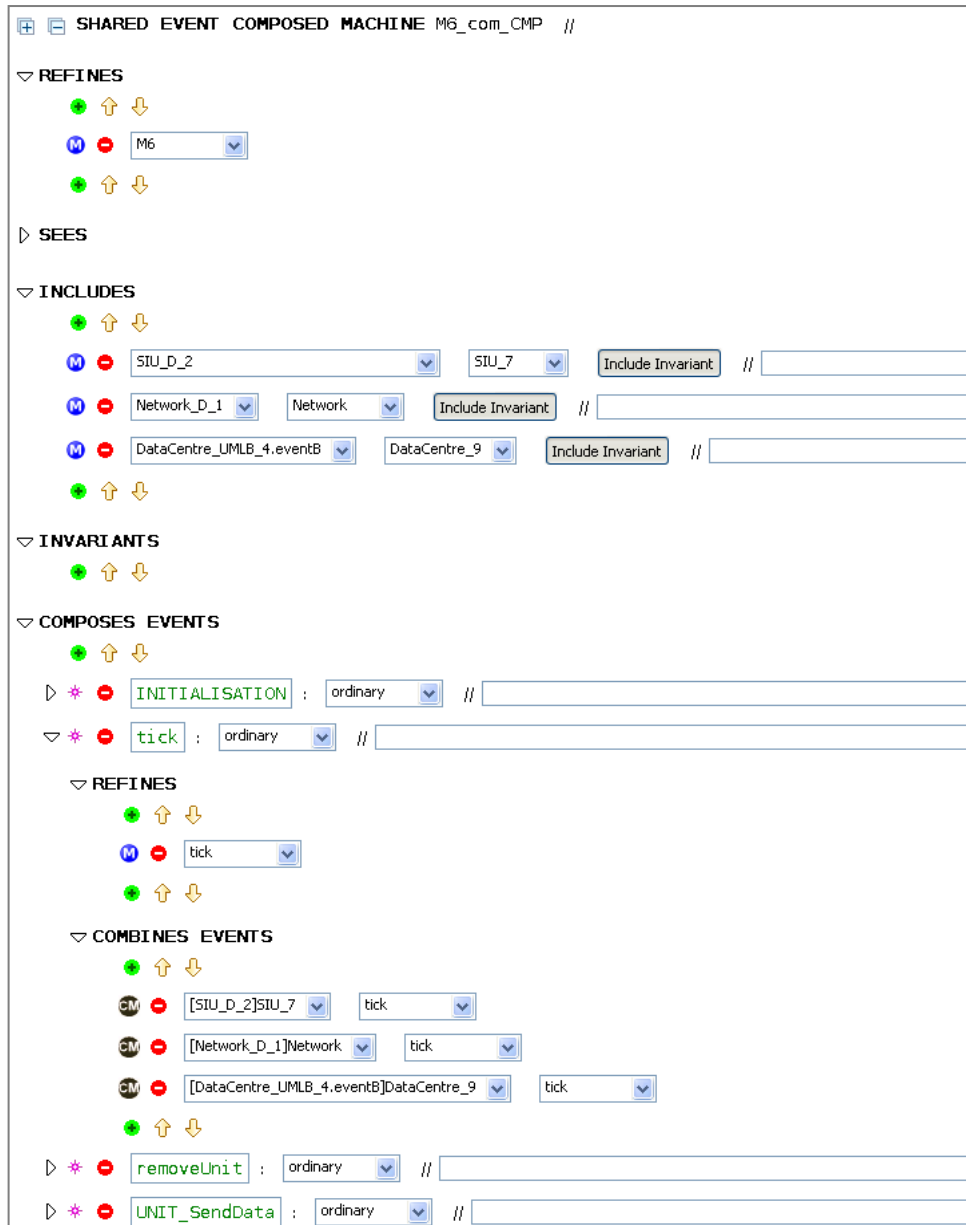


Figure 11 : Part of the Composition Machine file used to create the composed machine

As can be seen in Figure 11, the composed machine is set to refine M6, which is the source of the decomposition in the top-level model (see Section 5.1.1). If the proof obligations created with the composed file can be discharged this proves that in combination, the three decomposed models refine the original machine.

