



University
of Glasgow

MacIsaac, Liam J. (2013) Modelling smart domestic energy systems. PhD thesis

<http://theses.gla.ac.uk/4214/>

Copyright and moral rights for this thesis are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This thesis cannot be reproduced or quoted extensively from without first obtaining permission in writing from the Author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the Author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given.

Modelling Smart Domestic Energy Systems

Liam J. Maclsaac

Submitted to the University of Glasgow, School of Engineering, in fulfillment of the requirements for the degree of Doctor of Philosophy.

April 2013

©2013 Liam Maclsaac

Abstract

The increasing price of fossil fuels, coupled with the increased worldwide focus on their contribution to climate change has driven the need to develop cleaner forms of energy generation. The transition to cleaner energy sources has seen a much higher penetration of renewable sources of electricity on the grid than ever before. Among these renewable generation sources are wind and solar power which provide intermittent and often unpredictable energy generation throughout the day depending on weather conditions. The connection of such renewable sources poses problems for electricity network operators whose legacy systems have been designed to use traditional generation sources where supply can be increased as required to meet demand. Among the solutions proposed to address this issue with intermittency in generation are storage systems and automation systems which aim to reduce demand in order to match the available renewable generation. Such a transition would introduce a requirement for more advanced technology within homes to provide network operators with greater control over domestic loads.

Another aspect to the transition towards a low-carbon society is the change that will be required to domestic heating systems. Current domestic heating systems largely rely on Natural Gas as their fuel source. In order to meet carbon reduction targets, changes will need to be made to domestic buildings including insulation and other energy efficiency measures. It is also possible that present systems will begin to be replaced by new heating technologies such as ground and air source heat pumps.

Due to the effect that such technological transitions will have on domestic end-users, it is important that these new technologies are designed with end-users in mind. It is therefore necessary that software tools are available to model and simulate these changes at the domestic level to guide the design of new systems.

This thesis provides a summary of some of the existing building energy analysis tools that are available and shows that there is currently a shortcoming in the capabilities of existing tools

when modelling future domestic smart grid technologies. Tools for developing these technologies must include a combination of building thermal characteristics, electrical energy generation and consumption, software control and communications.

A new software package was developed which allows for the modelling of small smart grid systems, with a particular focus on domestic systems including electricity, heat transfer, software automation and control and communications. In addition to the modelling of electrical power flow and heat transfer that is available in existing building energy simulation packages, the package provides the novel features of allowing the simulation of data communication and human interaction with appliances. The package also provides a flexible framework that allows system components to be developed in full object-orientated programming languages at run time, rather than having to use additional third-party development environments.

As well as describing the background to the work and the design of the new software, this thesis describes validation studies that were carried out to verify the accuracy of the results produced by the package. A simulation-based case study was also carried out to demonstrate the features offered by the new platform in which a smart domestic energy control system including photovoltaic generation, hot water storage and battery storage was developed. During the development of this system, new algorithms for obtaining the operating point of solar panels and photovoltaic maximum power point tracking were developed.

Acknowledgements

I would like to take this opportunity to thank everybody who has helped me in some way or another during the preparation of this thesis. My supervisor, Professor Andrew Knox for his help and guidance throughout the project and for the time spent reviewing the material in this thesis. Also, Graham Morton, Jeremiah Anonymous Matthey and Ed Pratt who made sure that our office was never dull! I would also like to thank Dr James Buckle for his help and advice. I would also like to thank Calum Cossar, Peter Miller and Ian Young of the power group for their technical help throughout the project.

Finally, and most importantly, I would especially like to thank all of my friends and family who have been very supportive throughout the duration of my PhD and especially Rachael who has provided plenty of moral support and has put up with my constant stressing about this Thesis for the last year!

Also to everyone else who has helped me in some way during the preparation of this thesis and has not been mentioned here, thank you.

Publications

L. Maclsaac, A. Knox, "Improved Maximum Power Point Tracking Algorithm for Photovoltaic Systems", International Conference on Renewable Energies and Power Quality: Granada, Spain, March 2010

L. Maclsaac, A. Knox, "Improved Maximum Power Point Tracking Algorithm for Photovoltaic Systems", Renewable Energy and Power Quality Journal, Vol. 1, No. 8, April 2010

L. Maclsaac, A. Knox, "Domestic End-Use Modelling of Smart Grid Technologies", IEEE Innovative Smart Grid Technologies Europe Conference: Manchester, UK, December 2011

L. Maclsaac, A. Knox, "New Algorithm for Obtaining the Operating Point of Photovoltaic Systems", Proceedings of the Institution of Mechanical Engineers – Part A – Journal of Power and Energy, vol. 226, no. 6, 2012

Contents

Abstract	i
Acknowledgements	iii
Publications	iv
Contents	v
List of Figures	xiii
List of Tables	xix
List of Abbreviations	xxi
Chapter 1 Introduction	1
1.1. Aims and Objectives.....	2
1.2. Original Contributions of this Research	3
1.2.1. Flexible Software Platform	3
1.2.2. Photovoltaic Maximum Power Point Tracking	4
1.3. Outline	4
Chapter 2 Background	6
2.1. Smart Grids	6
2.1.1. 20 th Century Energy Networks	6
2.1.2. Low-Carbon Electricity.....	9
2.1.3. Low-Carbon Heating	10
2.1.4. Low-Carbon Transport.....	13
2.1.5. Advanced Metering Infrastructure.....	14
2.1.6. Smart Grids Summary.....	17
2.2. Smart Grid Software Modelling	18

2.2.1. EnergyPlus.....	20
2.2.2. Other Building Energy Analysis Software	23
2.2.3. Summary of Smart Grid Software Modelling.....	26
2.3. Summary of Project Background	30
Chapter 3 Software Design.....	31
3.1. Choice of Application Development Platform	31
3.2. Representation of Models	32
3.2.1. Techniques in Existing Packages	32
3.2.2. Design of New Technique	33
3.3. Physical Domain Representations.....	35
3.3.1. Electrical.....	37
3.3.2. Thermal	40
3.3.3. Data Communication	42
3.3.4. Asynchronous Communication	45
3.3.5. Others	46
3.4. Defining & Simulating Component Behaviour.....	46
3.4.1. Component Appearance and Configuration.....	46
3.4.2. Simulation Technique	49
3.4.3. Implementing Component Behaviour	51
3.5. Simulation Results	59
3.5.1. Recording Results	59
3.5.2. Mathematical Operations on Results	66
3.5.3. Analysing Results.....	67
3.6. Persistence of Models	68

3.7. Application Design Summary	76
Chapter 4 Implementation of Models	78
4.1. Electrical Components.....	78
4.1.1. Ideal Sources	78
4.1.2. Solar Panel	79
4.1.2.1. Operating Point Detection Algorithms	82
4.1.2.2. Comparison of Operating Point Detection Algorithms	86
4.1.3. Battery Storage	90
4.1.4. Connection and Control.....	92
4.1.4.1. Wire.....	94
4.1.4.2. Switch.....	95
4.1.4.3. Digital Switch	95
4.1.4.4. Relay.....	95
4.1.4.5. Smart Meter	96
4.1.5. Loads	100
4.1.5.1. Static Loads	101
4.1.5.2. Multi-Mode Loads	102
4.1.5.3. Time-Varying Loads	103
4.1.5.4. Dynamic Loads.....	104
4.2. Communication Components	107
4.2.1. Scheduled Data Generator	107
4.2.2. Repeater.....	107
4.3. Building Elements.....	108
4.3.1. Materials Database.....	108

4.3.2. Surface	109
4.3.3. Door	112
4.3.4. Window	113
4.3.4.1. ISO 10077-1 Window Geometry	113
4.3.4.2. Software Modelling of Window Geometry	114
4.3.4.3. Translating Hierarchical Model to ISO Standard Model	117
4.3.4.4. Thermal Transmittance Calculation	118
4.3.4.5. Window Model Definition	119
4.3.5. Room	119
4.3.6. Static Room	121
4.4. Heating and Ventilation	122
4.4.1. Natural Ventilation Component	122
4.4.2. Radiator	123
4.4.3. Electric Heater	125
4.5. Weather	126
4.5.1. Ambient Temperature Pattern	126
4.5.2. Random Ambient Temperature	126
4.5.3. Solar Irradiation Pattern	127
4.6. Other Components	127
4.6.1. Scheduled Asynchronous Messaging	127
4.6.2. List and Table Parameter Watch	128
4.7. Summary	129
Chapter 5 Testing & Validation	130
5.1. Unit Testing	130

5.2. Automated Functional Test Program	131
5.3. Graphical User Interface Testing.....	133
5.4. Experimental Validation Studies – Thermal Models	134
5.4.1. Experimental Methodology	134
5.4.2. Test Room Experiment – Heat Loss Response	136
5.4.3. Test Room Experiment – Heated Room Response	143
5.5. Experimental Validation Study – Electrical Circuit.....	150
5.6. Summary of Testing and Validation	153
Chapter 6 Case Study	156
6.1. Home Energy System Overview.....	156
6.2. Maximum Power Point Tracking.....	158
6.2.1. Background	158
6.2.2. Newly Developed Algorithm	161
6.2.2.1. Description of the New Algorithm.....	161
6.2.2.2. Algorithm Comparison.....	164
6.2.3. MPPT Controller	166
6.2.4. Battery Controller.....	167
6.3. Water Heater	168
6.4. Domestic Load Model.....	169
6.5. Metering System and Control Algorithm	170
6.6. Simulation Results.....	171
6.6.1. Baseline Case.....	171
6.6.2. Load Balance	172
6.7. Summary of Results.....	176

Chapter 7 Conclusions	178
7.1. Chapter Summary	178
7.2. Novel Contributions of the Research	182
7.3. Future Work and Improvements	183
Appendix A Component Scripting API	187
Appendix B Window Model Class Diagram	189
Appendix C Case Study Simulation Models	190
C.1. Case Study 1 – Home 1, Dining Room	190
C.2. Case Study 2 – Home 2, Lounge.....	191
C.3. Case Study 3 – Electrical Validation Model	191
Appendix D Power Analyser Unit	192
D.1. System Overview	192
D.2. Voltage and Current Sensors	193
D.3. Signal Conditioning Circuit	194
D.4. Signal Processing Circuit.....	194
D.5. PC Software	194
D.6. Power Analyser Circuit Diagrams	196
D.6.1. Mains Wiring.....	196
D.6.2. Signal Conditioning Circuit	197
D.6.3. Signal Processing Circuit	198
Appendix E MPPT Algorithm Comparison Method	199
E.1. Control Software	199
E.2. Simulated Solar Panel.....	201
E.3. In-Circuit Solar Panel	201

E.4. Result Recording	202
E.5. Maximum Power Point Tracking Algorithms	202
E.6. Control of Comparison Scenarios.....	202
Appendix F Software Implementation	204
F.1. Source Code Compilation	204
F.1.1. C# and VB.NET	204
F.1.1.1. Pre-Processing Code	205
F.1.1.2. Code Compilation	207
F.1.1.3. Loading Compiled Code	208
F.1.2. Mathematical Mark-up	209
F.1.2.1. Tokenising.....	210
F.1.2.2. Shunting Yard Conversion	211
F.1.2.3. Reverse Polish Notation Solution	211
F.2. Simulation Engine.....	213
F.3. Result Viewers.....	215
F.3.1. Table	215
F.3.2. Line Graph	216
F.3.3. Pie Chart.....	216
F.4. Parameter Value Editors.....	217
F.5. Graphical Model Editor.....	224
F.5.1. Overview	224
F.5.2. General Design Considerations for Editor	226
F.5.3. Editor Design	227
F.5.4. User Input Handling.....	229

F.5.5. Drawing 230

F.6. Graphical Component Editor 231

F.7. Implementation Summary 236

References 237

List of Figures

Figure 2-1: Diagram illustrating the current architecture of the UK's domestic energy system.	7
Figure 2-2: UK heating fuel mix in 2007 [4]......	8
Figure 2-3: UK overall fuel mix for all energy sectors in 2009 [5].	8
Figure 2-4: Share of citizens served by district heating schemes in selected European countries.....	12
Figure 2-5: Possible architecture of the future Smart grid, based on one proposed by the UK electricity networks strategy group. [47].....	18
Figure 3-1: Internal representation of a system model within the software package.	36
Figure 3-2: Graphical representation of a sample system within the simulation package.	36
Figure 3-3: Software representation of the graphical model shown in Figure 3-2.....	36
Figure 3-4: Illustration of the way in which (a) AC and (b) DC electrical pins are modelled on components within the simulation package.....	38
Figure 3-5: Equivalent schematic of an electrical node component.	40
Figure 3-6: Illustration of a thermal connection between two components. Heat transfer pins are expressed in terms of temperature T ($^{\circ}\text{C}$) and thermal resistance θ ($^{\circ}\text{C}/\text{W}$). The resulting heat transfer from left to right, Q (W), is illustrated.....	42
Figure 3-7: Illustration of some of the possible communications channels involved in a smart energy system. 1) A control and status reporting network between the smart meter, in-home display and smart appliances. 2) PC connections to the smart meter or smart appliances. 3) Uplink to the utility provider. 4) User connection to the utility provider for control and monitoring of account.	43
Figure 3-8: Activity diagram illustrating the method used within the package to evaluate a model.	51
Figure 3-9: Abstract class which provides the methods that component scripts should implement to define component behaviour.....	53

Figure 3-10: Activity diagram illustrating how the process of evaluating mathematical scripts representing component behaviour maps onto the 4-function interface used by C# or VB.NET scripts. Note that the “End” function is not included because it is not used when evaluating mathematical scripts.	58
Figure 3-11: UML diagram illustrating the design of the watch system used to collect results from the simulation.	61
Figure 3-12: Illustration of an electrical connection between two components showing complex voltages V_1 & V_2 , complex impedances Z_1 and Z_2 and complex current I	62
Figure 3-13: Illustration of a heat transfer connection between two components showing temperatures T_1 & T_2 ($^{\circ}\text{C}$), thermal resistances θ_1 and θ_2 ($^{\circ}\text{C}/\text{W}$) and heat flow Q (W).	64
Figure 3-14: Sequence diagram illustrating the processing of a communications message. The optional block at the end of the sequence of events indicates the way in which a message is intercepted for forwarding to an associated Data Watch object.	66
Figure 3-15: UML diagram illustrating mathematical operator system within the package.	67
Figure 3-16: XML schema for a component library file.	72
Figure 3-17: XML schema for a system model file.	73
Figure 3-18: XML schema for component behaviour elements.	74
Figure 3-19: XML schema for component instance elements.	75
Figure 3-20: Diagram illustrating the architecture of the new simulation package.	77
Figure 4-1: Illustration of component symbols and source code for DC and AC ideal voltage source components. Source code for these components is written in the mathematical mark-up language.	79
Figure 4-2: Two-diode solar cell model.	79
Figure 4-3: Single-diode solar cell model.	79
Figure 4-4: Illustration of the I/V characteristic of a BP SX-80 solar panel showing the equivalent load required to obtain each operating voltage.	82

Figure 4-5: Illustration of the new algorithm developed to determine the operating point of a solar panel when the load resistance is known. The output current $I(V)$ is obtained using (23) and $R(V) = V/I(V)$	85
Figure 4-6: Illustration showing the range over which the linear search algorithm was able to obtain I/V operating points for the BP SX-80 solar panel.....	87
Figure 4-7: Illustration showing the range over which the Newton’s method algorithm was able to obtain I/V operating points for the BP SX-80 solar panel.....	88
Figure 4-8: Illustration showing the range over which the newly developed algorithm was able to obtain I/V operating points for the BP SX-80 solar panel.....	88
Figure 4-9: Comparison of the time taken to reach a solution for each algorithm using an irradiation level of 1000W/m ² . The values for 500W/m ² are excluded from this illustration because they are of similar magnitude.....	89
Figure 4-10: Schematic of a connection or control component with impedance $Z_{\text{CONNECTION}}$ inserted in series between two components.....	92
Figure 4-11: Schematic illustrating the way in which the simulation package requires a series component to be modelled.....	93
Figure 4-12: Component which models the series resistance introduced by electrical wiring.	94
Figure 4-13: Smart electricity meter component.....	96
Figure 4-14: Model used for an electrical load component.	101
Figure 4-15: Schematic diagram illustrating the model used for a surface component.	109
Figure 4-16: Example of initial surface temperature gradient calculation.	111
Figure 4-17: Dynamic model of a material within a surface.	111
Figure 4-18: Final surface model.	112
Figure 4-20: Hierarchical model of a window.	115
Figure 4-21: Illustration of how a moveable sash is defined within the window component.	116
Figure 4-22: Custom configuration dialog included within the “Window” component to allow for the definition of window properties.	117

Figure 4-23: Schematic illustrating equivalent thermal resistance model for the window component.	119
Figure 4-24: Electrical analogy for the thermal model of a room.	120
Figure 5-1: Automated test tool test case editor.	132
Figure 5-2: Automated test runner interface.	133
Figure 5-3: 3D Model of dining room used in case study showing neighbouring rooms.	139
Figure 5-4: Results of 24 hour simulation of dining room temperature compared to measured room temperature.	140
Figure 5-5: Percentage error in simulated room temperature during 24-hour study.	141
Figure 5-6: Results of 48 hour simulation of dining room temperature compared to measured temperature.	142
Figure 5-7: Percentage error in simulated temperature during the 48-hour study.	143
Figure 5-8: 3D model that was developed as a reference geometry model for the home used in the second case study.	144
Figure 5-9: Comparison of measured and simulated room temperature in the second case study.	148
Figure 5-10: Percentage error in simulated temperature in the second case study.	148
Figure 5-11: Comparison of physical measurements of domestic appliance power consumption with simulated power consumption.	151
Figure 5-12: Comparison between the simulated and measured values of the total power consumption of all appliances.	152
Figure 6-1: Domestic energy system modelled in this case study. Solid lines indicate electrical connections while dashed lines indicate communication links.	157
Figure 6-2: Illustration of the current-voltage relationship of a solar panel for varying solar irradiation levels. Maximum power points are indicated on each curve.	158
Figure 6-3: Sample illustration showing points collected during different runs of a perturbation and observation algorithm and the maximum power point on each occasion (indicated in bold).	162
Figure 6-4: Activity diagram of the newly-developed learning maximum power point tracking algorithm.	163

Figure 6-5: Power output of a solar panel under experimental conditions when using the Incremental Conductance maximum power point tracking algorithm. The dotted line indicates the approximate maximum power point of the panel under fully-illuminated experimental conditions.....	165
Figure 6-6: Power output of a solar panel under experimental conditions when using the Learning based maximum power point tracking algorithm. The dotted line indicates the approximate maximum power point of the panel under fully-illuminated experimental conditions.....	166
Figure 6-7: Consumption of hot water per hour as recorded in a typical home within the Energy Saving Trust hot water survey.	169
Figure 6-8: Simulated electrical power demand for the home.....	170
Figure 6-9: Load profile of the home over 24 hours with no thermal or electrical storage...	172
Figure 6-10: Load profile of the home over 24 hours with only the thermal storage system enabled.....	173
Figure 6-11: Water heater temperature over 24 hours without control system enabled.	173
Figure 6-12:Water heater temperature over 24 hours with control system enabled.	174
Figure 6-13: Graph illustrating power imported from the grid when only the battery storage system is enabled.	175
Figure 6-14: Graph illustrating power imported from the grid when both the battery and hot water storage systems are used.....	175
Figure F-1: Illustration of the C# code required to implement component behaviour.	207
Figure F-2: Class for evaluating mathematical statements within the simulator.	209
Figure F-3: Tokenising a mathematical expression.	210
Figure F-4: Illustration of tokenised mathematical expression being converted into reverse polish notation.....	211
Figure F-5: Activity diagram illustrating the process used by the simulation engine thread to execute simulations while allowing the user interface to pause or stop execution.....	214
Figure F-6: Table result viewer control.	215
Figure F-7: Illustration of a line graph produced by the .NET 4.0 Chart control.....	216

Figure F-8: Illustration of the pie chart result viewer included in the package.	217
Figure F-9: Configurable parameter list control shown within the component properties viewer.	218
Figure F-10: Integer parameter editing dialog.	219
Figure F-11: Boolean parameter editing dialog.	219
Figure F-12: String parameter editing dialog.	219
Figure F-13: Decimal parameter editing dialog.	220
Figure F-14: Time parameter editing dialog.	220
Figure F-15: List parameter editing dialog.	221
Figure F-16: Table parameter editing dialog in normal operating mode.	222
Figure F-17: Enhanced version of the table editing dialog for use at component design time. Options to edit the table structure are also provided.	222
Figure F-18: Selection list editing dialog.	223
Figure F-19: Enhanced version of the selection list editing dialog used at component design time.	223
Figure F-20: Custom configuration parameter editing dialog.	224
Figure F-21: Annotated diagram illustrating the main features of the graphical model editor.	225
Figure F-22: Model editor class diagram.	228
Figure F-23: Properties page from the component library editor.	232
Figure F-24: Component library editor pin configuration page.	233
Figure F-25: Parameter editing page of the library editor.	234
Figure F-26: Source code editing page of the component library editor.	235

List of Tables

Table 2-1: Comparison of the domestic smart grid modelling capabilities of whole-building energy analysis tools. A solid dot indicates that the package fully supports the specified feature, a hollow dot indicates partial support and no dot indicates no support.	29
Table 3-1: Summary of the key features of the mathematical mark-up language that has been included in the simulation package.	56
Table 3-2: Illustration of the mapping process used in the mathematical mark-up language.	57
Table 3-3: Illustration comparing the implementation of a configurable voltage source in C# to the implementation in the mathematical mark-up language.	57
Table 3-4: Illustration of the memory requirements for recording numeric simulation results at different temporal resolutions. This table makes the assumption that 10 different result values are collected from the model on each iteration and that these results are floating-point values.	59
Table 4-1: Comparison of result errors in operating point detection algorithms.	90
Table 4-2: Parameters used to configure the smart meter component.	99
Table 4-3: Parameters that store metering results within the smart meter component.	100
Table 4-4: Measured operational characteristics of household appliances and calculated impedance value for use in models.	102
Table 4-5: Measured operational characteristics of household appliances with multiple operating modes and calculated impedance for use in models.	103
Table 4-6: Properties of the Dimplex ECSd100-580 100 litre 3kW immersion heater.	104
Table 4-7: Description of the measurements of glazed areas or opaque panels that are required for the ISO10077-1 window thermal transmittance calculation.	113
Table 4-8: Description of the measurements of the frame that are required for the ISO10077-1 window thermal transmittance calculation.	114
Table 5-1: Results of room materials survey for first test home.	137

Table 5-2: Survey of the materials used in the building construction around the room marked
“Lounge 1” in Figure 5-8. 145

Table 6-1: Results of simulation and experimental comparisons of maximum power point
tracking algorithms under slowly and rapidly changing atmospheric conditions.
..... 164

Table F-1: Illustration of the Reverse Polish Notation solution of the expression in Figure F-4.
It is assumed that before solving this expression the variable “A” is set to 3.
Trigonometric functions are solved in degrees..... 212

List of Abbreviations

AC	Alternating current.
AMI	Advanced metering infrastructure (otherwise known as “smart metering”).
API	Application programming interface.
ASHRAE	American society of heating, refrigerating and air-conditioning engineers.
AWG	American wire gauge.
BEV	Battery electric vehicle.
C#	C#, a Microsoft programming language.
CAD	Computer-aided design.
CCS	Carbon capture and storage.
CHP	Combined heat and power.
CPP	Critical peak pricing.
DC	Direct current.
DLL	Dynamic link library – a Windows module containing re-usable software.
FPGA	Field programmable gate array.
GUID	Globally unique identifier.
HAN	Home area network.
HVAC	Heating, ventilation and air conditioning.
ISO	International standards organisation.
LAN	Local area network.
MPPT	Maximum power point tracking.
.NET	.NET, a Microsoft application development framework.
OSI	Open systems interconnection – a standard model for networking.
PAN	Personal area network.

PHEV	Plug-in-hybrid electric vehicle.
PLC	Programmable logic controller.
SEP	Smart energy profile – a ZigBee profile designed specifically for communication between smart metering equipment.
TOU	Time of use electricity tariff.
T-TOU	Tiered time of use electricity tariff.
UML	Unified modelling language – a standard method of presenting software design.
VB	Visual Basic – a Microsoft programming language.
WAN	Wide area network.
XML	Extensible mark-up language – a standard method for machine- and human-readable exchange of structured data.

Chapter 1

Introduction

A significant proportion of current research into energy systems is partially motivated by government targets that have been set to reduce carbon emissions as part of the overall goal to reduce the effect that our production of carbon dioxide has on climate change. A number of advances are being made in improving carbon emissions at the point of energy generation through the use of cleaner renewable sources of energy and through the use of new technologies such as carbon capture and storage for traditional fuel-fired power stations. Advances are also being made at the point of use such as the reduction of building energy consumption through education of occupants and through technological advances which improve the efficiency of appliances at the point of consumption.

The development of new technologies to reduce carbon emissions has, however, introduced new problems within energy systems which must now be solved. Electricity generation which has typically been centralised at large power stations is now moving to more remote parts of the electricity grid where the availability of wind, tidal and wave power is at its greatest. This introduces constraints at locations on the energy network where generation can exceed the rated capacity of a section of network which was never designed to support large generators. Increasing renewable generation also poses network stability issues whereby the export of renewable generators is dictated by the weather rather than by the network operators, which can result in availability of energy being high when demand is low and vice versa.

The term “Smart Grid” is used to collectively describe a set of technologies which aim to mitigate some of these problems that are introduced by adding large amounts of renewable generation to the electricity network. This broad term covers many aspects of grid operation which are being made more intelligent. However, the overriding goal of all of these

technological improvements is to enable the introduction of renewable generation sources into an electricity grid which was not originally designed for them. During the background research phase of this project, shifting demand to times of higher generation availability and introducing storage within the grid were the two broad approaches that were considered to be most widely accepted as a potential solution to the intermittency of renewable generation.

Additionally, other technologies such as low-carbon heating sources, electric vehicles, and improvements to building thermal efficiency were identified as changes which could be introduced at the domestic level as part of the overall transition to the smarter grid. These, coupled with the introduction of smart meters which provide a two-way communications link between the home and utility supplier and the wider availability of low-cost computing devices pave the way for the development of smarter home energy control systems.

After a detailed study of existing software applications for modelling home energy systems, it was concluded that much of the existing software available for this purpose takes a traditional system modelling approach, with focus placed on the accurate simulation of electrical and thermal properties of buildings. The goals of this project were therefore chosen to examine home energy systems from a software engineering approach, focussing on the control logic and communication issues involved in developing domestic smart grid systems, while accommodating the simulation of the physical electrical and thermal domains which these control systems ultimately interact with.

1.1. Aims and Objectives

Chapter 1 The overall aim of this project is to take a novel software engineering based approach to the modelling of domestic energy systems, with the focus placed on the control and communication elements of systems which will become available during the transition to a smarter energy network. Rather than develop new algorithms which operate at grid-level, this research aims to provide a tool that can be used to model energy systems within a home as part of a wider grid-level control system.

The following project objectives have been set in order to work towards this overall aim:

- Carry out a survey of existing software for domestic energy modelling and identify the features that are available in existing packages for smart grid modelling and which new features would be desirable.
- Use the set of requirements developed in the literature survey to design and implement a new domestic smart grid modelling package with a particular emphasis on the software and data transfer aspects of the system. The new package should be able to support the modelling of electrical energy flow and building thermal characteristics to allow control systems to be modelled in context.
- Perform theoretical and experimental validation of the new package, where relevant, to quantify the accuracy of simulations which are carried out within the package.
- Carry out a domestic smart grid case study using existing technologies combined with novel ideas to demonstrate the suitability of the package for use within an overall smart grid system deployment.

1.2. Original Contributions of this Research

The research presented in this thesis provides a number of original contributions to knowledge in the field. These are described below.

1.2.1. Flexible Software Platform

A flexible software platform has been developed for the modelling of smart domestic energy systems. As well as allowing for the simulation of electrical power flow and heat transfer within a building – features commonly found in existing building energy simulation packages – the package offers a number of novel features. The incorporation of data communication simulation allows for systems within the home to communicate with each other to allow the simulation of distributed control or monitoring systems. Human interaction with appliances is modelled to allow for the consideration of the effect that human behaviour has on domestic energy consumption. The package provides the ability to implement component and control system behaviour directly in fully-featured object-orientated programming

languages directly, in contrast to existing building energy simulation packages which generally rely on custom scripting languages or the implementation of additional software modules for complex logic. The development of this package is the subject of a conference paper [1].

1.2.2. Photovoltaic Maximum Power Point Tracking

A case study was carried out during the research that illustrates the features of the package. During the development of this study, a new learning-based photovoltaic maximum power point tracking algorithm was developed that offers improvements over some existing methods of maximum power point tracking. The maximum power point tracking algorithm is the subject of a conference paper [2]. An algorithm was also developed to obtain the operating point of a solar panel when the load resistance on the panel's terminals is known. This algorithm is the subject of a journal paper [3].

1.3. Outline

The remaining chapters of this thesis are as follows:

Chapter 2 presents the background to this research project. Section 2.1 describes the transition from the fossil-fuel based 20th century energy systems to the future low-carbon smart grid and describes the need for software modelling packages to model the domestic-level effects of these changes. Section 2.2 provides a comprehensive review of existing software that could potentially be used for domestic smart grids which provides the basis for the requirements of the new software package.

Chapter 3 describes in detail the design of the new software modelling package that was developed during this project. The decisions behind the various parts of the package design are discussed and UML models provided for various parts of the software. The chapter concludes with an overview of the architecture of the new package.

Chapter 4 describes the selection of components that were developed for use with the package to simulate the behaviour of building elements. These include electrical components

(generators, appliances and switching); thermal components (building structural elements, heating, ventilation and weather); communication components (data sources and data routing); and many other components to control logic within simulations.

Chapter 5 describes the testing that was carried out during and after the development of the software package, including the theoretical and experimental validation of the simulation results that were obtained. Two case studies were carried out within homes to compare the output of the package against real-world scenarios.

Chapter 6 presents a case study of a proposed domestic smart grid system which was developed using the new software package. This system includes an improved maximum power point tracking algorithm for solar panels and an integrated home energy system which is capable of utilising hot water and battery storage to modify the daily load profile of a home.

The work presented in the thesis is summarised in the conclusions in Chapter 7. The chapter describes the key findings of the project and the future improvements that could be made to each phase of the project.

Chapter 2

Background

This chapter describes the current changes in energy generation, distribution and consumption patterns that are taking place around the world which have provided the motivation for this thesis. A particular emphasis is placed on the UK. A review of the changes from the “top-down” energy systems of the 20th century to the new “Smart Grid” systems of the 21st century is performed. An analysis of how these changes may impact the design of domestic energy systems then follows. The chapter concludes with a study of existing integrated building energy analysis software with a focus on assessing the capabilities of packages for modelling smart domestic energy systems.

2.1. Smart Grids

2.1.1. 20th Century Energy Networks

In the UK, electricity has traditionally been generated at large, centralised power stations. Coal, oil, gas and nuclear power stations are capable of producing a variable power output up to a fixed maximum capacity when provided with a continuous supply of the necessary fuel. While hydro-electric plants rely on a renewable source of energy in the form of stored water, they can also be viewed as a form of generation which can be deployed when demand requires, as long as the required volume of water has been stored or is available.

Traditional power stations are connected to a high-voltage transmission network which transports energy around the country to where it is needed. Energy is then carried by lower voltage distribution networks to the end-users. Throughout this process, a large amount of control and monitoring is placed on the transmission side of the network. The energy leaving power stations is monitored for both billing and quality purposes and the voltage and frequency are monitored at multiple points around the transmission network to ensure that

they are kept within acceptable tolerances. Careful balancing of supply and demand is required to keep the grid operating within its required voltage and frequency range.

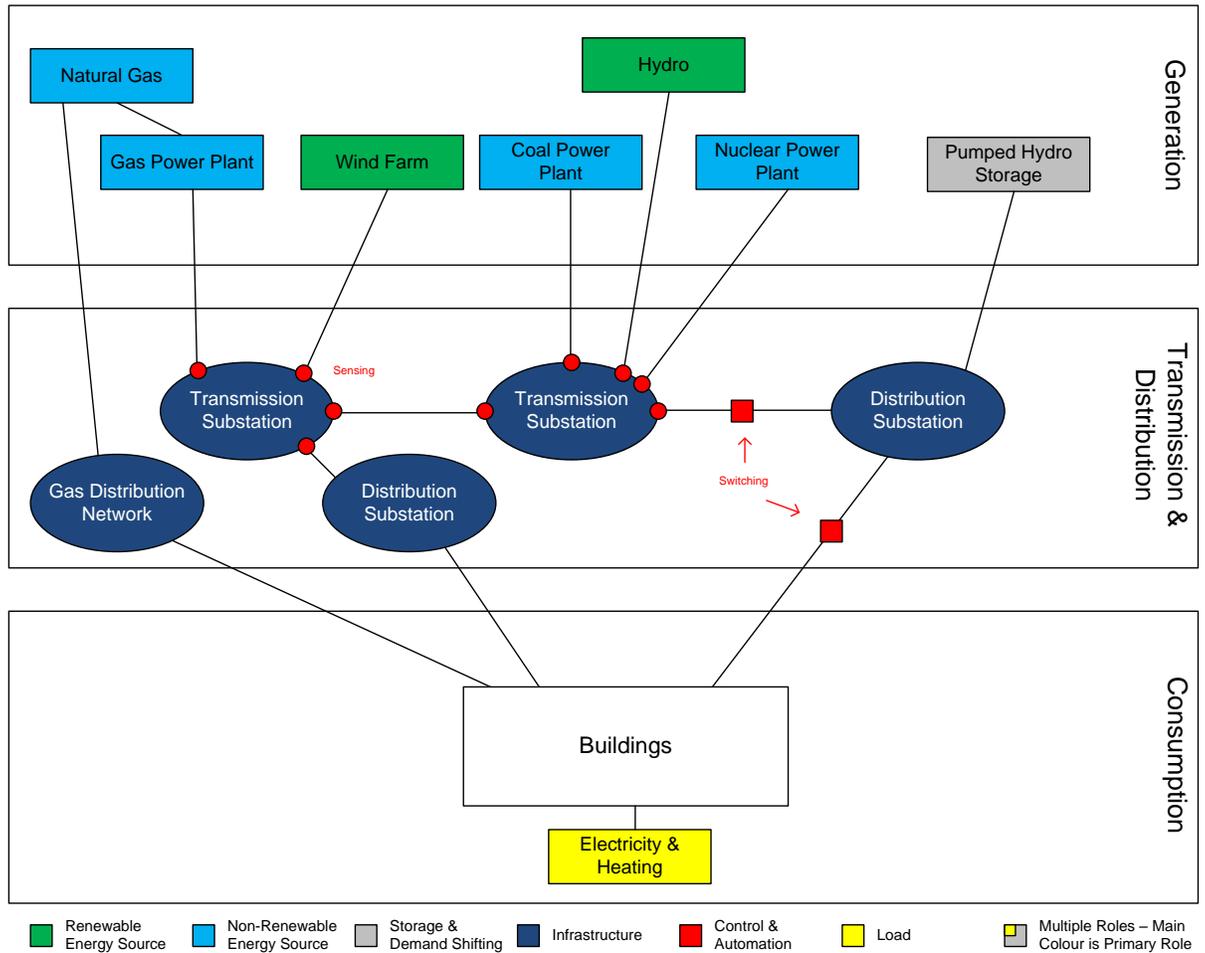


Figure 2-1: Diagram illustrating the current architecture of the UK's domestic energy system.

In contrast to a reasonably widespread use of automated control and monitoring on the generation and transmission side of the electricity network, there is significantly less control and monitoring on the distribution network. Electricity consumption at the domestic end-user level is carried out using either mechanical or digital meters from which the readings are periodically recorded by the utility company. As a result of this process, no information is available about the real-time use of energy on a per-user basis. An exception to this is for customers with Economy 7 tariffs. These tariffs use two meters in a time-switched configuration which provides energy companies with the ability to sell electricity at a cheaper

rate during a seven hour off peak window during the night. However, this primitive form of time-of-use tariff does not allow electricity suppliers to provide pricing which reflects real-time energy demand. Figure 2-1 above illustrates the approximate architecture of the UK's energy system from generation, through transmission and distribution to the end-user.

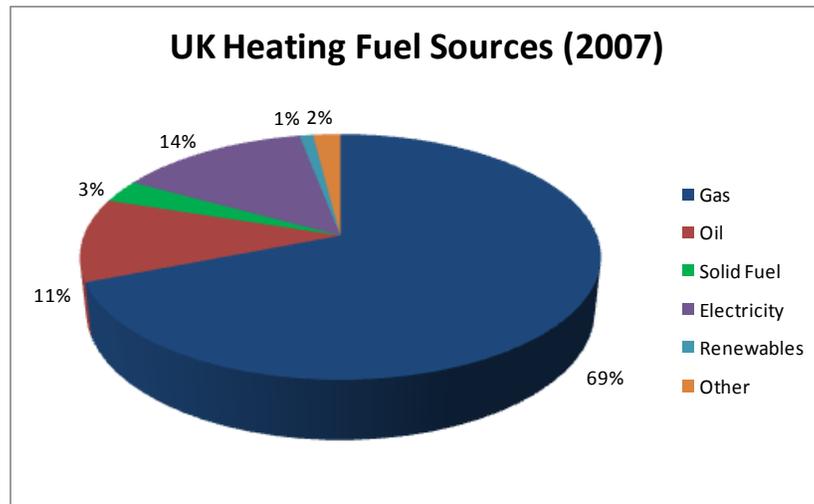


Figure 2-2: UK heating fuel mix in 2007 [4].

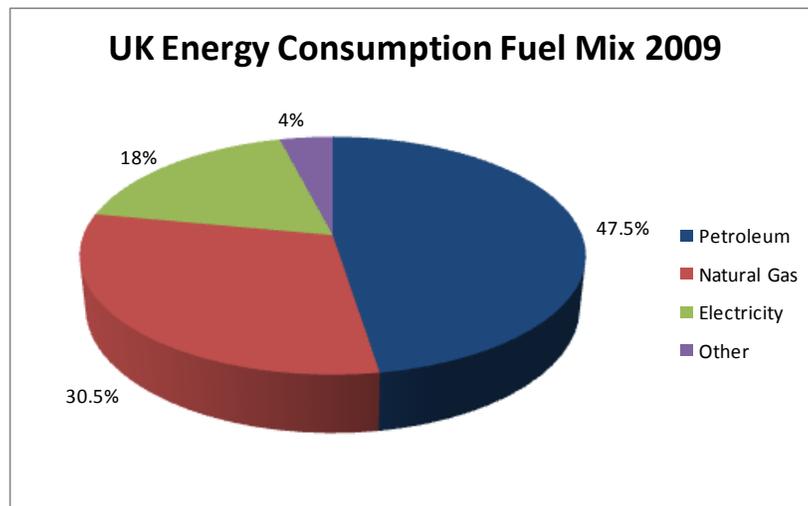


Figure 2-3: UK overall fuel mix for all energy sectors in 2009 [5].

Heating and transport also play a major role in worldwide energy consumption. In the UK, heating accounts for 46% of the overall energy budget with transport accounting for 37% [4]. Energy use statistics from 2007 shown in Figure 2-2 indicate that 69% of all heat energy was produced from natural gas, with the next highest source of heat energy being electricity with

a 14% share. Statistics from 2009 in Figure 2-3 show that the use of natural gas at the end-user level accounted for 30.5% of overall energy consumption whereas electricity only accounted for 18%. Figure 2-3 also shows that petroleum-based fuels which provide the majority of the UK's transport energy account for 47.5% of overall energy consumption. These statistics illustrate the large role that heating and transport play in energy consumption in the UK and also the extent to which the country relies on non-renewable fossil fuels to meet these energy requirements.

2.1.2. Low-Carbon Electricity

EU statistics from 2004 indicate that the UK is largely dependent on imported coal and, for the first time, is marginally dependent on imported natural gas due to depleting domestic stocks. [6] The dependence on depleting fossil fuel resources - especially those from foreign origins - coupled with European directives to reduce carbon emissions, has resulted in the UK government publishing a long term strategy which sets a target for producing 40% of UK electricity from low-carbon sources by 2020. [7] A proportion of this reduction will be made up by generating 30% of the overall electricity budget from renewable sources. Coal and gas use for generation with carbon capture and storage (CCS) in addition to new nuclear power stations could also feature as part of this plan. From a domestic end-user point of view, the main energy reductions dictated by this plan will be found by increased energy efficiency within the home. Financial incentives are also likely to be given to install low carbon electricity and heating systems, and energy companies will begin the roll-out of smart meters. The Scottish Government aims to exceed UK targets by generating 100% of electricity consumed in Scotland from renewable sources by 2020. [8] As of 2010, 24.1% of energy consumed in Scotland was produced from renewable sources, a slight decrease on the 27.3% figure of 2009 [9].

While the widespread incorporation of renewable generation sources into the power network is generally positive in terms of producing low-carbon sustainable electricity, its introduction also comes with some disadvantages. Renewable generation which depends on weather conditions, most notably wind and solar generation, are intermittent and unpredictable sources of energy generation. Hence excess energy may be generated at times

when it is not required, or conversely, not enough energy is generated when it is most needed. A recent example [10] of this phenomenon occurred in Scotland between April 5th and April 6th 2011 when larger than normal volumes of rain in conjunction with high wind speeds resulted in more wind and hydroelectricity being generated than was being used. This excess generation coupled with a network fault which prevented the export of excess electricity to England resulted in a number of wind generation companies being required to stop producing electricity and being compensated a total of £900,000 for the resulting loss of income.

Storage systems have been proposed in recent research as one method of smoothing out these intermittency issues [11-16]. A benefit of storage over other technologies for the mitigation of intermittency issues is that it does not require a behavioural change from end users – a continuous supply of energy is always available. A major disadvantage to widespread use of storage technology at present is the relatively expensive cost per kWh of electrical storage systems.

Another factor to be considered with a number of renewable generation systems – particularly solar power – is the non-linear voltage-current relationship of the generators. In the case of photovoltaic panels this means that for a given level of incident solar radiation, there is a particular voltage-current combination which will yield the maximum power generation. A large amount of research has been undertaken for various forms of renewable generation into methods to track maximum power points [17-25]. These maximum power point tracking algorithms (MPPT) generally operate by varying the load on the renewable generation system to increase the power yield from the system. This ideal load setting can be greater or less than the actual system demand.

2.1.3. Low-Carbon Heating

In section 2.1.1 it was stated that around 46% of the UK's domestic energy consumption is for heating buildings and water and that the primary domestic heating fuel in the UK is natural gas [4] - around 1000TWh of natural gas is consumed annually [5]. While coal-fired and gas-fired electrical energy generation with CCS is a viable option for producing cleaner

electricity from coal and gas on a larger scale, there is less that can be done to reduce emissions from gas on a domestic level. One idea proposed in the UK low-carbon transition plan is to replace existing gas boilers with more efficient models, therefore producing the same quantity of heat while producing a lower level of emissions due to a reduced fuel consumption. Other efficiency savings proposed include improving the levels of insulation in homes and installing triple glazed windows to reduce heat loss through windows.

While limited carbon reductions can be achieved through efficiency savings, in the long term different heating technologies will have to be used to achieve further reductions and to reduce dependency on foreign imports of fossil fuels. Solar water heating systems which use focussed sunlight to heat water pipes are already commercially available, as are ground, air and water source heat pumps which use a small amount of electricity to circulate fluid to recover stored heat. In areas where renewable sources of bio fuels are available in the form of timber, farm crop by-products or organic waste, combustion systems can be used for heating. While these heat sources may not be carbon-free, some of them may be considered “carbon-neutral”. An example of this is that when wood from trees is burnt, CO₂ is released, but trees which are planted to replace them are capable of breaking down CO₂ in the atmosphere. Over long periods of time, the carbon cycle of trees is therefore essentially neutral.

Another development in low-carbon heating technologies is the use of combined heat and power (CHP) systems which provide both heating and electricity. Small-scale systems use heat pumps, waste heat or combustion of fuel in conjunction with a heat-to-electricity conversion system such as a Stirling Engine [26-27] or thermoelectric generators [28-29]. Larger scale systems known as district heating schemes use a centralised heat source to provide hot water or steam for heating to surrounding homes and businesses through insulated pipes. While there are only a small number of district heating systems currently operating in the UK, this type of heat provision has been widely used throughout Europe for some time with a large number of European countries generating a major proportion of their domestic heating from such schemes. As illustrated in Figure 2-4, a number of European

countries are providing over half of their domestic heating energy from district heating schemes while Iceland produces nearly all of its domestic heating through district heating.

Some district heating systems rely on natural geothermal energy; others will use a readily available source of industrial waste heat such as the cooling water from a coal or nuclear power station while and others may rely on more traditional methods of heat generation such as the combustion of oil, gas or solid fuels. An increasingly common method is to use domestic waste as a heating fuel to reduce required landfill capacities. This method is used in a district heating scheme in Nottingham in the UK [30] to provide heating to a number of large commercial users and over 4600 domestic customers. Centralised heat generation systems are advantageous for upgrading to low-carbon heat sources because they do not require the expensive process of upgrading individual homes – the heat source only requires to be changed at the distributor level. An example of this is the PDHU project [31] in London which used waste heat from the now closed Battersea power station. When the power station closed, gas-fired CHP units and boilers were installed to continue providing heat to the connected homes.

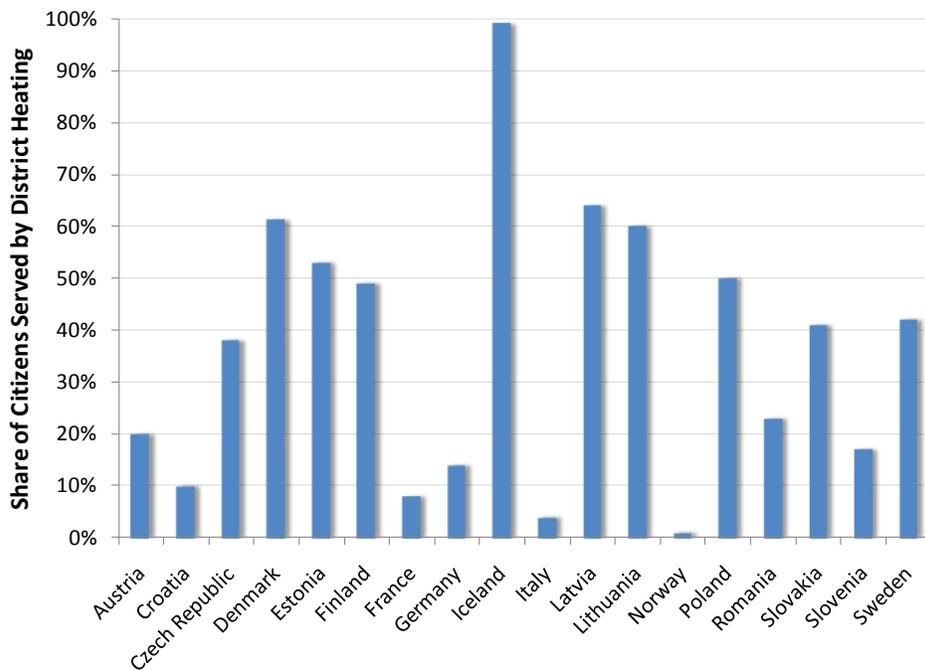


Figure 2-4: Share of citizens served by district heating schemes in selected European countries.

Multiple technologies have been described that could be used to make the transition to low-carbon heating in the UK, however, one factor which most of them have in common is that they require changes at the end-user level. This includes district heating schemes which, despite being commonplace in other European countries, would require a large initial investment in infrastructure in the UK. This is in contrast to the transition to low-carbon electricity sources in which the majority of changes can be made at the generation and distribution levels.

2.1.4. Low-Carbon Transport

In the document “Low Carbon Transport: A Greener Future” [32] published by the UK Department for Transport, a number of measures are outlined for the reduction in carbon emissions from the transport sector. The majority of the short-term measures described in this report focus on improving the efficiency of internal combustion engine vehicles to reduce their contribution to overall carbon emissions. However, the longer term measure of introducing lower carbon alternatives to existing vehicle technologies are of greater interest in the context of this work.

Plug-in hybrid electric vehicles (PHEVs) and battery electric vehicles (BEVs) which either partially or fully use stored electricity as their main energy source provide an alternative to internal combustion engine vehicles and offer very low or zero carbon emissions at the point of use. A project is currently underway in London [33] to encourage the uptake of these vehicles. As identified in the London project, the main barrier to the uptake of electric vehicles is the need for charging points to regularly charge the Vehicle’s battery. The project aims to address this by providing a large number of fast charging points (single phase 230V, 32A and three phase 500V, 200A) throughout the city in order to allow vehicles to be charged when they are away from the home.

In the context of this research, the interesting aspect of electric vehicles is when they are connected to the domestic power system for charging. In section 2.1.2, energy storage and intelligent scheduling of demand were presented as two technologies which can potentially be used to mitigate the power system stability issues posed by the intermittent generation

from renewable sources. Within the context of a smart home, an electric vehicle can operate as a schedulable load. Additionally, with the correct type of bi-directional charging system [34] the vehicle can operate as a battery energy storage system offering the ability to support the grid when renewable generation output is reduced. Two separate German studies have shown this use of electric vehicles to be a viable concept. The first [35] takes a conservative estimate that electric vehicles would be connected to the grid around 81% of the time and concludes that while unmanaged charging of vehicles increased peak grid load, charging can be managed in a way that shifts the load to a more desirable time for the grid with little or no effect on the mobility of the user of the vehicle. The second study [36], which takes a more optimistic estimate that vehicles will be connected to the grid 89% of the time, concludes that a reduction of up to 16% in the fluctuation of overall grid energy demand can be achieved with 1 million electric vehicles in use.

2.1.5. Advanced Metering Infrastructure

The installation of Advanced Metering Infrastructure (AMI) or Smart Metering as it is also known is an indication that the type of control and monitoring that was previously only present at the generation and transmission levels of the electricity supply systems is now filtering down to the end-user level. The UK smart meter policy [7] dictates that smart meters should be installed in all UK households by 2020. This exceeds EU policy which mandates that all EU countries should have deployed smart meters to 80% of the population by 2020. [37] A number of other EU countries also plan to exceed the requirement. Italy has already deployed smart meters to around 75% of its households and France has a penetration of around 25%. [38]

The initial aims of installing AMI as described in the UK low-carbon transition plan [7] are to encourage consumers to better understand their energy use and provide opportunities for energy saving. The government's intention is that the deployment will allow utility companies to provide alternative tariffs which can reward energy saving and encourage end-users to use energy when it is most desirable for the generation systems. Despite being government-mandated for these reasons, the installation of smart meters is also desirable

for the energy suppliers because it provides data which can be used to analyse usage patterns and also saves the manpower costs associated with reading traditional meters.

Despite these relatively modest initial requirements of smart metering systems, utility companies within the UK are already deploying meters which far exceed these requirements. British Gas have publicly released the specification for their smart metering system, which is developed by Landis+Gyr [39]. GPRS communication is used to implement a two-way connection between the meter and the billing company. This allows for the meters to either transmit readings at predefined intervals back to the billing system, or for the billing system to request readings from the meter at any time. Both prepayment and credit modes are supported with the option of various all-day, on- and off- peak and time-of-use tariffs with rate intervals as short as 30 minutes. A primitive form of load shedding is also available within the meters whereby a power consumption limit (for example, 5kW) can be imposed on the user. When power consumption is close to this level the meter will sound an alarm and if the limit is exceeded the meter will cut all power to the property.

The British Gas smart meter uses Zigbee Smart Energy Profile (SEP) [40] for local communication between the electricity meter, gas meter, communications unit and any in-home displays. This is an industry-standard communications interface for smart energy related systems and provides capabilities beyond smart metering such as the control of smart appliances, heating and air conditioning systems. A software upgrade to enable these features would allow the meters to dynamically control appliances in response to changing electricity prices.

Scottish Power have started to roll out smart meters to their customers which are also manufactured by Landis+Gyr and have similar capabilities to the British Gas meters [41]. Scottish and Southern Energy have also entered a partnership with Landis+Gyr to develop a smart meter with a particular focus on green energy [42]. This meter has the capability to allow the connection of local microgeneration to the grid as well as providing similar features to the meter already been discussed. OnStream, the National Grid's metering business, have also developed a smart meter which has the ability to automatically control home appliances [43].

A study into smart metering in the EU [38] has estimated that the installation of smart meters in every home in the EU will cost around €51 billion. The direct benefits from this installation to utility companies will be a saving in the operational costs of connection and disconnection of supplies and meter reading. This saving is estimated to be in the region of €26-41 billion which leaves a deficit of around €10-15 billion to recover. One of the main benefits of smart metering which the study proposes to meet this funding gap is the introduction of time of use tariffs (TOU) to reduce peak generation requirements.

Primitive time of use tariffs have been present in the UK for a number of years in the form of the Economy 7 tariff [44] which provides seven hours of discounted electricity during the night in return for paying slightly higher prices during the rest of the day. This tariff has typically been provided by using two meters with a clock or radio based time-switch. While it provides the ability to offset electric heating loads to a time of lower electricity consumption, it does not have the ability to encourage customers to reduce loads at times when this is critical to grid operation. The intermittent nature of renewable sources which was discussed earlier provides a greater motivation to be able to adjust the times of day that these types of peak and off-peak charges apply. The ability to reduce electricity load when renewable generation levels are low reduces the need for expensive grid-based storage as described in section 2.1.2.