The models to be included in the composition –those listed under the *INCLUDES* heading in Figure 11 – include the furthest refined machine from the SIU and data centre (see Figure 4 and Table 6 for reference). Although, composed machines can also be created using machines from earlier refinement levels if selected functionality wants to be investigated, or in order to check the validity of the models at each stage before continuing. Checking the validity of the models not only involves showing that the decomposed models refine the original machine – as mentioned above – but also involves simulating the composed model with ProB. The values received by the SIU are sent by the data centre and vice versa. However, as the data centre and SIU are modelled independently, the properties are

verified without any interaction between the two. The composed model introduces this interaction through the network element, and running ProB on the composed model can provide a check of this interaction. For instance, there may be deadlock when the models for the SIU and data centre are interacting, even though there is no deadlock present in the models when considered separately.

The detail of one of the combined events, *tick*, is also shown. In this case the equivalent *tick* events from all three machines are included in the composition of the event. However, this will not always be the case due to the distribution of events as indicated in Table 5; for instance, the *UNIT\_SendData* event will only combine the equivalent events from the SIU and network machines.

## 6. Conclusions and Next Steps

The methodology presented in this document can be applied to any distributed network system with a similar structure, i.e. data centre, monitoring and control system, and network. Having said this, the top-level model and decomposition process still need to be made more generic in order to encompass different network structures and hierarchies. For instance, there can be a variable number of tiers within the network, and this needs to be introduced into the model as a variable rather than a constant. Further to this, it could be the case that each unit can be assigned to different levels in the hierarchy, even if they are of the same type. The process also needs to be able to cater for more than one data centre or control system, and the option for units to send and receive messages with other units at selectable levels or at the same level. This will form part of the next stage of work, with the focus on the importance of retaining the reusability that has been expressed in this document.

The proof of concept in this report has utilised most of the main tools available during the ADVANCE workflow. It has become apparent that the multiple plug-ins for Rodin play an important role in the development of the model and eventual verification. The often more diagrammatic explanation - provided by tools such as the UML-B and Statemachine plug-ins - is clearer to those without much experience with formal languages. Also of particular value is the simulation provided by ProB. During the initial stages of the model this allows for a clearer idea of the behaviour that the model represents. It is often the case that the behaviour does not exactly match the implementation that was sought after or expected, and the model has to be modified. This modification is simple at the initial stages, yet if it is left to the later stages the changes potentially have a much larger impact, both on the foundation of the model and the amount of effort required to implement the changes. As demonstrated in Section 5.3.2, even once all of the proof obligations for the model have been discharged, problems are still found during simulation.

Both the decomposition and ProR tools provide an aid for team working where it would be difficult to perform otherwise. Once the model is decomposed each part can be worked on separately and composed at different stages to check the validity of the new functionality. The warnings in ProR - when either a requirement or the expression linked to a requirement is modified - make it possible to identify when the section of the model that an individual is working on will affect either the implementation that has already been completed, or that which is being developed in parallel.

The final stage in the workflow is code generation from the Event-B models. Although the results of which have not been demonstrated in the document, both the models for the data centre and SIU are close to the stage where enough lower level detail is implemented for code generation to become a viable option.

### 6.1 Update on Suitability for Influencing Tool Development

In Section 5 of [AD-1], the tools that the case study should provide feedback for, and validate any development of, were listed. An update on the current progress of this is shown below:

- **Decomposition:** Both the decomposition and composition tools are an integral part of the proof of concept as they provide the framework for the process. It has been possible to investigate the efficiency of different decomposition strategies, and the difficulties of making changes at a higher level once work has begun on the decomposed models has become apparent. The difficulties encountered are similar to those present when using refinement without decomposition, where any changes to the higher level model will impact lower levels, which then have to be adjusted.

As the decomposition involves several chains of refinement, it is even more vital that the suitability of the decomposition is ensured before the majority of the work starts on the decomposed parts.

- **Multi-simulation framework:** Only discrete simulation has been performed thus far, although the input to the monitoring units could be considered a continuous function once this is factored into the model.
- **Linking safety analysis with formal modelling:** No link with safety analysis has been performed yet, although it is planned to be included in the next phase of work.
- **Linking requirements and traceability:** The ProR tool was used throughout the development of the SIU model to ensure traceability of the requirements. Feedback was provided for the tool. In particular, it was felt that the ability to automatically generate a report showing coverage would be beneficial in an industrial scope, in order to convey the progress and coverage to a customer. Also, the inclusion of links from ProR to the relevant element in the Event-B model would improve usability.
- **Code Generation:** As mentioned, although code generation has not been performed yet, the models are close to a point where it will be feasible to do so in the next phase of work. The decomposition allows for a better application of code generation, as code can be generated for the data centre and SIU separately. This also means that code generation does not have to be applied to the network model, where it would not be as applicable or beneficial.
- **Automated proof and model-checking:** ProB was used throughout the modelling process, both during the initial modelling stages to check the implementation, and at the final stages to provide model checking.
- **Language extension / creating re-usable patterns:** The aim of the decomposition process was to create a top-level model which is reusable. In the next phase of work it will be necessary to test that this is the case by modelling more than one monitoring/control unit or control system.
- **Scalability:** The model is scalable in terms of the number of connected units at any one level of the system, as explained in Section 5.1.3. The next step is to extend the combined system so that it is also scalable in terms of the number of levels within the network.