Utility companies around the world, particularly in the USA, have developed time-of-use tariffs which are specifically designed to address the issue of reducing load at times critical to grid operation. In the state of California, a time of use pricing pilot project was carried out using four separate time-of-use schemes [45]. The most basic scheme trialled was traditional TOU that involves charging a price for a fixed peak period each day which is around double the price of the off-peak periods. This is comparable to the Economy 7 scheme already in use in the UK. The remainder of the schemes trialled are of a class known as critical peak pricing (CPP). These schemes charge between five and six times the off-peak price for a small number of critical hours during the day when the wholesale price of electricity is at its highest. Two variants of this scheme that were trialled were one where the critical hours were fixed and one where the critical hours could vary and participants were notified each

day when the critical priced hours, if any, would be. The advantage of the second scheme is that it is able to respond to forecasts of renewable generation. Additionally, a fourth trial was carried out where the CPP tariffs were assisted by smart thermostats which automatically control air conditioning systems based on price. The conclusions of this study were that the standard TOU scheme resulted in peak time energy use reductions of around 5% while the CPP tariffs resulted in a reduction of 8-15%. In the trial where smart thermostats were also used, peak consumption was reduced by as much as 25%. Another trial of CPP in Illinois which used email or SMS to communicate information about peak periods to customers each day showed similar results to the Californian trial, with peak reductions of around 15% [46]. It was noted that customers in this trial were more responsive to peak pricing events in the evenings. This is due to the fact that no automating technology was used and therefore customers had to be at home to be able to respond to price changes.

2.1.6. Smart Grids Summary

The primary motivation behind the transition to the smart grid from the existing power distribution networks of the 20th century is the increasing political pressure to move to low-carbon energy systems. As discussed in 2.1.2, the transition to renewable electricity sources introduces more intermittency into the electricity supply which requires either storage or demand management technologies to ensure a reliable supply when climatic conditions are causing a low yield of renewable energy. In 2.1.5, the concept of AMI was discussed as a possible solution to the problem of shedding load from the power network when renewable energy generation is low. Smart meters using price signals sent from energy companies, coupled with emerging technologies such as smart thermostats and smart appliances can allow domestic end-users to play a more interactive role in the overall energy network. In 2.1.3, renewable heating technologies were discussed. Of particular note was the fact that changes to renewable heating systems would require significant infrastructure changes at the end-user level as heat energy is generally produced where it is needed rather than centrally distributed.

In conclusion, the transition from the current energy system architecture shown in Figure 2-1 to a new Smart Grid system will involve extensive introduction of control and monitoring at

all stages of the energy system, from generation to the end user. New methods of renewable generation known as distributed or embedded generation will begin to be connected at remote points throughout the grid and microgeneration and storage devices may be connected at the consumer level to reduce dependency on grid supplies. Changes to buildings in the form of new heating systems and energy efficiency measures such as insulation, draft-proofing and window replacement may also be required. A possible architecture of this new grid is shown in Figure 2-5.

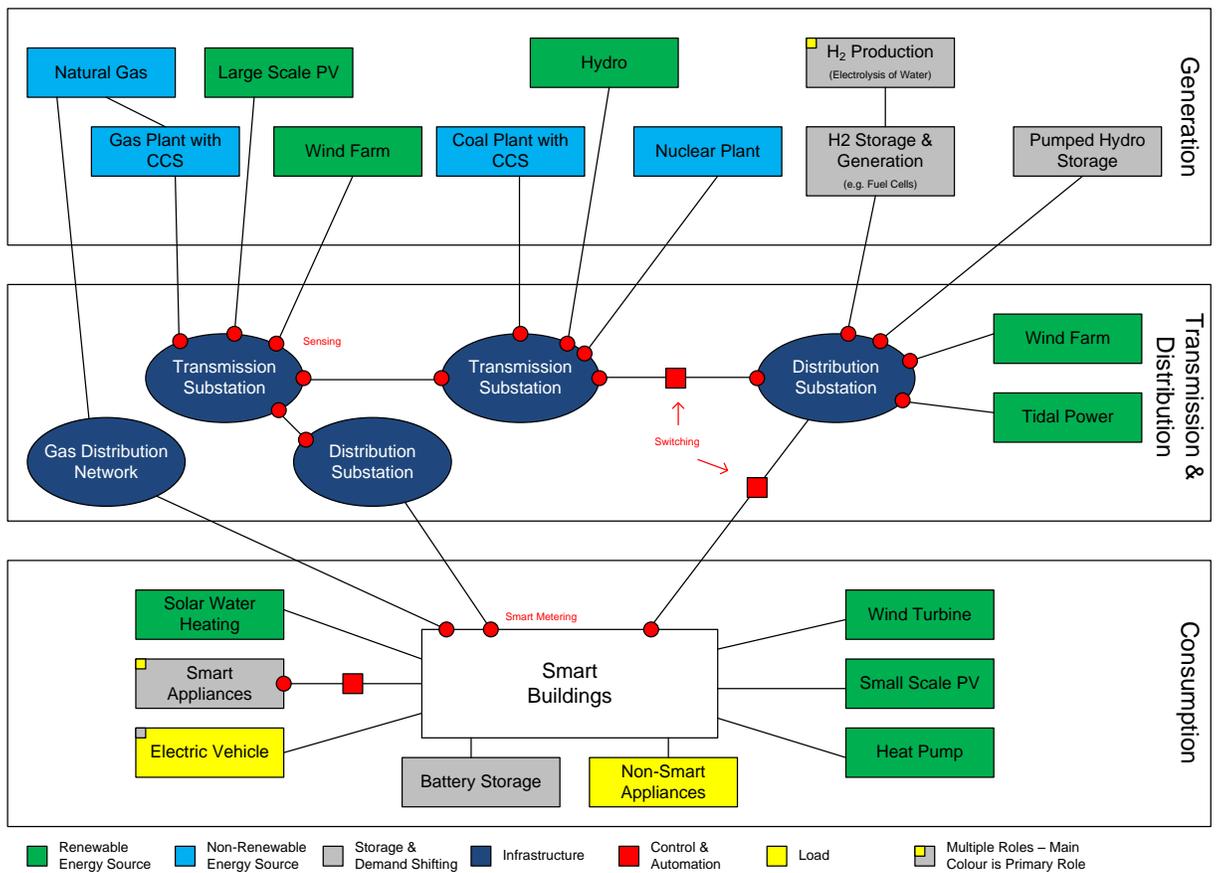


Figure 2-5: Possible architecture of the future Smart grid, based on one proposed by the UK electricity networks strategy group. [47]

2.2. Smart Grid Software Modelling

As discussed in section 2.1, the transition to Smart Grid systems will involve changes at the domestic and commercial end-user level in the form of smart metering systems, demand management, distributed generation, distributed storage and new methods of heating.

These changes involve new challenges when designing domestic energy systems as the flow of energy is no longer unidirectional from the electricity or gas supply system to the consumer. The use of energy from sources other than the grid will become more common and the net energy flow may be either from the grid to the home or from the home to the grid depending on installed microgeneration or storage capacity. Additionally, different control strategies may be implemented to allow the energy supply network to schedule the use of various domestic appliances and storage devices to coincide with times dictated by available grid capacity.

The introduction of smart grid technology will require an ever-increasing number of design decisions to be made when implementing domestic energy systems. Traditionally thought of as separate entities, electricity and heating will become more interdependent with the increased use of new technologies such as combined heat and power systems and electric heating from renewable electricity sources. Such interdependency will see new types of design criteria and constraints being considered when developing domestic energy systems. For example, when designing a single-building microgeneration and electricity storage system, maintaining a comfortable heating temperature while managing the use of electricity from the grid may be a design parameter. Criteria relating to the “smart” aspects of the future grid will also need to be considered. This would include communications between domestic appliances, price signals from the utility network and software control algorithms to co-ordinate the local use of energy.

Software modelling and simulation techniques have been widely applied to the solution of system-level engineering problems in the past and it is therefore sensible to investigate their application to the emerging concept of smart grids. In investigating software modelling tools, it becomes apparent that the most appropriate category of tools for use with domestic smart grid systems are tools known as building energy analysis tools. Building energy analysis tools use some form of description of a building’s construction in addition to other input parameters to perform analyses of the building’s energy usage. Inputs and outputs vary from tool to tool, however typical inputs may be: building construction materials and dimensions; weather data; local electricity generation and usage; choice of heating fuel; heating or air

conditioning system parameters; and occupancy data. Typical outputs are likely to include: energy efficiency rating; heating fuel and electricity consumption details; room temperatures; and costs. These tools are particularly appropriate for smart grid modelling due to their support for electricity, heating and control system analyses.

A comprehensive comparison of building energy analysis tools is given in [48]. Rather than reiterate the findings of previous comparisons in this thesis, a survey of the literature on one of the most commonly used tools, EnergyPlus, will be presented to give the required insight into the techniques widely used in this type of software. EnergyPlus, which is developed by the US Department of Energy, was selected specifically because of its widespread use internationally for building energy analysis and also because of the wealth of published research on its operation. The review of EnergyPlus will be followed by brief summaries of a number of other building energy analysis packages to compare the features available in each. The section will conclude with a summary of the current state of the art of integrated building modelling tools.

2.2.1. EnergyPlus

EnergyPlus [49-50] is a comprehensive building energy simulation tool which is developed by the US Department of Energy. It is based on features from two separate packages – DOE-2 [51] and BLAST [52] – which were its predecessors. The main purpose of these two packages is the simulation of heating, ventilation and air conditioning (HVAC) systems and as a result, the core capability of EnergyPlus is a very mature HVAC systems simulation. Nonetheless, the package also has the capability to model electrical loads and generation including some complex analyses such as the yield of renewable energy systems at a particular location.

EnergyPlus is a simulation engine which is designed to accept its input and generate results through text files. Inputs to the system are a complete model of the building's construction, definitions of the HVAC and electrical plant used in the building, and weather data for the building's location. Results including plant utilisation, heat flows, room temperatures and total energy demand are provided in similar text files. The purpose of using text files rather than a full user interface is to allow third parties to develop their own interfaces that are

suited to their particular problem, and to allow EnergyPlus to be integrated into larger simulation systems. The Department of Energy develop a graphical interface for the package which allows building models to be rapidly developed in Google Sketchup [53] and simulated using EnergyPlus. This interface, known as OpenStudio [54], permits the whole process of defining the building model, its simulation and viewing the results to be carried out from Sketchup. Various architectural design software packages also provide their own interfaces to EnergyPlus [55].

Simulations within EnergyPlus are undertaken in two distinct steps. For each time-step in the simulation, a heat balance model calculates the temperatures of zones and surfaces within the building in addition to the heat flow between zones and surfaces. The results of these calculations are then fed into a building systems model which iteratively evaluates the response of the building systems to the changes in temperature that were calculated.

The heat balance model within EnergyPlus is a single-pass calculation rather than an iterative solver. The model takes the states of building zones and plant from the previous time-step of the simulation and uses these to compute the changes in temperature and heat flow for the current time-step. Two distinct modules are utilised for this purpose. The air mass module deals with heat flow due to moving air, either through forced ventilation or infiltration. This module assumes that in each zone of the building the air is of uniform temperature. The surface mass module deals with conduction, convection and radiation of heat from surfaces. This module assumes that surfaces have uniform temperature, uniform radiated heat and one-dimensional conduction. Comprehensive coverage of the effects of windows on both heating and lighting of rooms is included in the heat balance model [56].

The building systems model is an iterative solver which represents the HVAC and electrical plant within a building. Building systems are simulated using a concept known as “loops”. A loop is a set of connected nodes which, in the case of HVAC systems, model heating and cooling plant connected by pipes or, in the case of electrical systems, model electrical generators and loads connected by wires [57]. At present two HVAC loop types are supported within the package – air loops and water loops. The iterative solver for HVAC loops defines loops with two halves – a demand side which provides the heating or cooling to

zones, and a supply side which provides the heated or cooled air or water. Temperature set-points are used for each zone to calculate the required heating or cooling input from its HVAC equipment. The solver then iteratively tries different configurations of all of the available plant within the room until the supply side of each HVAC loop can meet its demand side. Users can add control algorithms to this process to control the temperature set points of the rooms.

Electrical systems within EnergyPlus are also modelled as loops [58]. These loops can be either AC or DC electrical networks with a combination of loads, storage devices and generators. Conversion elements such as electronic inverters are included to bridge between DC and AC loops. There are a number of built-in generation modules including photovoltaic, wind, combined heat and power and internal combustion engines. Additionally, a number of control strategies for electrical systems are also included in the package which control the dispatch of generation and storage devices in response to both electricity demand and the target amount of power to be drawn from the grid. This allows for control strategies which encourage power consumption when grid prices are low and discourage consumption when prices are higher.

To conclude the study of EnergyPlus, consideration must be given to the accuracy of the results from its simulations. The package is tested using the ASHRAE standard 140-2007 (an update to ASHRAE 140-2001 [59]) which defines standard test scenarios for the benchmarking of building energy analysis software [60]. Results have been published which indicate that the results from EnergyPlus correlate well with those of other simulation packages [61-64]. In all of the tests presented in these results, the only situation in which the results from EnergyPlus fell significantly outwith the range of other simulation packages was in seven out of the thirteen building envelope tests where results varied by up to 15% compared with other packages. In the HVAC and fuel-fire furnace tests, EnergyPlus produced results that were within 2% of those of other packages.

2.2.2. Other Building Energy Analysis Software

As already described in section 2.2.1, the **BLAST** [65] and **DOE-2** [51] packages have been superseded by EnergyPlus. These packages contained a subset of the features available in EnergyPlus and were both focussed on the simulation of the thermal properties of a building and HVAC systems rather than the overall energy usage of a building. Similarly to EnergyPlus, both of these packages accepted input and produced output in text file format with a number of third parties producing graphical user interfaces.

ESP-r [66-67] is a complete building energy analysis tool which has been under development for over thirty years at the University of Strathclyde. The capabilities of ESP-r very closely match those of EnergyPlus in that it is capable of modelling heat transfer between building zones and surfaces, HVAC systems, mass flow of air and water, sunlight and shading and electrical power flow. In addition to the 1D heat conduction model that EnergyPlus employs for surface heat transfer, ESP-r offers optional 2D and 3D conduction models for more accurate simulations. An optional computational fluid dynamics (CFD) numerical approach [68] can also be used within zones in contrast to the standard uniform air temperature model to gain a greater understanding of the temperature gradients across each zone. ESP-r uses similar solvers to EnergyPlus for the mass flow of air and water throughout a building whereby a node-based representation is used to generate a model of the system [69]. Electrical power flow within a model is solved using a frequency-domain power flow solver [70] to obtain the steady-state power for each time-step within the simulation. The concept of “hybrid components” is used to enable components which convert electricity to heat and vice versa to be represented. These components interact separately with both the thermal and electrical solvers within the package. Definition of models within ESP-r can be carried out using a basic graphical user interface and also through the import of building geometry from CAD packages. A number of third party components are used to visually present simulation results. Numerous studies have been carried out by the program’s development team in order to validate the results produced by the package [71]. These studies include analytical comparisons to ensure that the package produces the expected results according to the theory that it is based on, empirical studies to compare the performance of the simulations

to real-world data, and comparative studies to compare the performance of ESP-r to other packages. In these validation studies, the package has been found to be in agreement with other similar packages.

BSim [72] is a suite of programs which is developed by the Danish Building Research Institute for undertaking building energy simulations. Standard features of the package include the ability to simulate heat transfer between building zones, zone temperatures and humidity conditions. The effects of sunlight on a building's climatic conditions and internal light levels, including the effects of shading by surrounding buildings can also be simulated. BSim does not provide any comprehensive coverage of electricity use within the modelled buildings. Add-ons for the package allow for more advanced simulation of moisture conditions within buildings [73-74] and for simulation of the approximate yield of photovoltaic panels installed on the building [75]. One of the programs within the suite is a graphical user interface [76] which allows for the definition of building models using a 3D view or 2D plan view. Another program within the suite offers the capability to import building models from CAD packages [77]. While the package has not been explicitly validated, the algorithms which make up the main part of its functionality have been validated in a previous version of the simulation engine.

The UK Building Research Establishment develops a simulation tool called **SBEM** (Simplified Building Energy Model) with graphical user interface in the form of a Microsoft Access database with macros called **iSBEM** [78-79]. Rather than being a complete building energy analysis tool, SBEM is designed to test the compliance of buildings with EU and UK building and energy efficiency regulations. This compliance is tested using a simplified set of calculations known as the National Calculation Method (NCM). Two versions of the calculation method are defined: one for Scotland [80] and another for England and Wales [81]. As the tool is primarily a compliance and energy efficiency, its outputs are whole-building energy metrics rather than detailed calculation of room and surface temperatures and electricity usage. The developers of the tool therefore indicate that it should not be used in design simulations and that other tools should be used for this purpose.

Energy-10 [82] is a proprietary software package which is aimed at assessing the effect of installing different energy efficiency measures during the design of small residential or commercial buildings. The focus of this package is on rapidly modelling buildings and running hourly simulations to assess the benefit (both environmental and economic) of installing various energy efficiency measures. It could therefore be described as a tool which assists in decision-making rather than a complete energy analysis tool. Little information about this package is available in the public domain due to its proprietary nature, however, available features are: daylight simulation, passive solar heating and cooling, natural ventilation, insulation, high performance windows, lighting and mechanical equipment.

The “**Transient System Simulation Program**” or **TRNSYS** [83-84] is a proprietary software package which is developed by the University of Wisconsin-Madison in partnership with a number of industrial partners. The package is a modular, general-purpose transient simulator. However, the majority of the built-in mathematical models reflect the fact that it is very much targeted at the modelling of energy systems. In contrast to the whole-building definition approach, TRNSYS provides a more flexible method of modelling where individual components which are represented by mathematical models that can be connected together to form a complete system. In simple terms, the output of one mathematical model in the system becomes the input to others. The TRNSYS solver performs a simultaneous solution of the algebraic and differential equations which make up the model on each time-step of the simulation and records the results [48]. Systems are designed using a graphical user interface by adding mathematical models (either user-defined or from the built-in library) and making connections between them with lines to indicate the flow of data. The OpenStudio plug-in for EnergyPlus [54] which provides integration with Google Sketchup is also supported by TRNSYS, allowing a geometrical description of a building to be automatically converted into a mathematical model block for use in the package.

Matlab [85] is a mature mathematical programming environment which has gained widespread acceptance in the engineering community. The Simulink [86] simulation environment is a model-based simulation system which accompanies Matlab and enables the creation of Matlab programs from a graphical drag and drop based modelling interface. From

a usability perspective, Matlab and Simulink provide a good environment for the rapid development of system models. The SimPowerSystems package which is available for Simulink provides a mature power system modelling capability and the Communications System Toolbox provides a comprehensive communications system modelling capability, from the high-level protocol level down to the physical link level. Where the package falls short in meeting the requirements for domestic smart grid simulation is its lack of building simulation capabilities.

2.2.3. Summary of Smart Grid Software Modelling

A comprehensive review of the current state of the art in integrated building energy analysis tools has been carried out and it is apparent that a number of the tools have at least some of the capabilities that will be required of simulation tools to model domestic smart grid technologies. To compare the capabilities of the packages that have been described and to identify any areas in which improvement is needed, a standard set of criteria have been developed. These criteria will focus heavily on the abilities that will be required to model new domestic smart grid technology within buildings, rather than to accurately model the buildings themselves. The criteria which are described below have been selected with careful consideration given to the review of smart grid technology carried out in section 2.1.

Core Features

- **Individual appliance modelling** – Simulations should be able to model down to the individual appliance level as smart grid control strategies may rely on controlling smart appliances to manage demand. In this case, the characteristics of each individual appliance are important in dictating the ways in which the appliance can respond to control signals.
- **Basic “lumped” models** – As well as advanced individual appliance models, it may also be desired to lump the behaviour of a number of appliances into one simplified block.
- **Closely coupled electricity, heat transfer and communications** – Every element within a simulation should be able to interact with electricity, heat transfer and communications simulations concurrently. This capability will be important in

modelling smart electrical appliances which may also have some thermal mass or thermal output to their environment.

- **Advanced software control of device behaviour** – The ability to script the properties of devices will allow more accurate modelling of their behaviour than a mathematical model and will allow the implementation of “smart” control strategies which communicate with other elements in the simulation.
- **Rapid development** – Packages should support rapid definition of models to aid in the process of prototyping different strategies.
- **Minimal constraints on system type** – There should be no constraints on the type or scale of the system being modelled. When modelling smart grid technologies it may sometimes be necessary to model a whole building or community or a small number of interacting buildings. In contrast there may also be situations in which modelling is focussed on developing an individual piece of control equipment.
- **Wide simulation time-step range** – Detailed simulations of control systems such as maximum power point trackers for renewable energy systems or battery management systems may require microsecond resolution whereas whole building systems may require only hourly, daily or seasonal simulation. Packages should accommodate this by offering simulation time-steps in the range of microseconds to months.

Environment

- **Weather** – The package should support the modelling of the effect of weather on buildings and their systems.
- **Importing weather data** – To assist with the modelling of climate, packages should be able to import weather data.

Electricity Modelling

- **AC and DC Systems** – Renewable energy sources and storage devices primarily utilise DC while domestic power systems are primarily AC and therefore both types of system and the conversion between each should be supported.

- **Renewable Generation** – As explained in section 2.1.3 of this review, distributed renewable generation including CHP systems will be a core feature in future climate policies and therefore its inclusion in modelling domestic smart grid systems is crucial.
- **Grid Connections** – The assessment of import from and export to the electricity grid will be important in studying demand management algorithms. A grid connection and metering capabilities are therefore required.
- **Electrical Storage** – Electrical storage systems will be important for both demand management and smoothing the intermittency of local renewable sources.

Thermal Modelling

- **Building Envelope** – A building’s construction plays a key role in its energy usage and therefore should feature in a smart grid modelling package.
- **Heat Conduction** – Heat conduction between spaces is an essential part of the building envelope modelling capability.
- **Infiltration and Natural Ventilation** – Should also be an integral part of the building envelope modelling capability.
- **Heating and Cooling Systems** – Modelling of both the central plant (e.g. boilers) and heating elements (e.g. radiators) which perform conversion of energy to produce heat.

Communication

- **Internal Communication** – Support for communication between appliances and control systems within the building is required to model entities such as smart appliances.
- **External Communication** – Support for communication with the utility company is required to model smart metering systems and dynamic pricing tariffs.

The above set of criteria are not exhaustive but serve as a basic set of requirements for a domestic smart grid modelling package from which further requirements can be drawn. They do, however, serve as a useful standard set of test criteria for performing an at-a-glance comparison of the available features in the packages that were studied in sections 2.2.1 and

2.2.2. Table 2-1 shows this comparison and was developed from research carried out in preparing this literature review and through existing comparison tables available in [48].

Table 2-1: Comparison of the domestic smart grid modelling capabilities of whole-building energy analysis tools. A solid dot indicates that the package fully supports the specified feature, a hollow dot indicates partial support and no dot indicates no support.

Criteria	Packages								
	Energy Plus	BLAST	DOE-2	ESP-r	BSim	SBEM	Energy -10	TRNSYS	Matlab
Individual appliance modelling	○	○	○	●				●	●
Basic "lumped" models	●	●	●	●	●	●	●	●	●
Closely coupled electricity, heat transfer and communications				○				○	
Advanced software control of device behaviour ¹	○			○					●
Rapid model development ²	○			○	●	○	●	●	●
No constraints on system type ³								●	●
Wide simulation time-step range ⁴	○			○				●	●
Weather / climate modelling	●	●	●	●	●	●	●	●	
Importing weather data from available formats ⁵	●	●	●	●	●	○	●	●	
Modelling of AC and DC systems	○		○	○				●	●
Renewable generation	●			●	○	○	○	●	●
Grid Connections	●			●				●	●
Electrical storage	●			●				●	●
Building envelope modelling	●	●	●	●	●	●	●	●	
Heat conduction between zones and surfaces	●	●	●	●	●		●	●	
Infiltration and natural ventilation	●			●	●	○		○	
Heating and cooling systems	●	●	●	●	●	○	○	●	
Internal communications between household devices									●
External communication for weather data and utility company connections									●

¹ Rather than using mathematical models, the application is capable of modelling systems through scripting or other means to allow for greater degrees of detail in component implementations and communication between components within a system.

² The application provides a graphical method of constructing models.

³ The application is not constrained to building modelling; it can model multiple buildings or individual building systems.

⁴ The application is capable of assessing both transient effects along with hourly, monthly or seasonal effects.

⁵ The application can import weather data in a non-proprietary format.

2.3. Summary of Project Background

Table 2-1 shows that some of the applications studied – most notably EnergyPlus, ESP-r and TRNSYS perform 80-90% of the requirements stipulated for a smart grid modelling package but that they are missing crucial features like the ability to model communication, advanced software control of devices within a building and the ability to model components which closely couple communications, heat transfer and electricity. This justifies the development of future tools which are specifically targeted at domestic smart grid systems. The set of criteria discussed in section 2.2.3 and listed in Table 2-1 will be used as the set of requirements on which to design a new smart domestic energy simulation package which is fully described in Chapter 3.

Within the scope of this project, a particular emphasis will be placed on the development of a platform that more tightly couples the interaction between the electrical and thermal physical domains and places a strong emphasis on the integration of software control within the home. The intention of this approach is to enable the merging of two branches of research discussed in this section – the need for smarter control systems to solve network-level constraints related to the introduction of renewables and the integration of these systems within the home.

Chapter 3

Software Design

In Chapter 2, it was established that while there is currently a broad range of open-source and proprietary software available to perform the modelling of some of the elements of a domestic smart grid system, there is still room for improvement in their capabilities. This deficiency has contributed to the motivation behind the main piece of work in this thesis – the development of an integrated simulation suite for domestic smart grid systems. An aim of creating such a simulation tool is to provide an environment that is capable of modelling the electrical and heat energy requirements of buildings in a similar way to existing tools but which also adds the element of communication between building systems and intelligent control within appliances themselves.

This chapter presents a detailed design for the new simulation package, using the criteria listed in Table 2-1 as a set of functional requirements. The design takes into account the methods that will be used to represent physical systems within the computer software; the methods that will be used to simulate AC and DC electrical systems, thermal systems and communication systems; and the methods used to run simulations and analyse the results. The chapter concludes with an overview of the design decisions and compromises made and presents the architecture of the software package.

3.1. Choice of Application Development Platform

The Microsoft .NET Framework and C# programming language were chosen as the main development platform for the application. The motivation behind this decision was that C# is

a high-level object orientated language which makes it suited to writing large, structured software applications.

There are a number of alternatives to C# when selecting an object orientated programming language, most notably C++ which can be used to program natively, and Java which targets the Java Virtual Machine, an alternative to .NET. The reason for selecting C# over these other languages was the selection of features provided by the .NET framework which seemed particularly suited to the development of this application. These were: XML reading and writing for model storage; windows graphical user interface libraries; the ability to invoke native code for optimisation where required; graph drawing libraries; and the ability to compile and run source code at runtime.

In terms of performance, native code written in C or C++ would be considered by most to be faster than code run on a virtual machine such as .NET or Java code. This was indeed found to be the case in a recent benchmark test [87] where, on average, code produced by the Microsoft C++ compiler was 15.8% faster than the equivalent .NET code. However, it was felt that this was an acceptable sacrifice in performance in return for the large selection of libraries provided by the .NET framework and the shorter development time.

3.2. Representation of Models

3.2.1. Techniques in Existing Packages

There are two perspectives to consider when selecting a suitable representation of the physical model: the representation that the user is required to use to present the model to the simulator; and the internal representation within the simulation package itself. Within the simulation packages discussed in Chapter 2, there are two distinct methods of model representation used. These are discussed in this section and used as the basis to specify a model representation for the new simulation package.

The more traditional building energy analysis tools which have their origins in HVAC simulation (EnergyPlus, ESP-r, BLAST, DOE-2 and BSim) require the user to provide input to the simulator in the form of a textual or CAD-based building geometry which is processed by

the package into a mathematical model that can be solved by the simulator. HVAC or electrical plant is represented separately in the form of a behavioural transfer function for the specific piece of plant and indications of the associated building zone and other plant that are connected to it. This particular model representation performs well when separate simulation subsystems are used to model the behaviour of the building and of the appliances within it. A disadvantage of this type of approach is that the user is constrained to modelling systems which fit the scenario of one or more building structures, each with a set of HVAC and electrical plant.

The packages which are more orientated towards system simulation (TRNSYS and Matlab/Simulink) represent each element within a model as a component which uses a set of mathematical transformations on its inputs to produce one or more outputs which can be fed into other elements of the system. Using this method of simulation requires the user to perform the transformation of the building geometry into a set of components with mathematical representations, possibly by using an automated tool. This requirement is a potential disadvantage of this approach as it requires more work on the part of the user when creating the initial model of the physical system. However, representing an entire system as a single mathematical model rather than using individual models for the different physical domains can also be advantageous. From the perspective of modelling domestic smart grid technology, it offers an advantage because multiple types of physical systems (e.g. electrical power flow, heat conduction, air flow, liquid flow, daylight) can be modelled without having a dependency on the capability existing within the package. This mathematical modelling technique is also more flexible in the types of system that can be modelled – a model could contain a single solar panel connected to a battery or could contain an entire street of buildings, each with its own smart energy system.

3.2.2. Design of New Technique

When designing the system modelling technique to be used in the Smart Grid modelling package, a combination of the two broad strategies discussed in section 3.2.1 was used. The connected-component approach of the latter strategy was chosen over the textual or CAD-based building geometry model of the former because it provides a greater flexibility in the

type of systems that can be modelled – the software is not constrained to building simulation. This strategy also allows the simulation of the electrical, thermal and communication domains to be more closely coupled whereby a single component within the simulation can contain logic that, for example, changes its electrical characteristics based on incoming communications.

Despite adopting this integrated modelling strategy, the concept of having separate simulation of electrical, thermal and communication systems within the package has been partly employed. The approach that has been taken is for every component within a simulation to have multiple connections, each with a defined type – electrical, heat transfer, communication or “other”. Each connection on a component may only be connected to a connection of the same type on another component. An advantage of this approach is that it allows the simulation package to record results easily from known types of connection. For example, electrical connections can support the monitoring of voltage, current and power flow; heat transfer connections can support the monitoring of heat flow and temperature; communication connections can support the monitoring of data transmission. The “other” type of connection allows for the custom implementation of physical domains which do not fall into the categories natively supported in the package – for example the flow of natural gas into a heating appliance.

A graphical user interface is provided to allow the user to create models in a similar style to Simulink and TRNSYS, where the user adds components to a model in the form of rectangular blocks and creates connections between them with lines. The behaviour of components can either be defined by a user or can be selected from built-in libraries of standard components. This allows for more rapid prototyping of systems by providing a more intuitive approach to designing the system than a textual definition language, giving constant visual feedback to the user of the structure of the system being modelled. Additionally, this approach allows for validation of the model as it is created by the user by preventing incorrect connections being made between components at design-time rather than having to subsequently validate the properties of the model during the running of the simulation.

Internally within the software, the user's graphical model will be mapped onto an object-orientated software representation of the model as shown in Figure 3-1 (page 36). This representation holds three collections of objects. The first contains each of the component types or behaviours which can be found in the model. For example, if a "Battery" component is used multiple times in a model, the number of external connection pins, the configurable parameters and the internal logic for the component need only be defined once regardless of the number of times that instances of the component are used. The second collection is a list of the specific instances of components that are used in the model, each with an associated behaviour and values for its configurable parameters. The final collection is a list of the connections that are made between components in the model.

To illustrate this mapping between graphical system models and an object-orientated software model of a system, a simple example system is illustrated in Figure 3-2. This system contains an electric heater connected to a grid supply which is used to heat a room. Figure 3-3 illustrates the properties of the software model that would be derived from such a graphical model.

3.3. Physical Domain Representations

In section 3.2.2, the method of model representation to be used in the new simulation package was defined. In the design of this method, it was proposed that different physical modelling domains would be separated by requiring each pin on a component to model a single physical domain and allowing a pin to be connected to another pin of the same type only. This section describes in detail how the properties of each of these physical pin types will be modelled in the package.

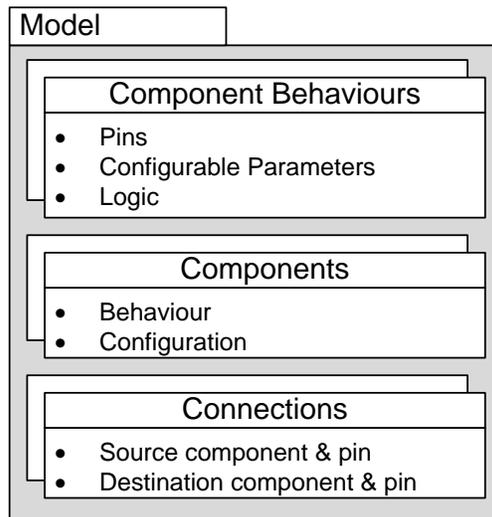


Figure 3-1: Internal representation of a system model within the software package.

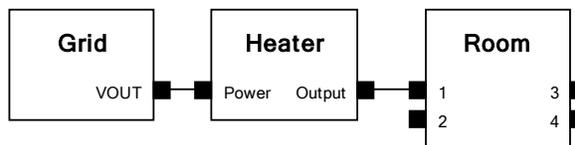


Figure 3-2: Graphical representation of a sample system within the simulation package.

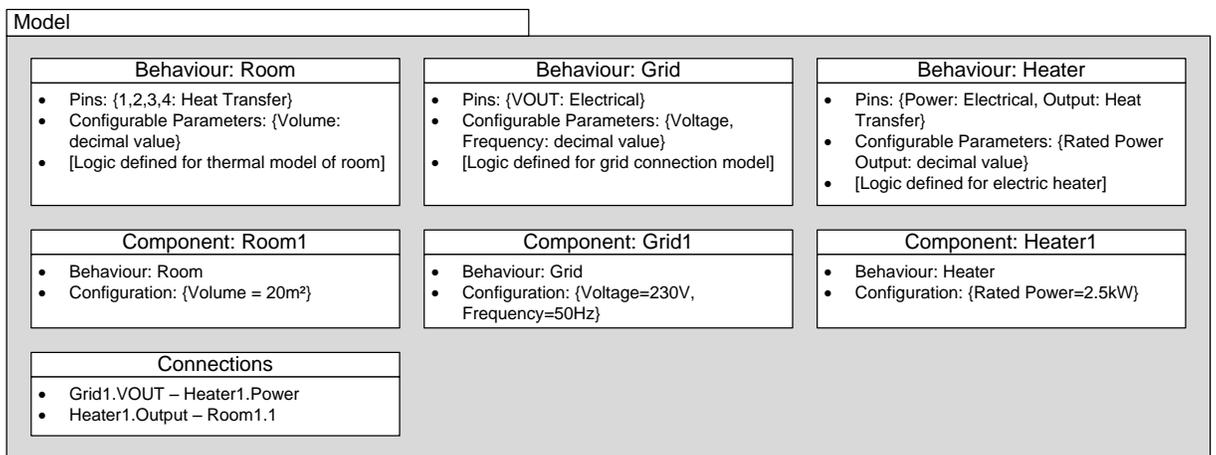


Figure 3-3: Software representation of the graphical model shown in Figure 3-2.

3.3.1. Electrical

The electrical energy simulation element within the package will be required to represent existing AC power systems within buildings, as well as representing new microgeneration and storage systems within buildings which are primarily DC systems. There are also two different temporal resolution constraints imposed on simulation systems when analysing the electrical components of new smart energy systems. When designing switched mode control devices for local storage and microgeneration systems it may be desirable to examine the transient performance of these devices at the sub-second level. When simulating the overall energy use of appliances in the home, the transient behaviour of AC systems becomes less important than the overall power flow from sources to loads.

The EnergyPlus package employs an electrical modelling system [58] where a central component known as the “Load Centre Distribution Manager” controls the overall energy flow within a building. Loads within the building are expressed in terms of their power consumption. Generators are expressed in terms of their total generation capability (which in the case of some generators such as solar may vary depending on external factors such as weather). Storage devices are expressed in terms of their maximum charge and discharge power ratings. The load centre distribution manager within the model controls local generators and storage to supply the local loads, with reference to a given energy management scheme. These schemes may be to minimise or maximise local generation, charge or discharge local storage or to meet certain targets on power imported from or exported to the utility provider. Any shortfall in local generation and storage capacity is met by importing power from the utility provider and any excess is exported. This electrical modelling system has shortcomings in what is required for a domestic smart grid simulation package: it is essentially a lumped power flow calculation for a single building with a utility provider and does not provide any detail into the power flows within the building or the transient effects associated with electronic control systems.

The ESP-r package provides a more detailed electrical simulation capability [88] than EnergyPlus where an electrical model is represented by a set of connected nodes. The model is solved using Kirchhoff's current law where simultaneous equations for the voltage at each

node are expressed in terms of the currents flowing in to or out of the node. To make the solver more suited to building energy systems where sources and loads are usually expressed in terms of power production and consumption rather than voltage and current, a manipulation is made on the standard Kirchhoff equations to express the nodal equations in terms of real and reactive power in and out of each node. This approach is suitable for steady-state analysis of AC systems and can also be applied to DC systems by disregarding the reactive power component. A disadvantage to this approach is that by expressing the network in terms of power flows, large sets of simultaneous equations must be solved which can become computationally intensive for larger systems.

Solution complexity for large sets of simultaneous equations is reduced in the new simulation package by imposing the restriction on the main simulation engine of only being able to solve electrical systems on a component by component basis. The solution chosen to represent electrical systems was to represent every electrical pin on a component as a Thévenin equivalent model. For AC pins this is a voltage source specified in terms of voltage (V), frequency (Hz) and phase angle ($^{\circ}$) in series with an impedance specified in terms of magnitude (Ω) and phase angle ($^{\circ}$). For DC pins this is a voltage source specified by voltage only, in series with a resistance (Ω). All voltages are assumed to be referred to a common 0V virtual ground point so that electrical systems only require a positive or live connection. The negative or neutral paths are inferred as returning to a common point. Figure 3-4 illustrates the equivalent circuits that are used to represent AC and DC electrical pins on components.

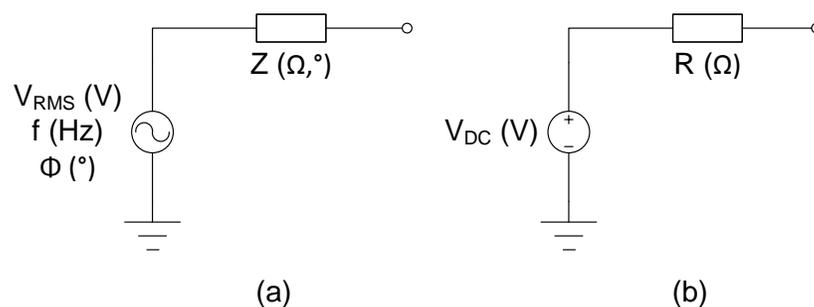


Figure 3-4: Illustration of the way in which (a) AC and (b) DC electrical pins are modelled on components within the simulation package.

In simplifying the electrical domain implementation by modelling electrical systems as component to component connections only, rather than as an entire electrical network, the simulator is potentially providing users with the difficult task of having to create intermediate “infrastructure” components. An example of this would be in connecting more than one load to a single generation source – an intermediate component would be required to provide the extra connections. In order to mitigate this potential pitfall, a built in “electrical node” component was included in the simulator. This component is a fixed-voltage node to which multiple electrical connections can be made. The Thévenin equivalent model of one pin on the node is the combined Thévenin equivalent of the devices connected to every other pin on the node. The component also has the special behaviour of being able to calculate the node voltage for use in the package’s result recording system which is described in section 3.5.1.

Figure 3-5 illustrates a number of electrical components connected to an electrical node. To simplify the implementation of electrical nodes, the design decision was taken to allow only electrical connections of the same frequency to a single node – a mixture of AC and DC sources or different frequencies of AC sources is not permitted at present. The voltage at a node V_{NODE} (V) can be calculated using (1), where V_X is the Thévenin equivalent voltage connected to pin X on the node, Z_X is the Thévenin equivalent impedance connected to pin X on the node and N is the number of pins on the node. V_{NODE} , V_X and Z_X are complex numbers expressed in terms of magnitude and phase angle.

$$V_{NODE} = \frac{\sum_{x=1}^N \frac{V_x}{Z_x} \prod_{y=1}^N Z_y}{\sum_{x=1}^N \frac{1}{Z_x} \prod_{y=1}^N Z_y} \quad (1)$$

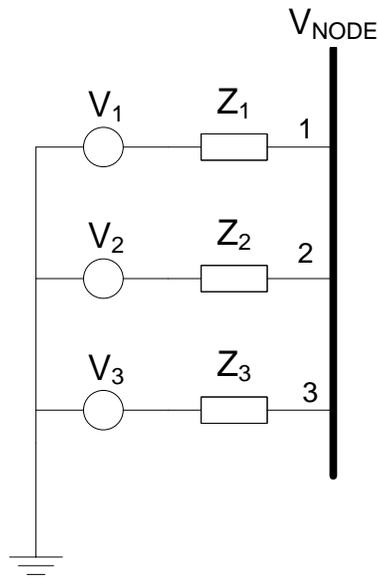


Figure 3-5: Equivalent schematic of an electrical node component.

The electrical model for a given pin on a node is the Thévenin equivalent combination of the electrical models connected to all of the other pins on the component. For a given pin A, the Thévenin voltage $V_{TH,A}$ is calculated using (2) and the Thévenin Impedance $Z_{TH,A}$ is calculated using (3).

$$V_{TH,A} = \frac{\left(\sum_{x=1}^N \frac{V_x}{Z_x} \prod_{y=1}^N Z_y\right) - \frac{V_A}{Z_A} \prod_{y=1}^N Z_y}{\left(\sum_{x=1}^N \frac{1}{Z_x} \prod_{y=1}^N Z_y\right) - \frac{1}{Z_A} \prod_{y=1}^N Z_y} \quad (2)$$

$$Z_{TH,A} = \left(\left(\sum_{x=1}^N Z_x^{-1} \right) - Z_A^{-1} \right)^{-1} \quad (3)$$

3.3.2. Thermal

The thermal simulation capabilities within the package will be required to simulate the transfer of heat between two elements within a simulation. This may be, for example, the transfer of heat between spaces within a building, the transfer of heat from a heating appliance to a space, the loss of heat from a space due to drafts or ventilation, or heat gain due to sunlight through windows.

Existing simulation packages which were discussed in section 2.2 varying in their thermal modelling capabilities, from simple 1D heat conduction models to advanced Computational Fluid Dynamics models, capable of modelling variation in air temperature across a room. The new simulation package is being targeted at smart grid system modelling rather than attempting to accurately re-create any of the existing simulation packages and it was felt that a basic model was suitable to model the heat transfer between spaces in a building. Using a one-dimensional model also allows heat transfer between two components in a simulation to be dealt with as a scalar rather than a vector quantity, simplifying the calculations that need to be performed.

ISO Standard 12831-2003 [89] which specifies a method for calculating the design heat load for a building when installing heating systems was chosen as a basis for developing the heat transfer modelling capabilities of the package. It uses a one-dimensional linear model for heat transfer between spaces within a building and its exterior. It also provides methods to simplify effects such as ventilation through infiltration and forced ventilation to their equivalent linear models.

The heat transfer calculations presented in ISO 12831-2003 are based on (6) below where Q is the heat transfer or heat loss (W), H is the heat transfer coefficient of a material (W/K or W/°C) and $T1$ and $T2$ are the temperatures (K or °C) on each side of the material.

$$\dot{Q} = H(T2 - T1) \quad (4)$$

Equation (4) utilises the heat transfer coefficient of a material which is analogous to electrical conductance. To improve consistency between the electrical and thermal models used within the new modelling package, the thermal resistance of materials will be used instead, resulting in a model of the form shown in (5), where θ is the thermal resistance of a material in K/W or °C/W.

$$\dot{Q} = \frac{(T2 - T1)}{\theta} \quad (5)$$

To allow components within a simulation to conform to this model, the thermal circuit shown in Figure 3-6 is used where each heat transfer pin is specified in terms of a temperature in °C in series with a thermal resistance in °C/W.

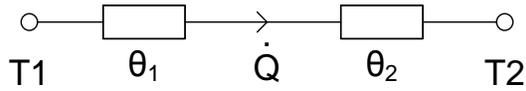


Figure 3-6: Illustration of a thermal connection between two components. Heat transfer pins are expressed in terms of temperature T (°C) and thermal resistance θ (°C/W). The resulting heat transfer from left to right, Q (W), is illustrated.

A special “temperature node” component has been built into the package to assist with thermal modelling. This component has a single pin which represents a user-specified temperature in series with a 0°C/W thermal resistance. The component can also be set to track the global “ambient temperature” setting.

3.3.3. Data Communication

One of the key features of the end-user experience of a smart grid is the concept of communications between the utility company, smart meter, in-home energy displays and any smart appliances. There may also be more complex interactions involved in the system between the user and the various components of the smart energy system. This may involve using a PC (either in the home or externally via the internet) to view smart meter data, configure the smart meter’s ability to remotely control appliances or directly communicate with internet-connected appliances without the involvement of the smart metering infrastructure. Figure 3-7 below illustrates these different types of communication channel by grouping them into four categories based on the potential underlying technology that would be used to implement each channel.

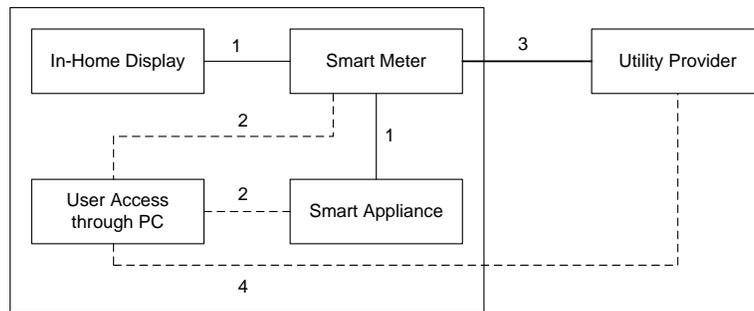


Figure 3-7: Illustration of some of the possible communications channels involved in a smart energy system. 1) A control and status reporting network between the smart meter, in-home display and smart appliances. 2) PC connections to the smart meter or smart appliances. 3) Uplink to the utility provider. 4) User connection to the utility provider for control and monitoring of account.

Current efforts to develop communications specifications for smart energy devices are focussing on the utilisation of standardised communications methods to increase the potential for interoperability between different manufacturers' devices. This is particularly the case in countries with open energy markets such as the UK where consumers may regularly switch energy suppliers. In the UK, the Home Area Network (HAN) which is used to communicate between one or more smart meters in the building and the in-home display is implemented using the ZigBee Smart Energy Profile [39, 90-91]. ZigBee is a wireless communications system which uses the unlicensed 2.4GHz spectrum and is based on the IEEE 802.15.4 personal area network (PAN) standard [92]. The Smart Energy Profile [93] extension provides a standardised framework for transmitting smart meter data over the ZigBee network. It is expected that new firmware for the current generation of smart meters in the UK will make use of the built-in ZigBee hardware and add the Home Automation Profile [94] to allow smart meters to control smart appliances. Similar functionality to ZigBee is provided by the proprietary Z-Wave wireless home automation system [95] to communicate between smart meters, home displays and smart appliances. Another possibility for this technology is to use the IEEE 1901 or HomePlug standard [96-97] which allows fast communication over a short distance of existing power line wiring however the ZigBee or Z-Wave standards would normally be preferred for their lower power consumption.

Numerous technologies also exist for the wide area network (WAN) connection between smart meters and the utility provider. Again, in the UK, interoperability has been defined as one of the key aspects of such a technology. However, unlike the standard specification of ZigBee as the implementation medium for the HAN, no standard technology has yet been proposed for the implementation of the WAN. The main reason for this is that no single communications technology has yet been defined which is capable of serving every smart meter in the country. To allow for interoperability between different systems, it is more likely in this case that a standard interface will be provided to connect to smart meter data once it has reached an IP-based network. Routing systems will connect the different last-mile connections to this central gateway. [90] The current methods used in the UK for connections between smart meters and the utility network include GSM (mobile phone) networks and dedicated long-range wireless networks. The smart meters that have already been rolled out by British Gas and Scottish Power use GSM communication although the British Gas specification [39] states that this can be easily replaced with a module to implement a different form of communication. In recent developments in long-range wireless communication, BT and a number of prominent smart meter manufacturers [98-99] have announced that they will support the Sensus FlexNET [100] wireless communication system – a proprietary system which operates in licensed 890-960MHz spectrum – to provide the backhaul network for remote meter reading and control. This will provide a dedicated network for smart metering which requires fewer base stations than the GSM network to provide UK-wide coverage and is not prone to congestion from voice and data communications in the way that the GSM network is. An alternative wireless communication technique which could also be used for this purpose but has yet to be considered for use in the UK is the IEEE 802.16 WiMAX [101] standard. This standard provides wireless network access in a similar manner to Wi-Fi but over much longer distances. A non-wireless option of power-line communication to the nearest substation has also been discussed in recent literature [102-103].

User interaction within the home energy system may involve scheduling the times that smart appliances may run at, viewing consumption data from the smart meter or remotely switching on or off smart appliances. This type of communication may be achieved locally

from a PC using standard networking protocols such as IEEE 802.3 Ethernet [104] or IEEE 802.11 Wi-Fi [105]. It may also be achieved remotely through the internet in two ways. The end-user may be able to interact with web pages that present historical data collected from the smart meter and may also be able to control appliances through these web pages by using the utility provider's connection to the smart meter. Alternatively, the user may be able to control internet-enabled appliances either directly or through special pages on the manufacturer's website.

This brief survey of the communication channels that are involved at the end-user stage of a smart energy system illustrates that there are a multitude of potential communications topologies and protocols that could be supported within a domestic smart grid simulation package. To reduce the complexity of the design of the communications simulation platform within the package, the different protocols that were discussed – Wi-Fi, Ethernet, WiMAX, GSM, FlexNET, HomePlug, Zigbee and Z-Wave – were analysed under the seven-layer OSI model [106].

When considering each of these methods, it can be noted that while the lower OSI layers of the protocols differ, considering only the transport layer and above allows every protocol to be represented as a stream of bytes being transmitted from one point to another. This is the communications mechanism that will be provided in the simulation package – one component may send a multi-byte communications packet through any of its pins which can then be read by the connected component. The weakness in this approach is that many of the underlying problems with the communications protocols such as transmission latency and corruption or loss of data during communication are not modelled. This, however, can be resolved by implementing intermediate components that introduce time delays or errors into transmissions to simulate these conditions.

3.3.4. Asynchronous Communication

In addition to data communication between electronically controlled systems within a home, it was observed that a more ad-hoc form of communication also occurs as a consequence of the interaction between people and their appliances. In order to represent this separately

from byte-orientated data communications, an asynchronous messaging system was also implemented within the simulation framework. This messaging system would allow one component within a simulation to send a text string to another to indicate that some form of external event (such as a system user activating a physical control) had taken place. Such a message would be handled instantly by the receiving component by executing a dedicated piece of code responsible for processing these messages.

3.3.5. Others

Thus far the design of three physical simulation domains – electrical, heat flow and communication – has been covered. One of the requirements of the software was that physical domains which were not natively supported within the application should be easily added. The obvious approach to the design of such a feature would be to ensure that during implementation of the physical domain modelling capabilities, a sufficient level of abstraction is used to ensure that new domains may be added with relative ease to the application’s source code. While this will be enforced as a design requirement, the design should go further than this and provide a basic method of simulating extra physical domains without having to alter the application’s source code.

This requirement of providing support for extra physical domains where the properties are not natively known by the application will be fulfilled by including a fourth type of pin that may be included on components simply known as an “Other” type pin. In this case, rather than providing a distinct set of configurable properties as in the case of heat transfer or electrical pins, a single configurable property which can contain a scalar value or data structure holding multiple values will be made available.

3.4. Defining & Simulating Component Behaviour

3.4.1. Component Appearance and Configuration

In order to meet the requirement that the application should be extendable and not constrained in the types of system that it can model, it is important that users should be able to define new components when needed to meet a particular system requirement.

The first stage in creating a component would be to define the name of the component and its physical interface to other components within a model. To implement this functionality, the application will provide a user interface to define the component name and a description as well as the pins on the component that will be used to connect it to other components. In line with the physical modelling domains described in section 3.3, the following types will be available for each pin: electrical, heat transfer, communication and “other”. Additionally, each pin will have the ability to be allocated a name – both for use internally for referencing the pin within the simulator, and as a visual aid to the user when making connections.

Another aspect of components that is important to define is the configurable parameters which will allow the customisation of a component without having to recreate new logic each time a slightly different version of the component is used. Examples of the use of such parameters would be the creation of a generic voltage source component with a parameter to specify the required voltage, or the creation of a room model for use in a building with parameters to specify the physical properties of the particular room. This allows for more general component models to be created and re-used in many situations. In accounting for all possible configuration types that may be required for a component, a number of different configurable parameter types should be allowed for components. The types that were eventually selected in the design stage can be grouped into three broad categories: scalar types, vector/matrix types and special types.