## 6.2 Update on Expected Outcomes and Success Criteria

In Section 6 of [AD-1] the success criteria of the case study were proposed. The adherence to these criteria is examined below:

- *The study can demonstrate that formal methods are beneficial when applied to smart grid solutions; i.e. the concept of a smart grid can be modelled and its properties verified.*

Although the process described in the proof of concept has not yet been applied specifically in the context of smart grids, there are still enough similarities that mean the techniques used will also prove beneficial when applied in this particular domain.

- *The study demonstrates that the complexity of the system can be mastered using refinement.*

The proof of concept has demonstrated that the complexity of each model has been reduced using refinement. Decomposition also plays a role in decreasing the complexity of the combined system.

- *The process demonstrates a reduced time-to-market by providing better levels of assurance than traditional engineering methods.*

This is an area which will be better explored as work on the individual components of the system progresses.

- *The study provides understanding on how the ADVANCE tools can be applied in commercial environments and in doing so provide a more efficient approach than traditional engineering methods.*

By applying the ADVANCE tools to the commercial SIU project and focusing on a domain which will provide multiple commercial solutions such as the SIU, this helps provide the understanding required to make a suitable assessment on the approach. Once different products are examined as the case study progresses this assessment will be strengthened.

- *The study uses real-world smart grid data to assist the modelling process.*

Real data has not been used yet as the models need to reach a higher level of maturity before this will be an advantageous step.

- *The case study demonstrates that the cost of development can be reduced by generating metrics.*

This will be demonstrated once the case study reaches the latter stages.

- *The study provides understanding on how safety analysis plays a role in and can be linked to formal methods.*

This has not yet been included but it is planned to be investigated in the next phase of work.

- *It is demonstrated that properties of the system can be verified through automated model checking and simulation.*

The use of ProB throughout the process – and the results found from this – satisfies this criterion in terms of the current progress.

- *The end product demonstrates that code generation is possible.*

As has been mentioned, although this has not been demonstrated, the model is nearing the stage where it will be possible.

- *The study provides examples of successfully utilising the state machines, UML-B and decomposition tools.*

Each of these tools has been used - with the result of clarifying the model or simplifying the modelling process - during the proof of concept.

## Appendix A. UML-B Key

The following tables should be referred to when studying the UML-B machine and context diagrams:

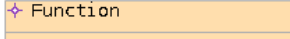
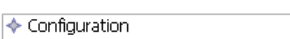
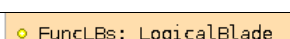
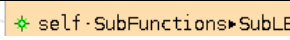
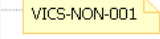

 Function	Class Type
 Configuration	Extended Class Type (refined class type from a more abstract level)
 FuncLBs: LogicalBlade	Class Type Attribute (static mapping)
 self·SubFunctions▶SubLE	Class Axiom (static requirement/property to be verified)
 VICS-NON-001	Comment (ignored during generation of Event-B model)
 Constant	Constant (static variable)

Table 8 : UML-B Context Types

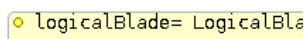
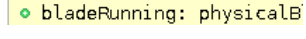
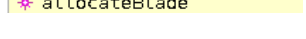
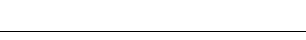
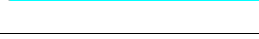
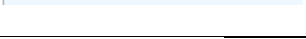

 logicalBlade= LogicalBlade	Class (has a Class Type defined in the Context)
 bladeRunning: physicalB	Class Attribute
 allocateBlade	Class Event
 true	Class Invariant (dynamic requirement/property to be verified)
 Machine Event	System Event
 Machine Variable	System Variable
 VICS-NON-001	Comment (ignored during generation of Event-B model)

Table 9 : UML-B Machine Types