The following scalar configuration parameter types can be used on a component:

- **Boolean** – To store true/false values.
- **Decimal** – To store decimal numeric values. This type will be represented internally as a floating point number to improve its performance in calculations over a fixed point representation.
- **Integer** – To store integer numeric values. A 64-bit signed representation will be used to provide a wide range of permitted values.
- **String** – To store textual data.
- **Date / Time** – Building simulations are heavily based on time of day and season, hence it is important to provide a date and time type. A particular option that should

exist within this type is the provision of nanosecond resolution to provide a time type which can be used even when simulating high resolution transient effects.

The following vector or matrix configuration parameter types will be provided:

- **List** – A list is a vector type parameter capable of holding a collection of a particular type of primitive parameter from the types defined above.
- **Table** – A table is used to hold a matrix of data for use by the component. Tables will be constrained to hold a single type of primitive parameter in each column.

Finally, there are additional settings for which it may not be logical to present to the user a primitive, vector or matrix parameter from the types given above, and two special types will also be provided to deal with these situations:

- **Option List** – The option list type will be handled internally in the same way as the “string” primitive type in that it will store a single string value. The main difference to the string parameter is that it will present to the user a drop-down list of pre-set values to choose from rather than providing the option to enter any string. This allows the user to be constrained to choosing a valid option for a given setting.
- **Custom Configuration Method** – For all other options where none of the built-in types are capable of easily representing the setting to be configured, a completely custom configuration option will be provided. This option will internally store the setting value as an array of bytes, allowing the implementer of the component to select the representation to be used for the setting. From the user’s perspective, the implementer of the component will be required to provide a reference to an external plug-in which will display a dialog to the user to collect the required configuration information.

Graphical user interfaces will be provided within the application to edit each provided configurable parameter type (with the exception of the custom type). A separate interface will also be provided to allow the editing of the available parameters on a component to define a name, description, type and default value for each.

3.4.2. Simulation Technique

In section 3.3, the physical domain representations were designed such that each point to point component interconnection is solved independently of the rest of the system, regardless of the physical domain being modelled. The key motivation behind this approach was to simplify the implementation of the main simulation engine so that it is not dependent on having solvers for each of the individual physical domains.

Despite the physical behaviour of a model being simulated entirely by the components themselves, there are still a number of tasks which need to be carried out by a central simulation engine. The first of these is the management of time during a simulation. There are two main objectives to temporal control within a smart-grid simulation. The first is to keep the simulated date and time current so that the behaviour of components can respond to the time of day when required. This is particularly important for weather-related components which may be influenced by season or time of day. The second aspect of time management is to control the granularity and duration of the simulation. These parameters will be configurable by the user to allow long, low-resolution simulations to be run to assess seasonal effects on a model while allowing shorter high-resolution simulations to assess for example the transient effects of switching in electrical systems. The time management component of the main simulation engine will be responsible for advancing the time by the required amount after the evaluation of the model for each time-step and stopping the simulation when the required duration has been simulated. To ensure that both seasonal and transient simulations can be supported in the software, simulation run times from 2ns to 10 years will be specified along with a time-step granularity of 1ns to 1 hour.

The second of the main tasks of the central simulation engine is the management of a global “ambient temperature” reference. This was seen as a useful property to provide globally in the simulation for use when creating building models because it provides a single point of reference for the exterior temperature, rather than having to implement a component to hold the property which must then be connected to multiple points throughout the model. The provision will also be made to allow this value to be changed by components during a

simulation – for example a “weather” component may alter the ambient temperature value over time.

The third main task which should be managed by the central simulation engine is the management of a set of global configuration parameters that may be accessed by all components. These may include data about the weather; settings to control the type of simulation carried out by the model; or may simply be a convenient way to pass data between components. These parameters should be allowed to be of any of the types defined for component parameters in section 3.4.1.

The final task required of the central simulation engine, and the most important, is the running of simulations which involves evaluating the behaviour of each component in a model for each time-step within the simulation. The design of this simulation method is important in the overall design of the package as it will also influence the design of the method used to model component behaviour. The initial stage of designing this method selected a general set of phases that the simulation engine would go through in order to evaluate a simulation which can then be developed into a full method of running simulations. A total of five logical phases were chosen as follows:

- A **Reset** phase will take place at the start of the simulation, resetting the simulated time to the user-selected date and time. All previous simulation results should be cleared and all components reset to their starting state.
- An **Initialise Time-Step** phase will take place at the start of each new time-step within the simulation to increment the time by the user-selected step size and prepare each component for evaluation for the current time-step.
- An **Evaluate** phase will perform an iterative evaluation for the current time-step until a stable solution is reached. This is an important phase because in a model where a component’s output depends on its input, the other components providing the inputs may not necessarily be evaluated before the component in question, so it may be necessary to perform multiple passes of evaluation of the model before all components converge to stable solutions.

- A **Result Recording** phase will take place after a stable solution for each time-step is reached to record the results for the time-step.
- An **End of Simulation** phase will run at the termination of the simulation to perform any final operations that components require in order to save their state and display the simulation results.

The design of the chosen simulation method is depicted formally in the activity diagram in Figure 3-8.

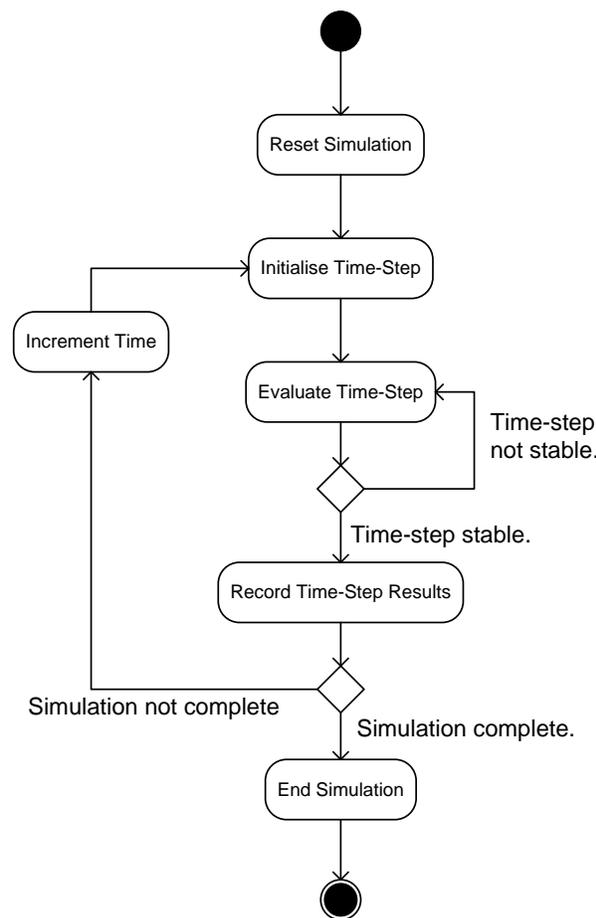


Figure 3-8: Activity diagram illustrating the method used within the package to evaluate a model.

3.4.3. Implementing Component Behaviour

In section 3.4.2, the simulation technique for the package was designed to have five phases, with four specifically related to performing operations on components to evaluate their

behaviour: **Reset, Initialise Time-Step, Evaluate** and **End Simulation**. In order to allow advanced software control of components' behaviours as stated in the original requirements it was decided that a scripting interface mapping directly to these five phases of the simulation technique should be exposed to users of the package in order to implement component behaviour.

Two options are available to design such a scripting interface: implement a custom scripting language for the application, or adapt an existing scripting or programming language to fit the system requirements. The former has the advantage that it allows for the development of a language that exactly meets the requirements of the application but has the disadvantage of longer development time due to the need to write a compiler and syntax guide for the new language. Implementation of a new language may also present problems for users of the package due to the learning curve involved with an entirely new language. The advantage of using an existing language to implement component behaviour is that many users will already be familiar with popular scripting languages and also that compilers are already available for existing languages. The disadvantage to using an existing language is that the features of the language may not map directly onto the requirements of the application and some intermediate-level "glue" code may be required to bind the scripting code onto the underlying simulation engine model.

The decision was taken to begin using a standard scripting language for faster development of the package, rather than spending time implementing a completely new scripting language. The choice of the C# language on the .NET framework 3.5 as the development platform for the application provided a convenient method for the inclusion of a scripting framework within the application. This is because the .NET framework provides the *Microsoft.CodeDom.Compiler* namespace which provides compilation services to application programmers as well as the *Microsoft.CSharp* and *Microsoft.VisualBasic* namespaces which provide language-specific compilers for the C# and VB.NET languages. All .NET compilers generate common intermediate language (IL) code which is then interpreted by the .NET runtime components to perform the actual computation on the target processor. This feature can be exploited to take source code entered by the user at run-time in any .NET

language, compile it and then load it into the running application regardless of the language that the running application was written in. The use of this technique to add run-time scripting to a .NET application is demonstrated in the Microsoft article “Script Happens.NET” [107].

In order to allow a user script to implement the required simulation phases, the abstract class shown in Figure 3-9 was designed to define a set of functions which component scripts can implement to control the behaviour of the component. The default implementations of these functions are empty to allow the user to omit functions which are not required from the component’s script code.

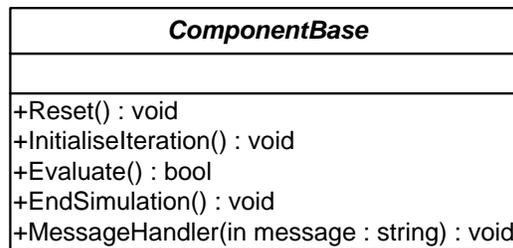


Figure 3-9: Abstract class which provides the methods that component scripts should implement to define component behaviour.

In line with the simulation technique design from section 3.4.2, the *Reset* method should return the component to a known starting state; the *InitialiseIteration* method should prepare the component for evaluation of the current time-step and the *EndSimulation* method should update the component’s parameter values if required at the end of the simulation. The *Evaluate* method implements the iterative evaluation process of the simulation engine and should return a Boolean value – *true* if the component state is stable, or *false* if the component has changed any of its pin or parameter properties during the time-step. The *MessageHandler* method is responsible for implementing the asynchronous communications system described in section 3.3.4. Each time that an asynchronous communication message is sent to a component, the *MessageHandler* function will be called with the message passed as a parameter to allow the component’s code to process the message as required.

In order to allow C# or VB.NET scripting code to control the behaviour of a component, an API will be provided which will be exposed to the scripting code as two variables. One will be a class which provides access to the simulation engine in order to obtain global parameter values, the current time-step value, and various methods to generate error messages and control the running of the simulation. The other will be a class which provides access to the component being controlled by the scripting code to manipulate the component's pins and parameters and determine the properties of other components which are connected to the pins of the component.

During the design of this scripting interface, it became apparent that while the four-method interface using C# and VB.NET was extremely flexible for implementing components with complex behaviour, it was also cumbersome for simpler components which require few computational operations to define their behaviour. Examples of such components are voltage sources or resistors which have a single fixed value that is set at the beginning of a simulation and never changes. In order to simplify the modelling of these components a mathematical mark-up notation was added to the selection of programming languages available to model components. While the language was purpose-built for this particular application, it borrows standard mathematical syntax from the majority of current programming languages and some pre-processor and comment syntax from the C programming language. Table 3-1 illustrates the key features which have been included in this language. The notable features missing from this language that are available in the full programming languages are: support for communications, accessing non-numeric configurable parameters, and storing of state variables between iterations of the solution process.

The language provides a method of mapping configurable parameters and pins from components onto variables within a mathematical script. A number of standard mathematical operators and functions are then provided which can be used to manipulate these values to implement the component behaviour. Once the execution of the mathematical script is complete, any changes to the mapped variable values are copied back to the main simulation engine. Three optimisation options are provided in the language to

specify the way in which mathematical scripts defining component behaviour are evaluated. Table 3-2 illustrates the way in which mapping features of the language work and the activity diagram in Figure 3-10 illustrates the way in which the evaluation of mathematical models maps onto the 4-function evaluation interface used by C# and VB.NET code that is shown in Figure 3-9. Table 3-3 provides an example of the implementation of a configurable DC voltage source in both C# and the mathematical mark-up language.

Table 3-1: Summary of the key features of the mathematical mark-up language that has been included in the simulation package.

Comments	//	Indicates the start of a single line comment. No multiple line comment equivalent is provided.
Operators	+ - * /	Basic mathematical operators: addition, subtraction, multiplication.
	^	Power.
	%	Modulo.
	!	Factorial.
	== !=	Logical equality and inequality.
	< >	Logical less than and greater than.
	<= >=	Logical less than or equal and greater than or equal.
	=	Variable assignment.
Functions	Sin(x), Cos(x), Tan(x)	Trigonometric functions operating in radians.
	ASin(x), ACos(x), ATan(x)	Inverse trigonometric functions returning values in radians.
	Sinh(x), Cosh(x), Tanh(x)	Hyperbolic trigonometric functions.
	Exp(x)	Raises e to the power x .
	Ln(x), Log10(x), Log(x,a)	Natural logarithm of x , logarithm to the base 10 of x or logarithm to the base a of x .
	Max(a,b), Min(a,b)	Minimum or maximum of two values.
	Deg(x), Rad(x)	Converts a value in radians to degrees or a value in degrees to radians.
	Infinity(), NegativeInfinity()	Returns the values infinity or negative infinity.
	Floor(x), Ceiling(x), Round(x)	Rounds values down to the nearest integer, up to the nearest integer or to the nearest integer either way.
	If(x, true_value, false_value)	Logical if statement: returns the true value if the condition x is met or the false value if not.
	Abs(x)	Gets the absolute value of a number.
	Sign(x)	Returns -1 if x is negative, 1 if positive or 0 if 0.
Global Variables	Time	The time in seconds since the beginning of the simulation.
Mapping Keywords	#parameter variable="name"	Maps a configurable parameter of the component called name to the variable called variable within the mathematical script.
	#global variable="name"	Maps a global parameter called name to the variable called variable within the mathematical script.
	#pin variable="name"	Maps the pin called name on the component to a mathematical variable called variable .
Optimisation Keywords	#opt none (Default Option)	Specifies that the component's mathematical mark-up must be evaluated on each pass of the iterative simulation engine.
	#opt time	Specifies that the component's mathematical mark-up must be evaluated only once per time-step.
	#opt single	Specifies that the component's mathematical mark-up only needs to be evaluated once per simulation.

Table 3-2: Illustration of the mapping process used in the mathematical mark-up language.

Mapping Syntax	Variables Made Available in Mathematical Script	Description of Variables
#pin Name="Pin 1" <i>Where "Pin 1" is an electrical pin.</i>	Name.vmag	Voltage magnitude (V) of the pin.
	Name.vrms	RMS voltage (V) of the pin (equal to magnitude if DC).
	Name.phase	Phase angle (°) of the pin if AC.
	Name.freq	Frequency of the pin (Hz) if AC.
	Name.dc	1 if the pin is DC or 0 if AC.
	Name.zmag	Magnitude of series impedance (Ω).
	Name.zphase	Phase angle of series impedance (°).
	Name.connected.vmag	Properties of the pin that is connected to the named pin (read-only variables).
	Name.connected.vrms	
	Name.connected.phase	
	Name.connected.freq	
	Name.connected.dc	
Name.connected.zmag		
#pin Name="Pin 2" <i>Where "Pin 2" is a heat transfer pin.</i>	Name.temperature	Pin temperature (°C).
	Name.resistance	Pin thermal resistance (°C/W).
	Name.connected.temperature	Properties of the pin that is connected to the named pin (read-only).
	Name.connected.resistance	
#parameter a="Parameter 1"	a	Parameter containing the value of "Parameter 1".
#global g="Global 1"	g	Parameter containing the value of the global parameter "Global 1".

Table 3-3: Illustration comparing the implementation of a configurable voltage source in C# to the implementation in the mathematical mark-up language.

C# Code	Mathematical Mark-Up Code
<pre>public override void Reset() { double v = Component.Parameters["Voltage"]; ElectricalPin Ep = Component.Pins["OUT"]; Ep.DC = true; Ep.Voltage = v; Ep.LoadMagnitude = 0; Ep.LoadPhase = 0; }</pre>	<pre>#opt single #parameter v="Voltage" #pin Ep="OUT" Ep.dc=0 Ep.vmag=v Ep.zmag=0 Ep.zphase=0</pre>

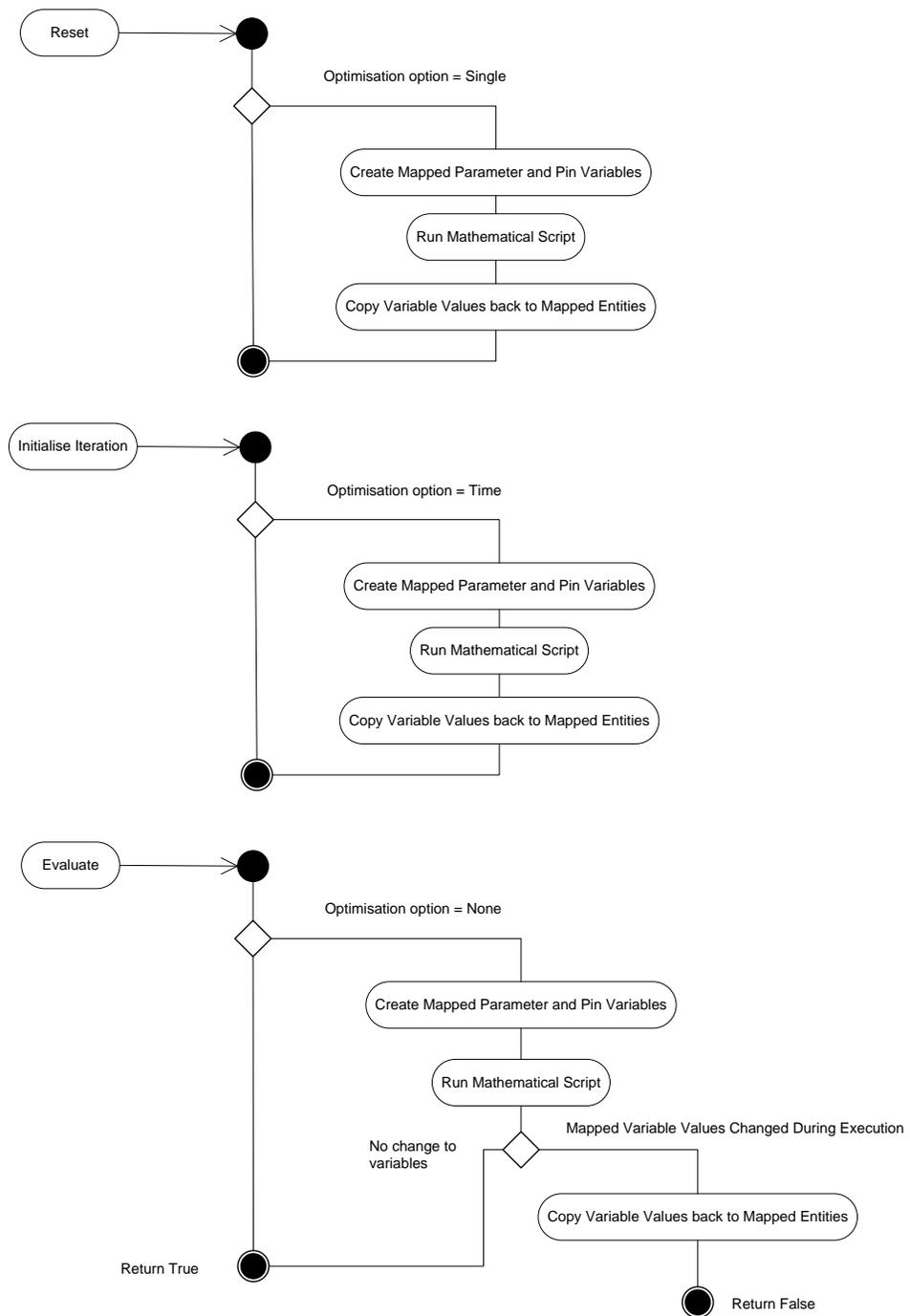


Figure 3-10: Activity diagram illustrating how the process of evaluating mathematical scripts representing component behaviour maps onto the 4-function interface used by C# or VB.NET scripts. Note that the “End” function is not included because it is not used when evaluating mathematical scripts.

3.5. Simulation Results

3.5.1. Recording Results

When simulating across multiple physical domains, there is a potentially very large number of results that could be obtained. In the electrical elements of a simulation the important values would be voltage; current; phase angle in AC systems; power flow (real, reactive and apparent in AC systems); and energy use in terms of both kWh used and cost. In the thermal elements of the simulation, the important values would be heat flow and temperature. In the communication elements, data transfer between systems would be recorded.

Considering the number of different parameters that could be recorded in a simulation and considering that even in the simplest models there may be a large number of connected components, it becomes clear that recording every possible value for each time-step of a simulation would begin to put a strain on system memory and disk resources. It is therefore imperative that the design of the results recording mechanism considers only the relevant results for the particular study being carried out and that the resolution of result recording should be independent of the temporal resolution of the simulation. Examples of the memory requirements of the package for recording results at different temporal resolutions are provided in Table 3-4.

Table 3-4: Illustration of the memory requirements for recording numeric simulation results at different temporal resolutions. This table makes the assumption that 10 different result values are collected from the model on each iteration and that these results are floating-point values.

Result Recording Resolution	Memory Requirement per Second of Simulated Time	Total Memory Requirement for a Simulation of 1 Hour Simulated Time
1ns	3.2 GB	11.25 TB
1s	320 bytes	1.09MB
1min	5.3 bytes	18 KB
1hour	0.08 bytes	288 bytes

Taking these requirements into account, the result recording mechanism was designed on the principle that users should be able to instruct the package to “watch” certain properties

within a model with the values of these properties recorded in memory during the result recording phase at the end of each time-step, which is depicted in Figure 3-8. To reduce the volume of results generated for analysis, the user will be provided with the option to record results at a lower resolution than the simulation is evaluated at.

The user interface for recording results will be designed so that the user can select a component, electrical node or connection to obtain results from. Options will then be presented for the type of results that can be obtained from the selected object – these will depend on the physical domains that are represented by the selected object. In addition to physical values from a model, the option will also be included to take snapshots of both component and global configurable parameter values during a simulation. This will allow components such as controllers to report their state by altering parameter values during the course of a simulation.

The object-orientated software representation shown in the UML class diagram in Figure 3-11 was designed for the result recording capabilities within the software. The top-most interface *ISimulationResultGenerator* provides a generic interface for an entity that can record result values from a simulation. This provides a property containing a description of the result generator, an array of results generated and methods to perform result-gathering operations. These methods clear stored results at the beginning of a new simulation, allocate memory to store the results for a simulation and record results for a given time-step in the simulation. Pre-allocation of memory was selected over dynamic allocation of memory during a simulation so that any memory shortages could be identified before beginning to run a simulation rather than the program being unable to allocate memory after a considerable length of simulation time. The *SimulationResultValue* class is used to store each individual result value and contains properties to store numeric (double-precision floating point), string or binary (byte array) values. This covers all possible value types that can be generated by physical domain models, parameters and data communication.

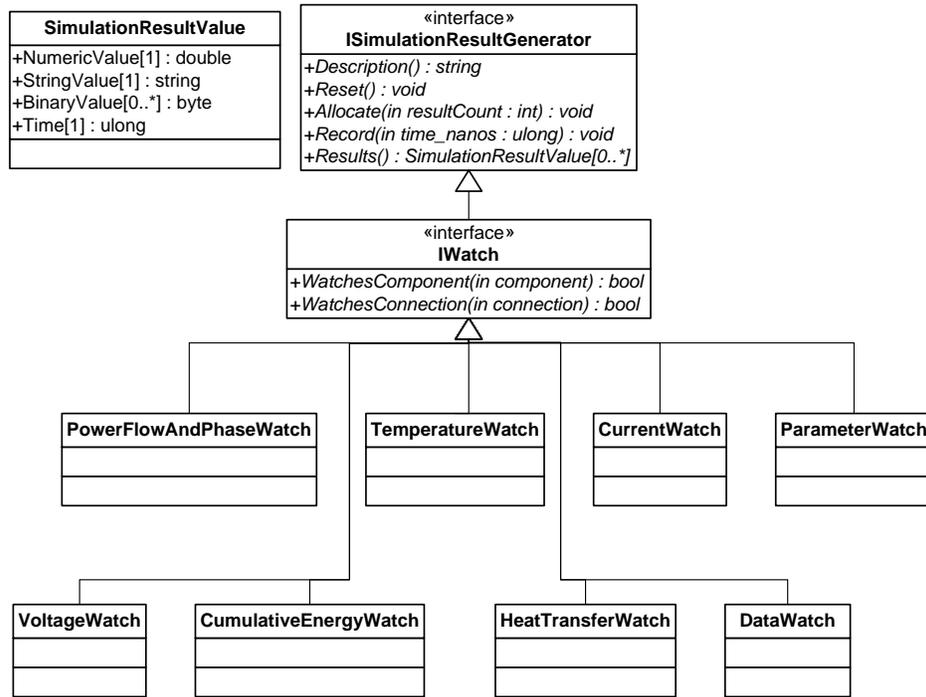


Figure 3-11: UML diagram illustrating the design of the watch system used to collect results from the simulation.

A more specific interface for recording physical properties from the model is specified in the form of the *IWatch* interface. This defines additional methods which determine if the watch records values from a specific component or connection. These methods are provided to allow watches to be removed from the model if the associated component or connection is removed. This interface is then implemented by individual watch classes which record results from a model: Voltage; Current; Power Flow, Phase Angle and Power Factor; Cumulative Energy Flow and Cost; Heat Transfer; Temperature; Data Flow; Parameter Value. The remainder of this chapter describes the methods used to record each of these values.

The Voltage, Temperature and Parameter Value watch classes contain the simplest design in the package – these values are directly extracted from the model. Section 3.3.1 described how the design of electrical nodes involves the computation of node voltage. Voltage watches placed on a node read this voltage value to record their results. If the instantaneous rather than RMS voltage is requested for AC systems, it is obtained using equation (6) where V_{RMS} is the RMS voltage computed by the node, f is the AC voltage frequency at the node, φ

is the phase angle of the voltage relative to a reference phase angle of 0° and t is the time in seconds since the beginning of the simulation.

$$V(t) = \sqrt{2}V_{RMS} \cos(2\pi ft + \varphi^\circ) \quad (6)$$

Section 3.3.2 described how components could be set to support the concept of having temperature (for example a component representing a room within a building). Temperature watches extract the currently set temperature from components which support this. Parameter watches behave similarly by taking a snapshot of either a component parameter or global parameter when requested to record a result.

Electrical current, power flow and phase angle values are not directly available within models and therefore must be calculated when required by the associated watch objects. To illustrate how these values are calculated, we will consider the electrical connection of two Thévenin equivalent circuits shown in Figure 3-12.

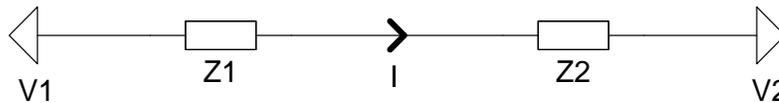


Figure 3-12: Illustration of an electrical connection between two components showing complex voltages V1 & V2, complex impedances Z1 and Z2 and complex current I.

In this system the voltages V1 and V2 and impedances Z1 and Z2 are complex values. For RMS AC values and DC values, the same calculations can be used under the assumption that the phase angle of the values will be set to zero for DC systems. The Ohm's law calculation in (7) is used to obtain DC or RMS current where the variable I is a complex number with magnitude I_{RMS} (A) and phase angle φ ($^\circ$). Instantaneous AC current can be obtained using (8) in a similar manner to the instantaneous voltage where f is the voltage frequency on the link (Hz)⁶ and t is the time in seconds since the beginning of the simulation.

⁶ As described in section 3.3.1, the simulator places the restriction on models that all voltages in an electrical network must have the same frequency.

$$I = \frac{V1 - V2}{Z1 + Z2} \quad (7)$$

$$I(t) = \sqrt{2}I_{RMS} \cos(2\pi ft + \varphi^\circ) \quad (8)$$

The phase angle of an AC system is the phase of the current relative to the voltage. As the simulator refers all currents to a reference phase angle of 0°, the phase angle of a particular link φ_{AC} (°) can be calculated using (9) below where φ_V is the voltage phase angle and φ_I is the current phase angle.

$$\varphi_{AC} = \varphi_I - \varphi_V \quad (9)$$

AC Power flow values are computed using the magnitude of the complex current value obtained in (7) along with the magnitudes of the voltages V1 and V2 from Figure 3-12 and the phase angle φ_{AC} . Real power P (W), reactive power Q (VAr) and apparent power S (VA) are calculated using equations (10), (11) and (12) respectively.

$$P = Re(I) \cdot (Re(V1) - Re(V2)) \cdot \cos(\varphi_{AC}) \quad (10)$$

$$Q = Re(I) \cdot (Re(V1) - Re(V2)) \cdot \sin(\varphi_{AC}) \quad (11)$$

$$S = (Re(V1) - Re(V2)) * Re(I) \quad (12)$$

The AC power factor can be calculated from the phase angle φ_{AC} using equation (13).

$$PF = \cos \varphi_{AC} \quad (13)$$

For DC systems, the DC power P_{DC} (W) can be calculated using the magnitudes of the voltages V1 and V2 and the current I as shown in equation (14).

$$P_{DC} = |I| \cdot (|V1| - |V2|) \quad (14)$$

The final electrical watch included within the package will be the cumulative energy flow watch. This watch will calculate the net import or export of energy over the course of a whole simulation along a connection with imports indicated by a positive value and exports by a negative value. To calculate the cumulative flow of energy, the real power P or P_{DC} value – depending whether the connection is part of an AC or DC system – is used. As energy is the

integral of power flow with respect to time, a trapezium method approximation is used to compute the energy flow between the previous simulation time-step and the current time-step $E_{TIMESTEP}$ (J). This relies on the knowledge of the present power flow P_{NOW} (W), previous time-step power flow P_{PREV} (W) and the simulation time-step size T as in (15).

$$E_{TIMESTEP} = T \cdot \frac{P_{NOW} + P_{PREV}}{2} \quad (15)$$

Once the energy use in the last time-step has been computed, it can be used in three ways depending on the type of energy metric required. If the energy use in joules is required directly, $E_{TIMESTEP}$ is simply added to a running energy use total. If the energy use in kWh is required then $E_{TIMESTEP}$ is divided by 3.6×10^6 to convert joules to kWh and then added to a running total. If energy cost is required then $E_{TIMESTEP}$ is divided to 3.6×10^6 to convert to kWh, then multiplied by the user specified price per kWh before being added to the running price total.

In section 3.3.2 it was proposed that heat transfer connections should use a series temperature and thermal resistance circuit to make their definition analogous to the electrical simulation features within the package. This means that calculation of heat transfer between two points is very similar to the calculation of electrical current. Given the thermal link illustrated in Figure 3-13 with the temperatures $T1$ and $T2$ ($^{\circ}\text{C}$) and thermal resistances θ_1 and θ_2 ($^{\circ}\text{C}/\text{W}$), the heat flow Q (W) can be calculated using (16).

$$\dot{Q} = \frac{T1 - T2}{\theta_1 + \theta_2} \quad (16)$$

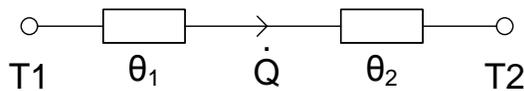


Figure 3-13: Illustration of a heat transfer connection between two components showing temperatures $T1$ & $T2$ ($^{\circ}\text{C}$), thermal resistances θ_1 and θ_2 ($^{\circ}\text{C}/\text{W}$) and heat flow Q (W).

In addition to the transfer of heat between two components, it is also desirable at times to measure the temperature at points within a model. As described in section 3.3.2, ambient and absolute temperature points within a model are represented by special “temperature node” components. The option is also provided to allow components, such as those representing heated spaces, to have a temperature. A watch object is provided which can read this temperature value from temperature nodes and also other components which indicate that they provide temperature information.

In section 3.3.3, it was indicated that data communication between two components is modelled as a stream of bytes being transferred from one component to another. The initial design for this communication process involved having an input queue for every data pin on a component. When a component’s implementation performed a data transmission operation to another component, the transferred data was written directly to the input queue of the connected component. This process proved to be problematic for observing the data transmission between components so it was redesigned such that an intermediate “communications processor” was added to the main simulation engine to assist with the recording of transmitted data. A data watch object is provided to record the data transmitted along a connection. When a data watch is created, it registers itself with the intermediate communications processor entity so that any transmitted over the watched connection is sent to its destination component and in addition is also recorded by the watch object. Data is recorded in raw binary form and can be processed by a result viewer into decimal, hexadecimal or ASCII for viewing. The process of sending data from one component to another while it is being recorded by a data watch is illustrated in the sequence diagram in Figure 3-14.

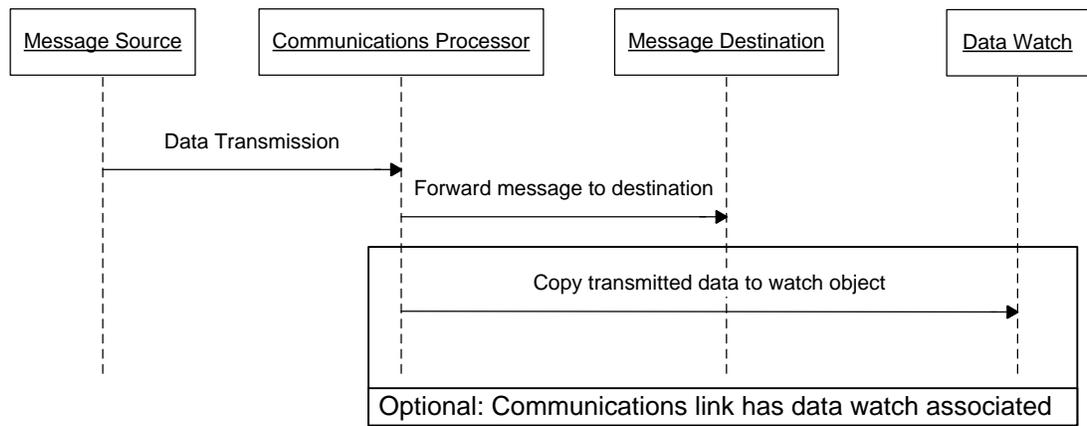


Figure 3-14: Sequence diagram illustrating the processing of a communications message. The optional block at the end of the sequence of events indicates the way in which a message is intercepted for forwarding to an associated Data Watch object.

3.5.2. Mathematical Operations on Results

Often when analysing data from simulations it is desirable to perform some form of mathematical operation on one or more results to generate further useful data. In light of this, a limited number of mathematical operators are built into the package and should generate results in the same way as watch objects. The main difference is that mathematical operator values for a time-step will be computed using watch results for that time-step and will therefore be evaluated after the successful evaluation of watch data. The UML diagram in Figure 3-15 illustrates the *IMathOperator* interface which extends the *ISimulationResultGenerator* interface in the same way as *IWatch* to provide additional functionality. Mathematical operators provide the same interface as watches so that the Simulation Engine may record results using them in the same way, but add an extra method to determine if a mathematical operator depends on a specific watch. This allows the operator to be removed from the model in the case where one of the watches that it depends on is removed.

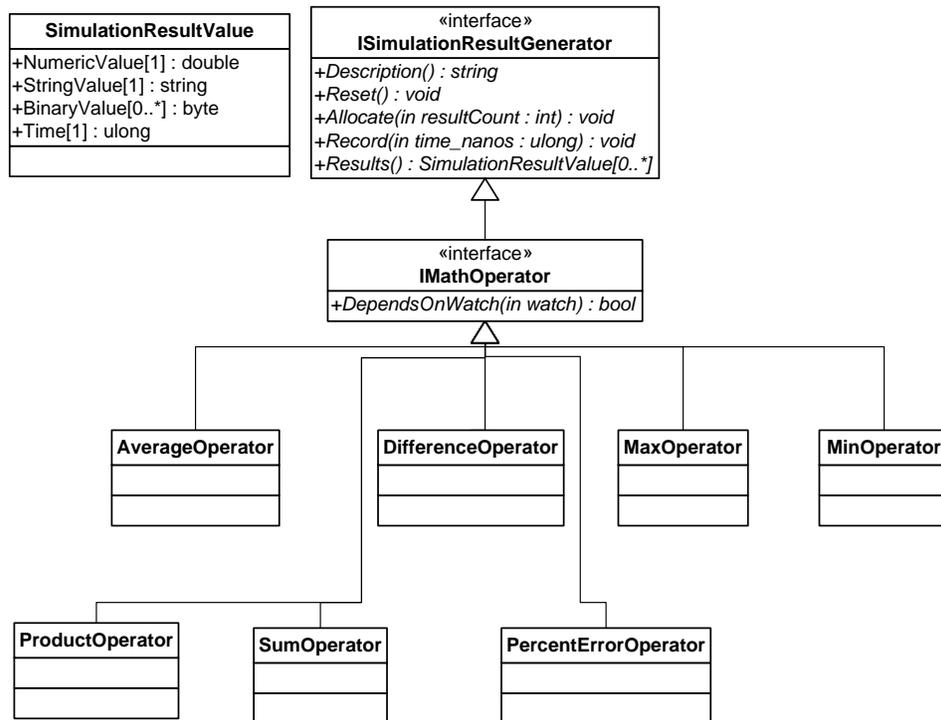


Figure 3-15: UML diagram illustrating mathematical operator system within the package.

The Average, Maximum, Minimum, Product and Sum operators operate on two or more watches to obtain their values. The Difference operator computes the difference between two watch values. The Percentage Error operator computes the percentage error in a given actual value when the expected value is also available within the model (for example, a controller output based on a given reference signal). The value for a percentage operator is computed using (17).

$$\%Error = \frac{Abs(expected - actual)}{Abs(expected)} \cdot 100\% \quad (17)$$

3.5.3. Analysing Results

Two options were identified for the analysis of the results generated by the simulation package. The first was to rely on external packages to analyse results. This could be achieved by either using the API provided by a third party data analysis package – for example Microsoft Excel – to export data and generate the required graphs and tables upon the

completion of simulations, or more simply by saving simulation data to a file format which is supported by such tools.

The second option was to include data analysis tools within the simulation package itself, removing the requirement for any third party software to be installed on the system. This would require the implementation of various visualisation tools, adding to development time, but provides the benefit of providing a more streamlined process for running simulations. In this scenario, settings within a model can be changed, the simulation run and the built-in result viewers checked to see if the model has achieved the required output. The main disadvantage to this type of approach would be that the analysis features may not be as comprehensive as those provided in a dedicated package.

The final design for the results analysis features incorporates the best parts of both of these approaches. Three simple visualisation tools are provided within the package itself to allow for more rapid development and verification of models, with the ability to also export the results in a CSV spreadsheet format for compatibility with most off-the-shelf data analysis packages, thereby providing the capability for more advanced analysis. The three built-in analysis features that were chosen were: tables to view numerical values or communication link data values; line graphs to view numerical results over time; and pie charts to show the contribution of a number of variables to a total value. Line graphs were selected because they clearly illustrate how the value of a particular variable (for example, electrical power consumption) varies over the course of a simulation. Pie charts were selected because they illustrate the contribution that various values (for example, the sources of heat loss from a building) make to a total value (for example, the total heat loss from a building) at a given instant in time.

3.6. Persistence of Models

In order to be able to re-use components between different models, as well as to store and re-open models themselves at a later date, there needs to be some form of storage system built into the application. In assessing the storage requirements for the application, the storage of libraries of components and the storage of system models were identified as two

separate actions that users may wish to perform. Separating the storage in this way allows libraries of components to be re-used between different simulations and updated centrally, rather than having different versions of a component throughout different model files.

In order to save a component library or a model to a file, a transformation of the application's internal object-orientated representation into a representation that can be saved to a file is required. The XML format [108] was chosen for this purpose because it is particularly suited to storing collections of data in a hierarchical fashion. XML has the benefit of being human-readable which is useful for debugging purposes and reading and writing of XML files is natively supported by the .NET framework which simplifies implementation of this file format. The XML format does have the disadvantage of using more storage space than binary file formats due to its human-readable nature – especially if large amounts of whitespace are used to display the hierarchical structure of the document. This larger space requirement can however be mitigated by compressing the XML data before storing to disk, for example in the ZIP or GZIP formats which are also natively supported in the .NET framework.

When designing the file formats for storing both component libraries and models, some important clarifications about the exact method of storage had to be made. These were: what to do if a model is used on a workstation other than the one it was designed on which does not have the required component library files; what to do if a model uses a component for which the library is subsequently upgraded; how to make changes to the file format to incorporate new features without making the application unable to load older files; how to store information about the type of simulation required and which results to display; and how to uniquely identify elements of a model and the relationships between them when saving to files.

To mitigate the issue of simulating models on different workstations which may or may not have the required libraries, and to handle upgrades to component libraries, a version number is included within the definition of each component. Additionally, when saving a model to a file, the model file will contain cached copies of all of the components used within the model to ensure that the model can be used on any system, including those which may have a

different version of the component in a library file. In the case where a component library has been updated since a model was created, the provision is made in the application to update affected components to the latest version.

As the development of any software package evolves it is likely that new features will be added or existing features may be changed. To reflect this, changes may be made to the storage format. To avoid having to re-create model files each time features within the software are changed, it is essential that the software should be backward compatible with files created in an older version. To allow for this, a version number will be included in each file created by the application. For each new file version that is defined, a set of rules will be created which instruct the application how to modify the XML content of a file to upgrade from the previous file format to the current file format. A per-version upgrade system like this provides the advantage that it allows any previous file format to be used by the application by incrementally upgrading the file structure until it is at the latest format. It also simplifies the logic which converts the internal representation of system models and components to their XML counterparts and vice versa since this logic will always operate on the latest file version.

To store the simulation properties which were introduced in section 3.4.2 – start time, duration, time-step size, global parameters and ambient temperature – it was decided that an extra section known as a “Simulation Environment” would be added to the definition of a model. As well as these simulation-engine-related properties, the result generator objects discussed in section 3.5.1 and information about the types of viewer (graph, pie chart or table) selected to view these results would also be included under this section. The advantage of taking this approach to storing simulation properties is that multiple different types of simulation may be defined for each model, from short transient simulations to longer seasonal simulations, and the appropriate results generated for each simulation.

There is a relationship between an instance of a component used within a model and the underlying component behaviour. Internally within the application, this is represented by an object reference within the component class to the associated behaviour class. When models are saved to a file, this relationship must also be preserved and therefore a method is

required to uniquely identify a component's behaviour so that it can be referred to in within the XML representation of a particular component instance. The chosen solution to this problem was that in addition to a descriptive name, each component behaviour would be assigned a GUID (Globally Unique ID). GUIDs are 128-bit integers generated by an algorithm built into Windows which is designed to produce numbers based on the current time that are statistically unlikely to be repeated and can therefore be considered "globally unique". GUIDs were chosen as the preferred unique identifier because they allow component libraries created on different computers to be used on other systems with very little risk of ID clashes between components. The XML definition of a component instance will contain a field specifying the GUID of the associated behaviour.

Similarly, a component link is represented internally by object references to the start and end pins that the link connects. These must also be converted to a textual representation. The chosen textual representation in this case is to store the name of a component instance and pin that represents each end of the link. Pin names can be used directly as long as the restriction of no duplicate pin names on components is enforced. For this representation to work, component instances will also need to be uniquely named within a model. Again, a GUID could be used for this purpose but it was felt that because the scope for duplicate names was limited to an individual model file and was not global, user-defined names would be used to represent individual components within a simulation, with the package initially providing a default name based on the associated component behaviour name (for example, "Building_1").

Figure 3-16 and Figure 3-17 illustrate the hierarchical structure of the XML storage of component libraries and system models. For clarity, component behaviour and component instance elements from the XML files have been shown separately in Figure 3-18 and Figure 3-19 respectively. Component library files will contain only a list of component behaviours to be used within new models while system model files will contain a system model definition (including the required component behaviours) as described in section 3.2.2 and illustrated in Figure 3-1. Additionally, a system model file will make the provision for including illustrative

line, rectangle and textbox elements which can be used to annotate models for the benefit of users viewing the models.

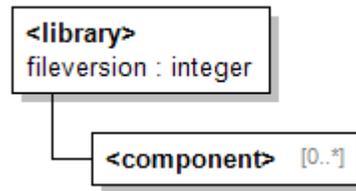


Figure 3-16: XML schema for a component library file.

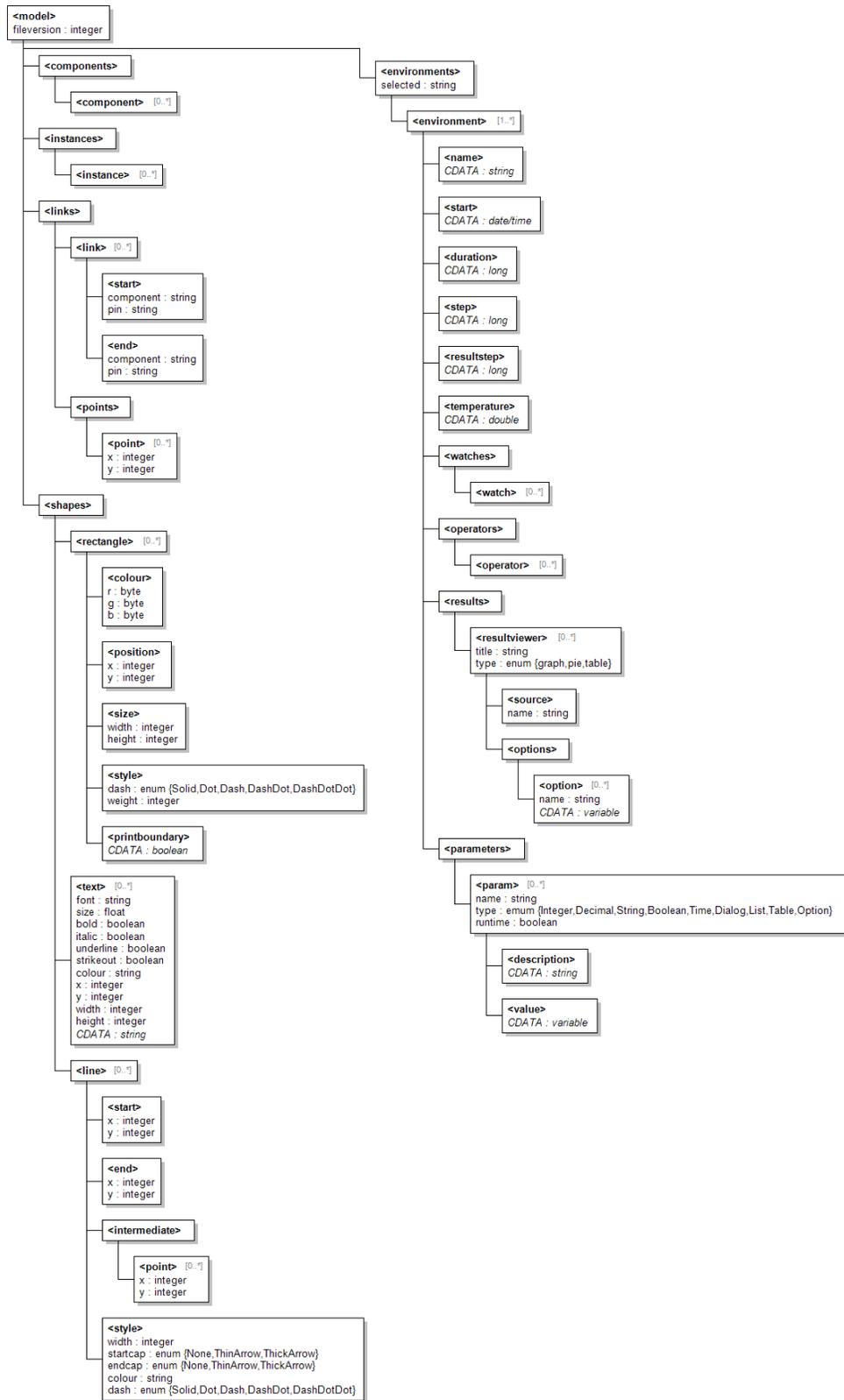


Figure 3-17: XML schema for a system model file.

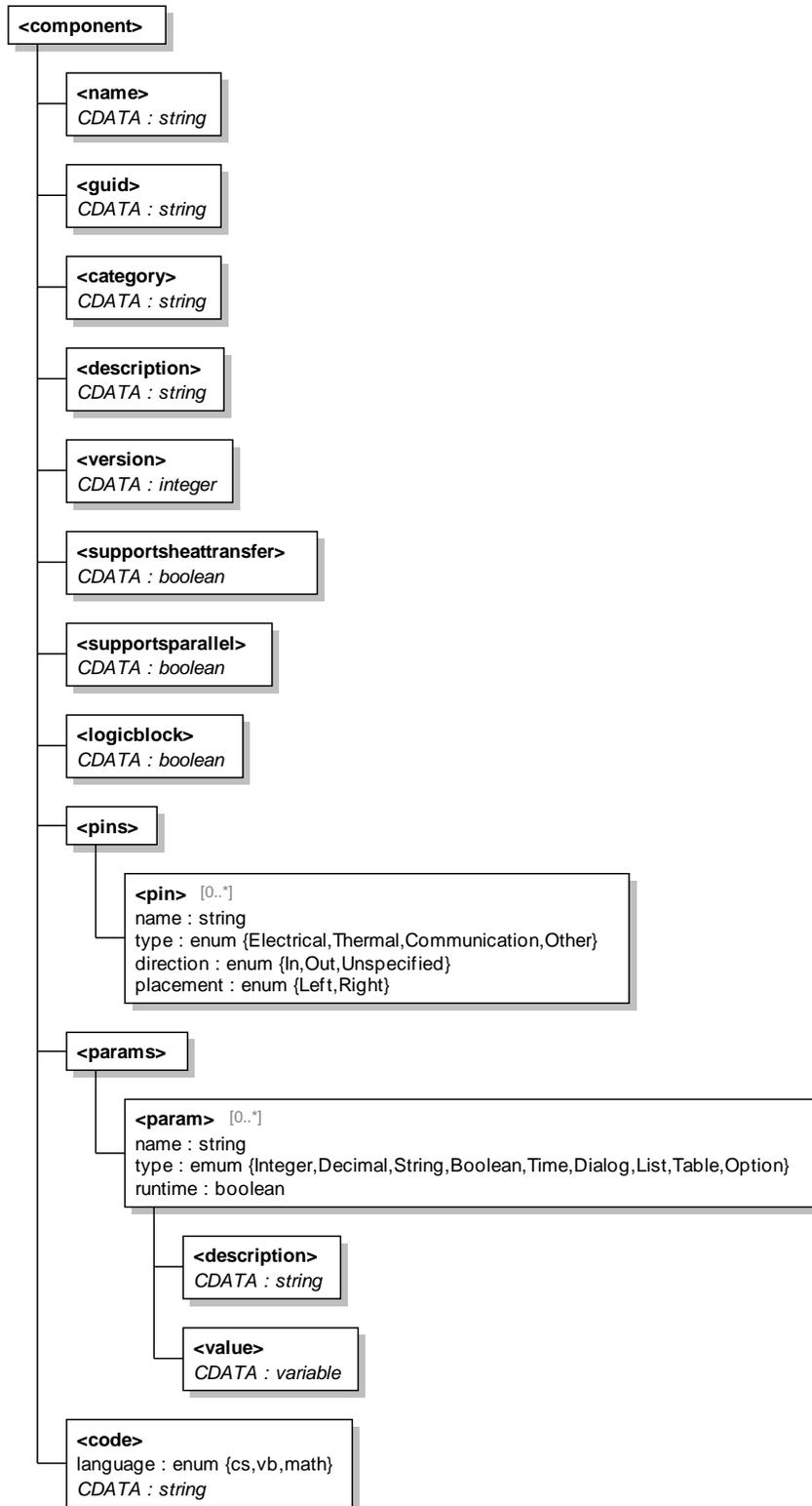


Figure 3-18: XML schema for component behaviour elements.

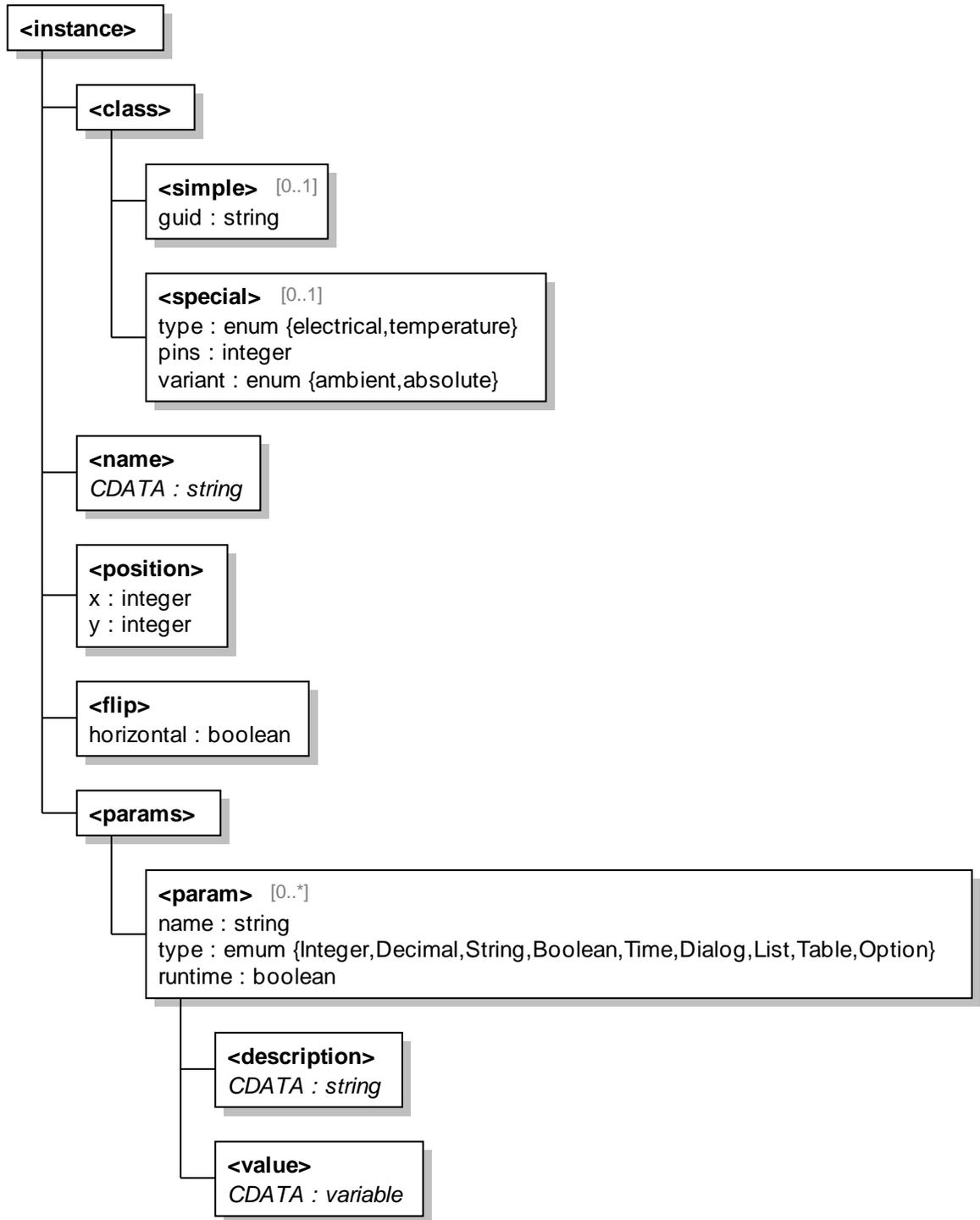


Figure 3-19: XML schema for component instance elements.

3.7. Application Design Summary

The diagram in Figure 3-20 illustrates the architecture of the new domestic smart grid simulation package. The package will perform the simulation of a component-based system, where each component in the system may contain one or more pins that represent an electrical system, heat source or sink or data communications endpoint. The behaviour of each of these components will be governed by source code which can be written in a choice of three languages. Each component may also have one or more configurable parameters to provide control over its behaviour in a specific system. A graphical editor will be provided to enable the creation and editing of libraries of components.

A graphical model editor will be provided to create system models. These models will be created by linking pins on components selected from the available libraries. Models created using this editor can then be passed to the simulation engine within the package which will perform the simulation of the models.

Results generated by the simulation engine are passed to one or more result viewers specified by the user. These can be pie charts, line graphs or tables. The results displayed in these viewers can be exported to image file formats or spreadsheet formats for further processing and analysis.

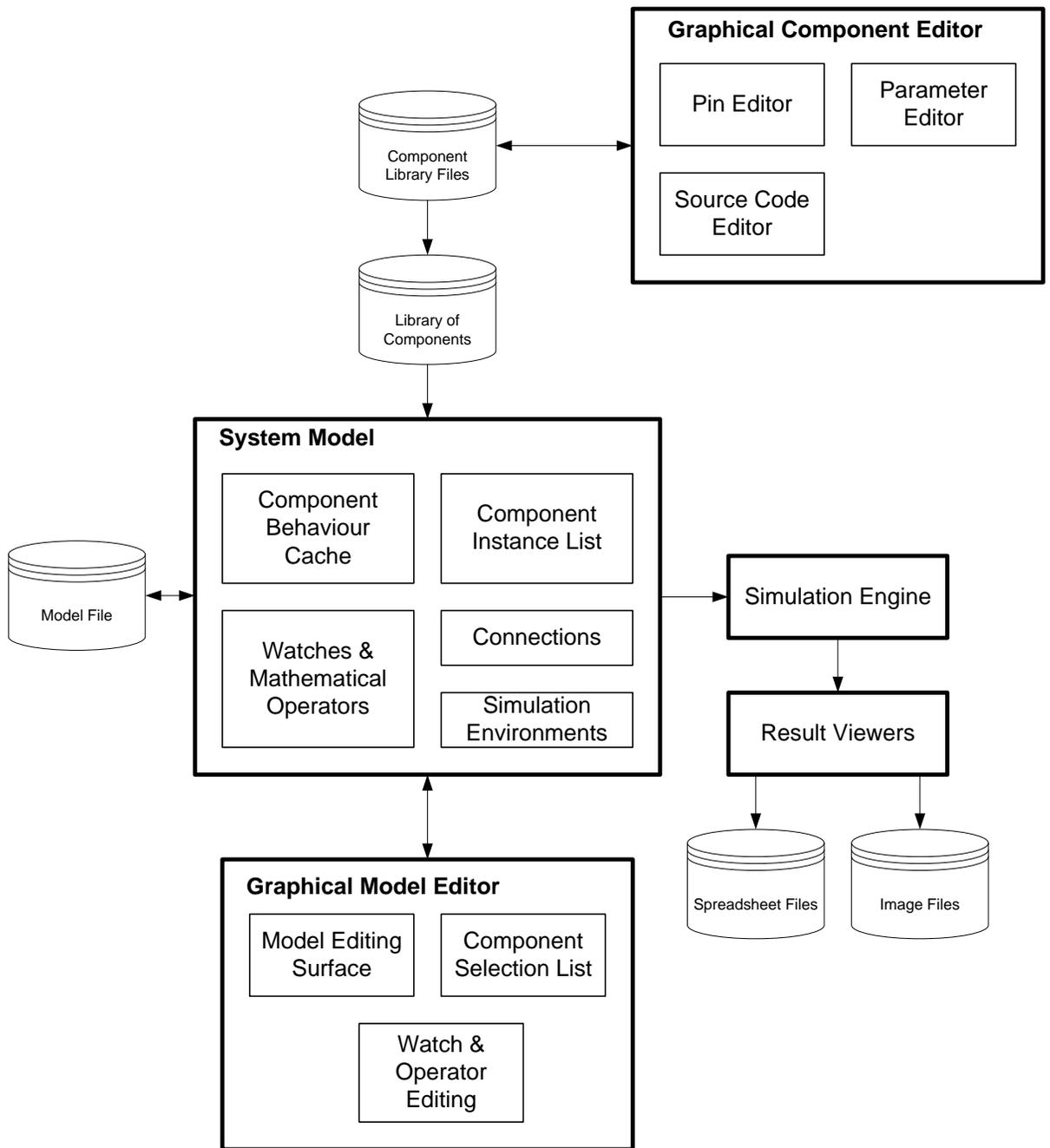


Figure 3-20: Diagram illustrating the architecture of the new simulation package.

Chapter 4

Implementation of Models

Chapter 3 discussed the design and implementation of a new software simulation package for domestic smart grid systems. Detailed information about the implementation of the package is provided in Appendix F. This package is a generic simulation platform for such systems which provides the basic framework upon which system models can be built. To enable such models to be built, libraries of components must also be created for use within the package. This chapter describes the collection of components which have been developed to date.

4.1. Electrical Components

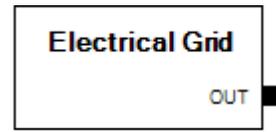
4.1.1. Ideal Sources

Two single-pin components were created to represent idealised DC sources and AC grid connections. These components are implemented as a voltage source in series with zero resistance or impedance. A configurable parameter is provided on each component to allow the required voltage to be specified. On the AC source component, additional parameters are provided to specify the frequency and phase angle (with respect to a system reference of 0°) of the voltage source. Figure 4-1 below illustrates the component symbol and source code for each of these components.



```
#opt single
#param voltage="Voltage"
#pin out="OUT"

out.zmag = 0
out.zphase = 0
out.vrms = voltage
out.dc = 1
```



```
#opt single
#param freq="Frequency"
#param phase="Phase"
#param voltage="Voltage"
#pin out="OUT"

out.zmag = 0
out.zphase = 0
out.vrms = voltage
out.freq = freq
out.phase = phase
out.dc = 0
```

Figure 4-1: Illustration of component symbols and source code for DC and AC ideal voltage source components. Source code for these components is written in the mathematical mark-up language.

4.1.2. Solar Panel

Photovoltaic electricity generation is a popular form of microgeneration: existing buildings can easily be retro-fitted to accommodate the technology through the use of roof-mounted solar panels. Solar cells have a non-linear current-voltage relationship which is commonly modelled using either the two-diode model [109-112] shown in Figure 4-2 or the single-diode model [113-115] shown in Figure 4-3.

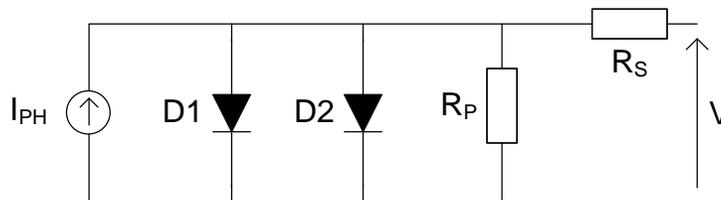


Figure 4-2: Two-diode solar cell model.

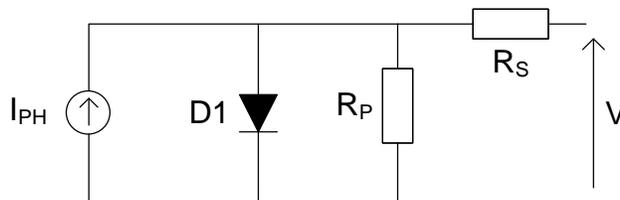


Figure 4-3: Single-diode solar cell model.

For the purposes of the simulation package, the single-diode model was selected since it has fewer parameters to compute when fitting the model to a particular type of solar panel and is less computationally demanding to evaluate.

The single diode model in Figure 4-3 represents a solar cell as a current source in parallel with a diode, a series resistance R_S which represents the losses in the connection to the cell and a shunt resistance R_P which represents losses in the photovoltaic conversion process. This model is described by equation (18) as:

$$I_{OUT}(S, T, V) = I_{PH}(S) - I_{RS}(T) \left(\exp\left(\frac{V + R_S I}{m \cdot V_T(T)}\right) - 1 \right) - \frac{V + R_S I}{R_P} \quad (18)$$

I_{OUT} is the output current of the solar cell (A), V is the voltage across the cell (V), R_S and R_P are the series and shunt resistances (Ω), and m is the dimensionless ideality factor of the diode in the solar cell model. V_T is the thermal voltage for a given cell temperature T (K) given by (19) where k is the Boltzmann constant (1.38×10^{-23} J/K) and q is the electron charge (1.6×10^{-19} C).

$$V_T(T) = \frac{k \cdot T}{q} \quad (19)$$

I_{RS} is the reverse saturation current (A) which is a temperature-dependent loss in the photovoltaic conversion process given by (20) where I_{RSR} is the reverse saturation current at a given reference temperature T_R (K), ϵ is the band gap energy of the semiconductor material used to manufacture the solar cell (1.12eV for silicon) and m is the diode ideality factor described previously.

$$I_{RS}(T) = I_{RSR} \left(\frac{T}{T_R}\right)^3 \exp\left(\frac{\epsilon}{m} \left(\frac{1}{V_T(T_R)} - \frac{1}{V_T(T)}\right)\right) \quad (20)$$

I_{PH} is the photocurrent (A) generated by the cell which is dependent on incident solar irradiation as shown in (21) where I_{PHR} is the photocurrent (A) generated by the cell at a reference irradiation level S_R (W/m^2) and S is the incident irradiation level (W/m^2).

$$I_{PH}(S) = I_{PHR} \cdot \frac{S}{S_R} \quad (21)$$

In equation (18), current appears on both sides of the expression and hence requires the use of numerical methods to obtain a solution. In order to allow the equation to be solved analytically, the assumption can be made that if R_S is sufficiently small and R_P is sufficiently large then these values can be removed from the expression, giving the simplified model in (22).

$$I_{OUT}(S, T, V) = I_{PH}(S) - I_{RS}(T) \left(\exp\left(\frac{V}{m \cdot V_T(T)}\right) - 1 \right) \quad (22)$$

In order to model a panel made up of multiple solar cells connected in series and parallel, some changes must be made to the model in (22). In equation (21), I_{PHR} is given by $I_{PHR} = I_{PHR_PANEL} / N_P$ where I_{PHR_PANEL} is the panel's photocurrent at reference conditions and N_P is the number of parallel networks of cells in the panel. Similarly in (20), I_{RSR} is given by $I_{RSR} = I_{RSR_PANEL} / N_P$. To obtain the output current of the panel, the modified version of (22) given in (23) should be used where N_P is the number of parallel networks of cells in the panel, N_S is the number of cells in series within each parallel network and V is the voltage across the panel.

$$I_{OUT}(S, T, V) = N_P \left(I_{PH}(S) - I_{RS}(T) \left(\exp\left(\frac{V}{N_S \cdot m \cdot V_T(T)}\right) - 1 \right) \right) \quad (23)$$

From (23), the output current of a solar panel can be obtained trivially if the voltage is known. The equation can also be re-arranged to obtain the voltage if the current is known. However, it is non-trivial to determine the voltage and current if only the resistance of the load connected to the panel is known. Within the smart grid simulation package, this is the scenario in which the properties of the solar panel must be calculated.

To simplify the solution to this problem, consider Figure 4-4 which illustrates the model from (23) using the parameters of a BP SX-80 solar panel ($I_{PHR}=4.7A$, $I_{RSR}=9.84 \times 10^{-10}A$, $m=2.13$, $S_R=1000W/m^2$, $T_R=298.15K$), with atmospheric conditions of $S=1000W/m^2$ and $T=25^\circ C$ (298.15K). Additionally, the plot shows the load resistance with respect to voltage.

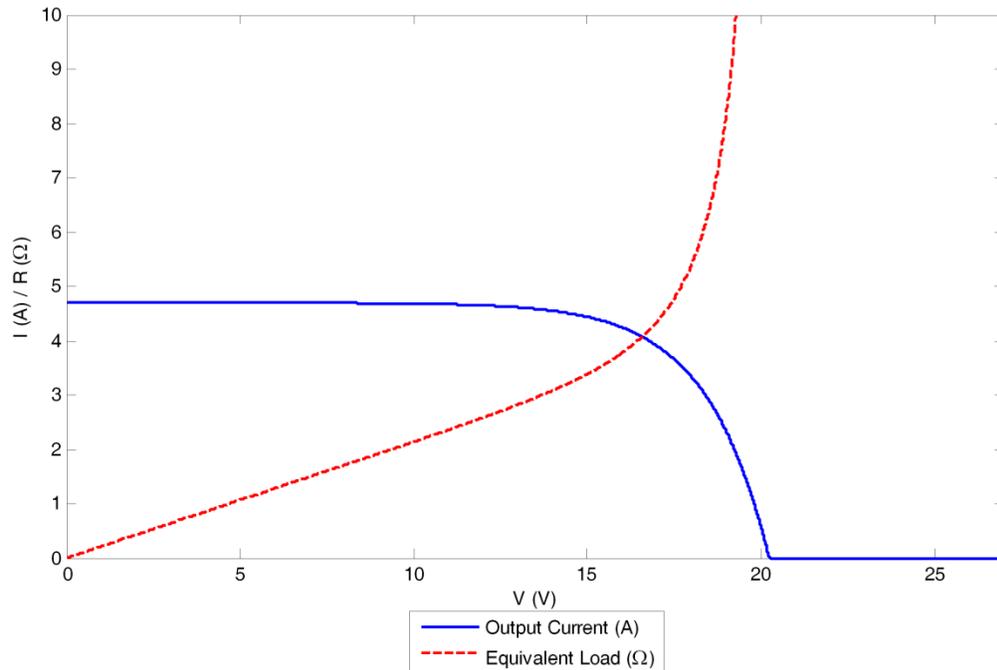


Figure 4-4: Illustration of the I/V characteristic of a BP SX-80 solar panel showing the equivalent load required to obtain each operating voltage.

If the load resistance is known, it is possible to determine the operating point of the solar panel by obtaining the operating voltage from the R/V curve and hence the current using (23). This technique is a variation on the “load line” technique for finding the operating point of a diode as described by Hambley et. al. in [116]. However, when carrying out simulations involving photovoltaic systems, a graphical method provides little benefit and therefore some form of computational equivalent is required. Section 4.1.2.1 describes two existing algorithmic options for the detection of the operating point of a solar panel and presents a new algorithm that was developed during the course of this research.

4.1.2.1. Operating Point Detection Algorithms

Linear Search

The most basic form of evaluation of the operating point is to find the desired point on the R/V curve for the solar panel and hence compute the current for the given voltage. Search

algorithms allow for the R/V curve to be searched for a given load resistance and the voltage to be extracted from the curve. The linear search algorithm [117] is one of the simplest forms of search algorithm.

To determine the operating point using the linear search algorithm, the current should be computed using (23), initially setting $V=0$. The resistance can then be calculated by dividing the voltage by the calculated current. If the resistance is less than the desired load value, the voltage should be incremented by a small value and the process repeated until the algorithm either finds the desired resistance, or finds a value larger than the desired load. The accuracy of this algorithm is directly dependent on the value that the voltage is incremented by – smaller values will produce more accurate results but cause the algorithm to take longer to reach the solution.

Newton's Method

An alternative to searching for the operating point by using the panel's R/V curve is to use Ohm's law in conjunction with equation (23) to express resistance in terms of voltage, as in (24). To determine the operating point of the solar panel, (24) can be solved for V for a given value of R . The voltage obtained may then be used with equation (23) to determine the current.

$$R = \frac{V}{N_P \left(I_{PH}(S) - I_{RS}(T) \left(\exp\left(\frac{V}{N_S m V_T(T)}\right) - 1 \right) \right)} \quad (24)$$

Although there is no analytical solution to (24), Newton's method [118] may be used to solve for the voltage. Newton's method, defined in (25) is a successive approximation solver which begins with an initial estimate at the solution, x_n and uses this to determine a more accurate approximation to the solution, x_{n+1} . This process is continued with each approximate solution until x_{n+1} is within a given tolerance of x_n .

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad (25)$$

In order to solve (24) using Newton's method, it must be rearranged into the form given in (26) and additionally the derivative given in (27) is required.

$$f(v) = \frac{V}{N_P \left(I_{PH}(S) - I_{RS}(T) \left(\exp\left(\frac{V}{N_S m V_T(T)}\right) - 1 \right) \right)} - R \quad (26)$$

$$f'(v) = \frac{1}{N_P \left(I_{PH}(S) - I_{RS}(T) \left(\exp\left(\frac{V}{m N_S V_T(T)}\right) - 1 \right) \right)} + \frac{I_{RS}(T) \cdot V \cdot \exp\left(\frac{V}{m N_S V_T(T)}\right)}{N_S \cdot N_P \cdot m \cdot V_T(T) \cdot \left(I_{PH}(S) - I_{RS}(T) \cdot \left(\exp\left(\frac{V}{m N_S V_T(T)}\right) - 1 \right) \right)} \quad (27)$$

New Operating Point Detection Algorithm

For the purposes of the smart grid simulation package, a new solar panel operating point detection algorithm was developed. The algorithm utilises a variant of the well-known binary search technique [119] to search the R/V curve of a solar panel for a known load resistance to find its operating point.

The algorithm, illustrated in Figure 4-5, begins by setting a voltage variable to zero. Equation (23) is then used to calculate the current and hence the resistance for that voltage. If the resistance is less than the known load resistance, the voltage is incremented by a given step size and the algorithm begins its next iteration. If the resistance is greater than the known load resistance, the voltage is set to its previous value and the step size is divided by 10. This process continues until either the load resistance is found or the step size reaches a specified minimum value.



Figure 4-5: Illustration of the new algorithm developed to determine the operating point of a solar panel when the load resistance is known. The output current $I(V)$ is obtained using (23) and $R(V) = V/I(V)$.

The overall effect of the algorithm is to rapidly converge to an approximation close to the known load value and then to search more precisely to reach the optimum value. The principal property which governs the accuracy of the algorithm is the minimum step size. Smaller values will result in the algorithm taking longer to converge to a more accurate solution while larger values will result in faster convergence to a less accurate solution.

On each simulation time-step, the new operating point detection algorithm is used to determine the operating point of the solar panel for the connected load. However, a further processing step is required to define a Thévenin equivalent model of the panel at this operating point. This step is to find the equation of the tangent to the I/V curve at the computed operating point that provides a linear model of the panel's I/V characteristic which is accurate at the computed operating point. The tangent to the I/V curve at a particular operating voltage V_{OP} is defined by (28) where $I_{OUT}(S,T,V)$ is given by (23) and $I_{OUT}'(S,T,V)$ is the derivative of (23) with respect to V , given by (29). The Thévenin equivalent model can be obtained from the equation of the tangent by setting $V=0$ to obtain the Norton current I_N and then setting $I=0$ to obtain the Thévenin voltage. The Thévenin resistance can be computed using $R_{TH} = V_{TH}/I_N$.

$$I_{TANGENT}(S, T, V) = I_{OUT}(S, T, V_{OP}) + I_{OUT}'(S, T, V_{OP})(V - V_{OP}) \quad (28)$$

$$I_{OUT}'(S, T, V) = -N_P \cdot I_{RS}(T) \cdot \frac{\exp\left(\frac{V}{m \cdot N_S \cdot V_T(T)}\right)}{m \cdot N_S \cdot V_T(T)} \quad (29)$$

4.1.2.2. Comparison of Operating Point Detection Algorithms

A comparison of the new algorithm was carried out with the linear search method and the Newton method to measure its performance against that of the other algorithms. Comparisons will be performed on the range of resistance values for which the algorithms can obtain a solution, the time taken to reach a solution and the accuracy of the solutions obtained. The comparison was carried out by selecting voltage points between 0V and 25V in increments of 0.01V and calculating the current and resistance for each for the a BP SX-80 module. The calculated resistances are used as the test inputs to each of the algorithms and the voltage and current values are used as the reference operating point for each resistance which the values calculated by the algorithms can be compared to. To provide a fair comparison between each algorithm, the tolerance for the Newton solver was set at 1nV and the minimum step sizes for the linear search algorithm and the new algorithm were also set at 1nV. The Newton solver was set to have an initial guess of 0V for the operating point in each instance.

The first comparison that was carried out was of the range of resistance values for which the algorithms can obtain a solution. Figure 4-6 shows the results for linear search algorithm, Figure 4-7 shows the results for the Newton method algorithm and Figure 4-8 shows the results for the newly developed algorithm. It can be seen from these results that the linear algorithm is able to solve over the entire range of resistances provided to it, the Newton method algorithm begins to fail nearer the open circuit voltage and new algorithm solves for all given values, with a slight error near the open circuit voltage. The reason for the Newton solver failing near the open circuit voltage is that equation (30) has a stationary point near the open circuit which will cause a division by zero error in the Newton method calculation when it operates in this region.

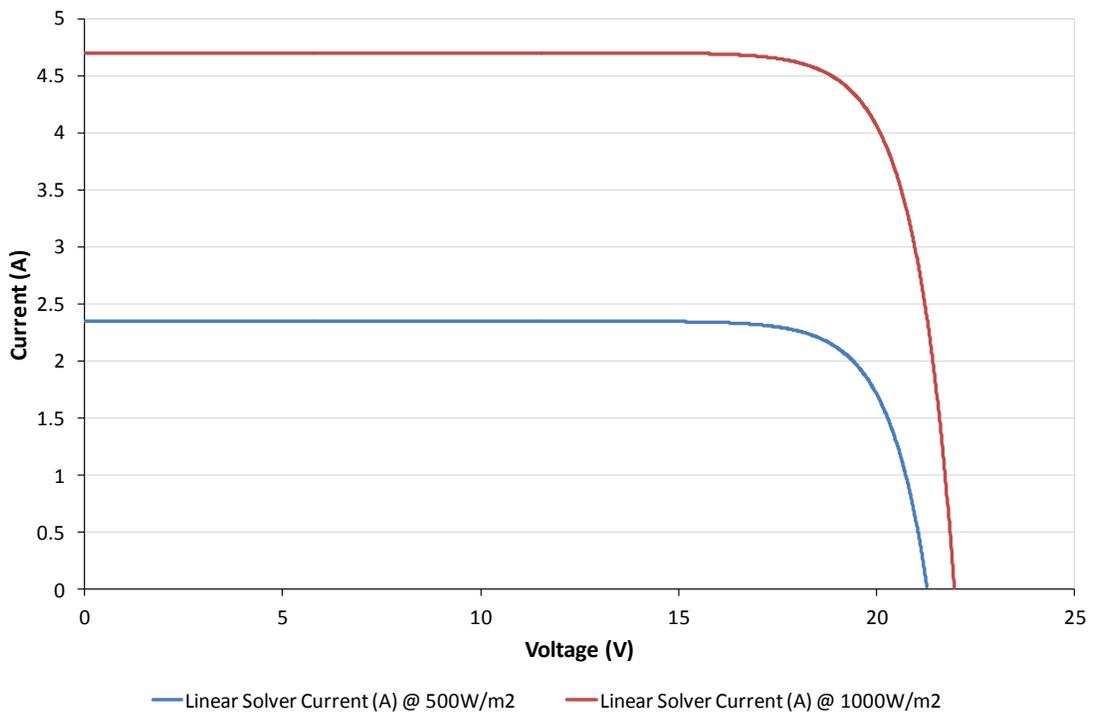


Figure 4-6: Illustration showing the range over which the linear search algorithm was able to obtain I/V operating points for the BP SX-80 solar panel.

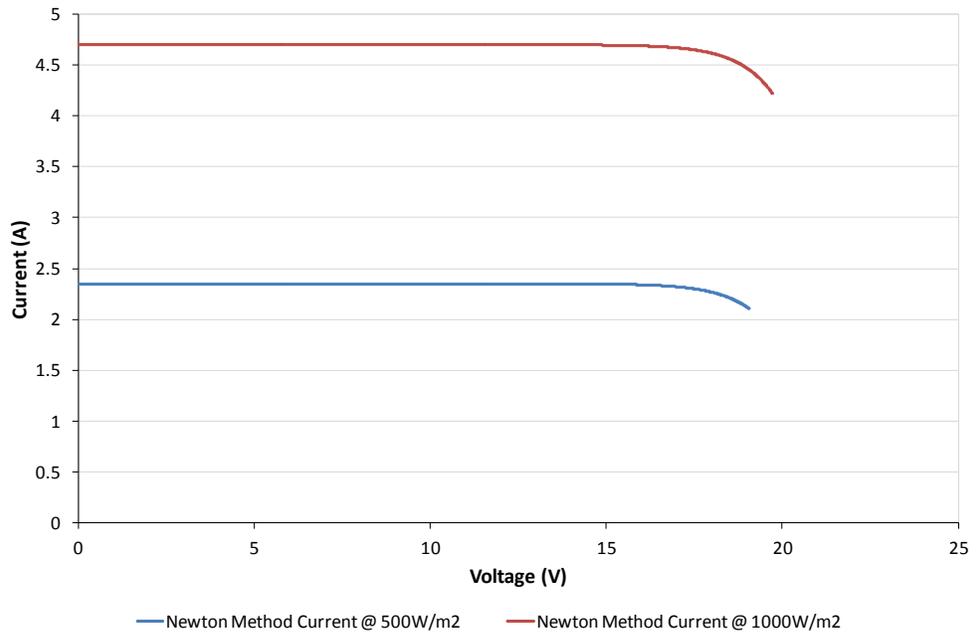


Figure 4-7: Illustration showing the range over which the Newton's method algorithm was able to obtain I/V operating points for the BP SX-80 solar panel.

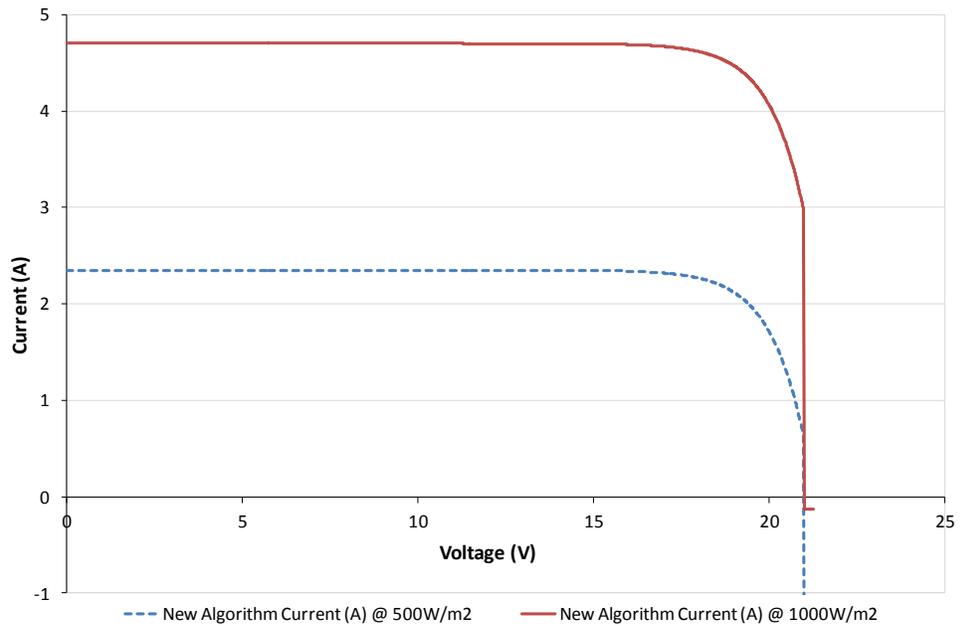


Figure 4-8: Illustration showing the range over which the newly developed algorithm was able to obtain I/V operating points for the BP SX-80 solar panel.

The second comparison was the time that was taken for each algorithm to obtain a solution for the same set of test data used previously. The results of this comparison using an

irradiation of 1000W/m² are shown in Figure 4-9. It can be observed from these results that Newton's method was the fastest to reach a solution with an average time of 0.01ms, followed by the new algorithm with an average time of 0.1ms. The linear search was considerably slower as the time taken to obtain each solution was directly proportional to the resistance value. The average time for the linear search algorithm to find a solution was 2.14s.

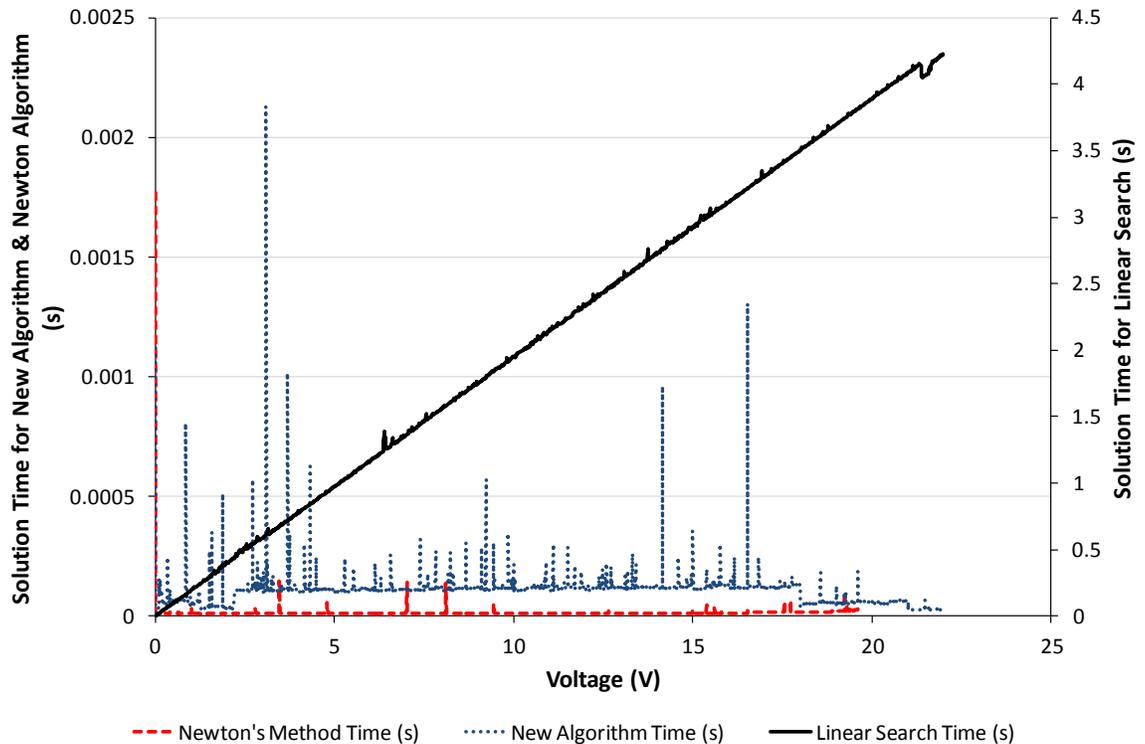


Figure 4-9: Comparison of the time taken to reach a solution for each algorithm using an irradiation level of 1000W/m². The values for 500W/m² are excluded from this illustration because they are of similar magnitude.

The final comparison performed between the three algorithms was one of the accuracy of each algorithm. In this case, the error between the operating voltage obtained by each algorithm and the actual operating voltage of the panel was used for the comparison. The comparison was carried out using irradiation levels of 500W/m² and 1000W/m². The results are presented in Table 4-1. It can be observed from this table that the Newton's method solver provides the most accurate results, followed by the linear search algorithm with the new algorithm performing most poorly. It should be noted, however that the large error

contribution in the new algorithm is in the voltage range within 1V of the open circuit voltage, which is well beyond the range that the Newton method solver is capable of obtaining solutions in. If this range is disregarded then the maximum error is reduced to around $1 \times 10^{-9} \text{V}$ in both cases.

Table 4-1: Comparison of result errors in operating point detection algorithms.

Algorithm	500W/m ² Comparison			1000W/m ² Comparison		
	Minimum Error (V)	Average Error (V)	Maximum Error (V)	Minimum Error (V)	Average Error (V)	Maximum Error (V)
Linear Search	0	6.56×10^{-7}	1×10^{-6}	0	6.35×10^{-7}	1×10^{-6}
Newton's Method	0	9.61×10^{-17}	1.06×10^{-14}	0	3.91×10^{-17}	1.06×10^{-14}
New Algorithm	0	0.012	1	0	0.022	1

4.1.3. Battery Storage

Energy storage was highlighted in Chapter 2 as one of the technologies which could become important in mitigating the supply stability issues introduced by adding increasing amounts of renewable generation to the electricity network. A representation of a battery was added to the simulation package in order to allow for the modelling of electrical storage at the domestic level. A number of different battery chemistries are available and all have the potential for use in future storage systems. These include Lithium Ion (including Lithium Polymer), Nickel Cadmium, Nickel Metal Hydride and Lead Acid. Different battery chemistries have different electrical characteristics and therefore different mathematical models defining their behaviour. A number of mathematical models have been created for simulating the properties of batteries, some battery chemistry specific [120-122] and others independent of battery chemistry [123]. The model that was chosen for use is the battery model provided within the SimElectronics package within the Matlab / Simulink software [124]. This generic battery model models batteries as a controlled voltage source in series with a resistance which makes it ideal for use in the simulation software. The only dependent parameter within the model is the state of charge and hence it is less computationally intensive to evaluate during a simulation compared to other models.

The voltage of the source V (V) within the battery model is described by (30) where V_0 (V) is the battery open-circuit voltage when fully charged, x (%) is the state of charge of the battery and α and β are constants which govern the charge and discharge rate of the battery. The model is simplified by making the assumption that the charge and discharge I/V characteristics of the battery are the same.

$$V = V_0 \left(\frac{\alpha(1-x)}{1-\beta(1-x)} \right) \quad (30)$$

The battery model parameters α and β are calculated using (31) and (32) respectively where V_1 (V) is the battery open-circuit voltage at a known state of charge x_1 (%) which is less than 100%.

$$\alpha = -\frac{V_0x_1 - V_1x_1}{V_1x_1 - V_1} \quad (31)$$

$$\beta = -\frac{V_1 - V_0x_1}{V_1x_1 - V_1} \quad (32)$$

The series resistance in the battery model is the internal resistance of the battery which governs the charge and discharge rate of the battery.

Within the simulation package, the battery model was implemented with the following configurable parameters: Fully-charged open circuit voltage (V_0 , V); Calibration Voltage (V_1 , V); Calibration Charge State (x_1 , %); Battery Capacity (Q_T , Ah); Initial Charge ($Q_{INITIAL}$, Ah); Short-Circuit Current (I_{SC} , A).

The battery model is initialised by setting the internal resistance to V_0 / I_{SC} . The state of charge x is initialised to ($Q_{INITIAL} / Q_T$). On each time-step of a simulation, the change in state of charge is calculated by performing a trapezium rule integration using the instantaneous output current value measured at the battery's output terminal (negative if charging) and the instantaneous current measured during the previous time-step. Logic within the model prevents the state of charge from falling below 0Ah or from going above the total capacity Q_T . After calculating the change in state of charge, the new battery voltage V is calculated using (30).

4.1.4. Connection and Control

Each of the electrical connection and control components is inserted between two components to perform either a control action such as switching or to model the properties of a conductor. Connection or control components are therefore desired to behave like the schematic shown in Figure 4-10 where V_1 , V_2 , Z_1 and Z_2 model the properties of the two components and $Z_{\text{CONNECTION}}$ is the series impedance of the connection or control component.

Due to the way that electrical pins are modelled in the simulation package, inserting a two-pin intermediate component between two other components results in the equivalent schematic shown in Figure 4-11.

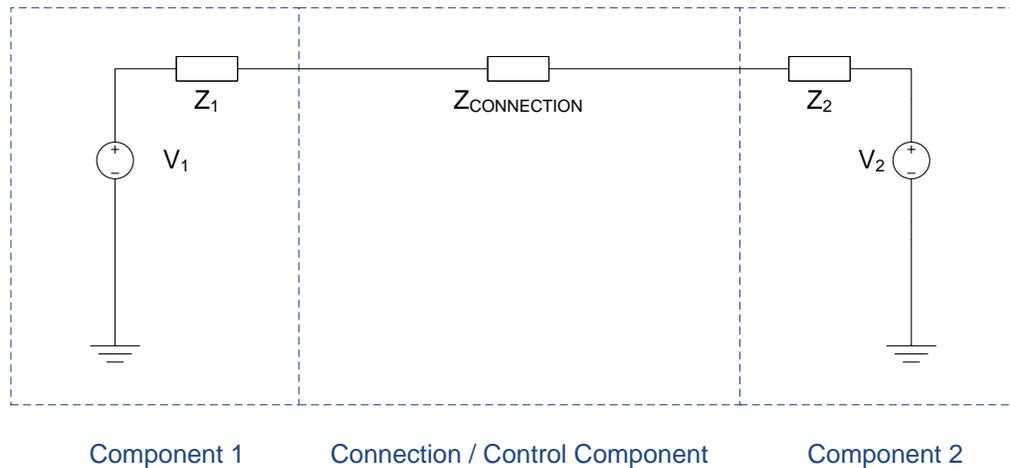


Figure 4-10: Schematic of a connection or control component with impedance $Z_{\text{CONNECTION}}$ inserted in series between two components.

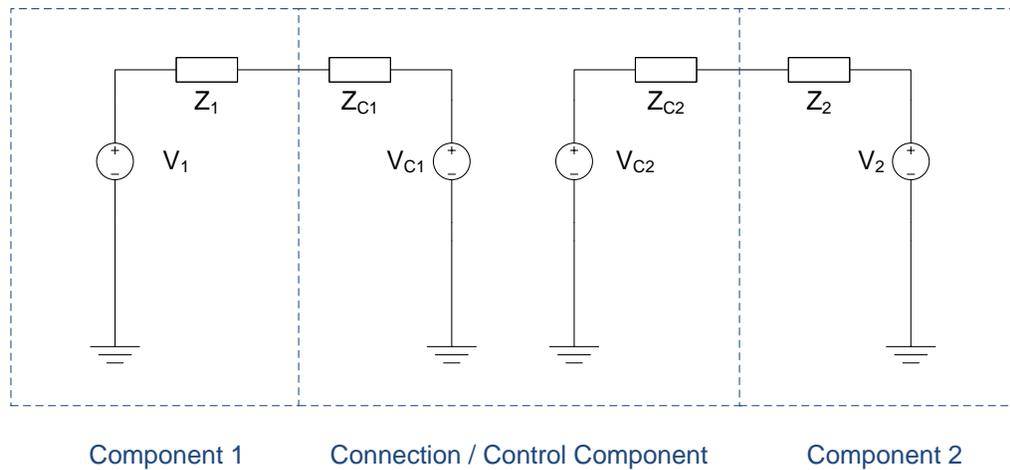


Figure 4-11: Schematic illustrating the way in which the simulation package requires a series component to be modelled.

In order to model a connection or control component which follows the schematic shown in Figure 4-10 using the topology set in Figure 4-11, the properties of the component must be set as shown in (33).

$$\begin{aligned}
 V_{C1} &= V_2 \\
 V_{C2} &= V_1 \\
 Z_{C1} &= Z_2 + Z_{\text{CONNECTION}} \\
 Z_{C2} &= Z_1 + Z_{\text{CONNECTION}}
 \end{aligned}
 \tag{33}$$

The properties shown in (33) are set during the **Evaluate** phase of connection and control components' code execution. The **Evaluate** function will return *false* if any changes are made to the connected pin properties V_1 , V_2 , Z_1 or Z_2 and *true* if these properties have not changed since the last evaluation of the component's behaviour. This will ensure that the iterative simulation engine continues to run until a stable circuit is obtained for each side of the connection and control component.

This generic approach is used for all of the connection and control components that were designed. The remainder of this section details with the specific behaviour of each of the different components.

4.1.4.1. Wire

Electrical connections between two components in the simulation package are ideal zero-resistance connections; however, wiring in a building has a non-zero resistance which is a factor that must be considered in building models, especially when long cable runs are used around a building. A special component, shown in Figure 4-12, was included in the package to account for the series resistance introduced by wiring. This component can be connected between any two electrical components.

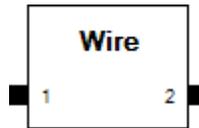


Figure 4-12: Component which models the series resistance introduced by electrical wiring.

The component has the following configurable parameters: wire construction material (Copper, Aluminium, Silver or Lead), wire length (metres), wire cross-sectional area (mm^2 or AWG).

The resistance of the wire, R (Ω), is calculated during the component's **Reset** phase using (34) below where ρ is the resistivity of the material used to construct the wire (Ωm), l is the length of the wire (m) and A is the cross-sectional area of the wire (m^2).

$$R = \frac{\rho l}{A} \quad (34)$$

The cross-sectional area of the wire is converted from mm^2 to m^2 by multiplying by 10^{-6} . In the case where the wire size is specified in AWG, formula (35) is used to convert the AWG size to mm^2 , before then being converted to m^2 .

$$A_{\text{mm}^2} = 0.012668 \times 92^{\left(\frac{36-AWG}{19.5}\right)} \quad (35)$$

A lookup table of standard resistivity values for Copper, Aluminium, Silver and Lead is used to retrieve the resistivity for the selected wire material.

Once the resistance of the wire has been obtained, the method described at the beginning of section 4.1.4 is used in the component's **Evaluate** phase to model the wire component as a series resistance.

4.1.4.2. Switch

The switch component is designed to allow for the simulation of users switching appliances on or off. The component uses the asynchronous communications mechanism of the package to receive "ON" or "OFF" messages to control its state. When in the *on* state, the series resistance of the switch is set to a contact resistance which is specified through a configurable parameter. When in the *off* state, the series resistance of the switch is set to infinity, making it act as an open circuit.

4.1.4.3. Digital Switch

The digital switch component is designed to provide electronic control over an appliance using a communications link from a control device. The switch has a configurable one-byte address which is used to uniquely distinguish it from other switches using the same communications network. When a two byte packet containing the configured single-byte address followed by 0 is received, the switch is set to the *off* state. When the address is received followed by 1, the switch is set to the *on* state. The digital switch behaves in an identical manner to the basic switch component in the *on* and *off* states by acting as a configured contact resistance when *on* and as an open circuit when *off*.

4.1.4.4. Relay

The relay component is designed to provide lower-level electronic control over appliances than the digital switch. In place of the communications pin, the relay provides an electrical pin on which it expects a DC control voltage. The coil resistance and switch-on threshold for this pin can be set through configurable parameters. Similarly to the previous two switch components, the relay functions as a configured contact resistance when the control voltage is above the switch-on threshold and as an open circuit when the control voltage is below the threshold value.

4.1.4.5. Smart Meter

A smart electricity metering component was seen as an essential element of a smart grid modelling package because, as discussed in section 2.1.5, smart metering has been identified as one of the key features of future domestic energy systems. A metering component is also important when designing new smart energy control systems as it can illustrate the energy used by the system during a given period of time, or the total cost of energy used.

The implementation of the smart meter component is based on the Smart Meter Technical Specification that was made publicly available by British Gas [39]. Following the requirements set out in this specification allows the smart meter component to closely follow the features available in industry-standard meters. Additional communication features relating to the control of in-home appliances, storage and generation systems which are not yet in use in deployed meters have also been added to provide a degree of future-proofing to the software package. The meter component, shown in Figure 4-13, has electrical connections for the in-home power system and the electricity supplier network. Communication connections are provided for communication with the electricity supplier and with in-home systems.

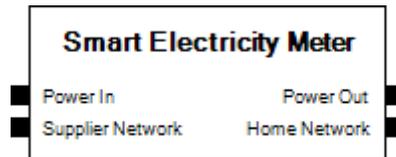


Figure 4-13: Smart electricity meter component.

The European Standard BS EN 50470-1 [125] specifies the requirements for the electrical and physical properties of electricity metering devices used in the UK. Among these requirements is that the meter should monitor active (real) power consumption over the metered time period as shown in (36), where P is the metered power in kWh over the given time period T in hours and p is the instantaneous power in kilowatts.

$$P = \int_0^T p dt \quad (36)$$

This is the calculation that is used within the smart metering component to monitor power consumption. As simulations within the modelling package are carried out in discrete time steps, the trapezium integration rule [118] is used to perform the integration of power as shown in (37).

$$P = \text{TIMESTEP_SIZE} \times \frac{\text{POWER} \mp \text{PREVIOUS_TIMESTEP_POWER}}{2} \quad (37)$$

The electricity meter's characteristics will be modelled using the method for modelling series components defined at the beginning of section 4.1.4, with the meter having zero insertion resistance – essentially behaving as a short circuit between the input and output pins. Instantaneous power flow out of a pin on a component is a property provided by the package's scripting API hence no calculation of power flow is required within the component's logic.

Three power consumption tariff systems are provided within the component. These tariff systems were selected because they are available within the smart meters currently being deployed by British Gas in the UK. These are the Time of Use (TOU), Tiered Time of Use (T-TOU) and Critical Peak Pricing (CPP) systems. The Time of Use (TOU) tariff system divides the 24 hours of the day into 48 half-hour segments and a price is specified for each segment. The Tiered Time of Use (T-TOU) operates by using an 8x8 grid of electricity prices. The columns within the grid represent 3-hour segments of the day. The appropriate column for looking up electricity prices is selected by the meter based on the time of day. Each row within the grid represents a different pricing tier. The appropriate row to look up electricity prices in is based on the amount of energy already used in the current billing period. For example, one rate may be charged for the first 250kWh used, followed by a lower or higher rate for the next 250kWh and so on. Either TOU or T-TOU is selected within the meter as the active pricing scheme for normal electricity use.

The Critical Peak Pricing (CPP) tariff is used in conjunction with one of the two regular pricing schemes mentioned above. Critical peak periods are invoked by the utility provider sending a message to the meter to notify that a critical peak period is in operation, during which a significantly higher price per kWh will be charged. A second message returns the meter to the regular TOU or T-TOU scheme when the critical peak period is over. The meter stores a single configurable price to use during critical peak periods.

The smart metering component also has support for a feed-in tariff in the case that local microgeneration is generating more power than can be used locally and is feeding power back into the grid. For simplicity, a single configurable rate is used by the meter for billing power fed back into the grid.

Three different types of communication are supported on the utility provider side of the meter. These are:

- Notification of entering a critical peak period.
- Notification of leaving a critical peak period.
- Message to request meter readings. The meter responds with a 48x1 table of TOU readings or an 8x8 table of T-TOU readings, depending on the tariff that is currently in use.

To enable control of domestic appliances based on meter pricing events, three user-configurable tables are specified within the component. These specify communication messages to be sent in response to changing prices. The three types of messaging supported are:

- Sending one or more messages when the pricing tier changes.
- Sending one or more messages when the Time-Of-Use period changes.
- Sending one or more messages when entering or leaving a critical peak period.

Additionally, devices within the home may communicate with the meter to query the billing price per kWh in use or the instantaneous power consumption.

Configuration of the smart meter component is undertaken through the use of a number of configurable parameters provided on the component. These are shown in Table 4-2. Results can be collected from the component by monitoring one or more of the read-only parameters shown in Table 4-3.

Table 4-2: Parameters used to configure the smart meter component.

Parameter	Type	Description
Pricing Scheme	Option List	Specifies the default pricing scheme to use for normal metering – TOU or T-TOU.
TOU Prices	48x1 Table	Specifies the price per kWh in £ for each of the time of use pricing slots.
T-TOU Prices	8x8 Table	Specifies the price per kWh in £ for each of the T-TOU pricing slots.
T-TOU Thresholds	8x1 Table	Specifies the kWh thresholds for each of the tiers in the T-TOU scheme.
T-TOU Period	Integer	Specifies the billing period over which the T-TOU pricing scheme operates before resetting back to the first tier.
Feed-In Rate	Decimal	Specifies the price in £/kWh for energy fed back into the grid.
CPP Rate	Decimal	Price in £/kWh to use during critical peak periods.
Meter ID	Integer	A unique ID number for the meter which is used in communications with the utility provider. This allows multiple meters to be used on the same communications link.

Table 4-3: Parameters that store metering results within the smart meter component.

Parameter	Type	Description
Instantaneous Power	Decimal	Instantaneous power flow in kW.
Active TOU Slot	Integer	A number (1-8 for T-TOU or 1-48 for TOU) indicating the active time of use slot used for billing.
Active Tier	Integer	A number (1-8) indicating the active billing tier.
CPP Active	Boolean	A flag indicating whether or not Critical Peak Pricing is active.
Feed In Active	Boolean	A flag indicating whether or not the meter is in feed-in mode.
Tiered Rate Register	Decimal	The energy use in kWh since the beginning of the current billing period if T-TOU is active.
Tiered Rate Day Count	Integer	The number of days since the beginning of the current billing period if T-TOU is active.
Active Rate	Decimal	The current rate in £/kWh that is being used to bill electricity.
TOU Readings	48x1 Table	A table holding the kWh readings for each billing slot if the TOU scheme is in use.
T-TOU Readings	8x8 Table	A table holding the kWh readings for each billing slot if the T-TOU scheme is in use.
CPP Reading	Decimal	The amount of energy used in kWh during critical peak periods.
Feed In Readings	Decimal	The amount of energy in kWh that has been fed back into the grid.
Consumption Cost	Decimal	The cost of energy consumed in £ during normal billing periods.
CPP Cost	Decimal	The cost of energy consumed in £ during CPP periods.
Feed In Payment	Decimal	The amount due from the utility provider in £ for energy fed back into the grid.

4.1.5. Loads

Domestic electrical loads are classified into multiple groups for the purposes of modelling in software. These groups are: static loads which have constant power consumption when switched on; multi-mode loads which have constant power consumption for each of their operating modes (for example, full power and standby); time-varying loads in which power consumption varies predictably over the duration of use of the appliance; and dynamic loads in which electric power consumption is based on the properties of another dependent parameter (for example the temperature within a water heating appliance).

Each of the load components has a single electrical connection which represents its connection to the home AC power system. The pin is modelled as an impedance-to-ground connection as shown in Figure 4-14. The magnitude and phase angle properties of the impedance are evaluated by each component's logic as described in sections 4.1.5.1 - 4.1.5.4.

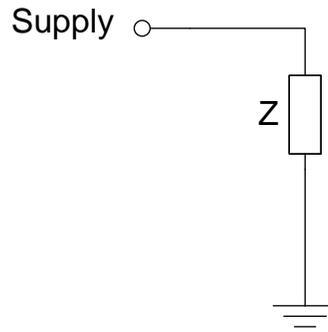


Figure 4-14: Model used for an electrical load component.

4.1.5.1. Static Loads

Static loads – loads where the power consumption is constant while power is applied to the device – were characterised in a set of in-home experiments that were carried out during this project. An in-line power analyser was used to measure the voltage, current and the phase difference between voltage and current. The value of the impedance for the model was calculated using Ohm's law as in (38) where Z is the complex impedance value used in the model, V is the measured RMS voltage (V), I is the measured RMS current (A) and Φ is the phase difference between the current and voltage ($^{\circ}$).

$$Z = \frac{V \angle 0^{\circ}}{I \angle \phi^{\circ}} = \left(\frac{V}{I}\right) \angle -\phi^{\circ} \quad (38)$$

Table 4-4 shows the measured properties of the devices that were characterised and the resulting impedance value that was used in the model for each.

Table 4-4: Measured operational characteristics of household appliances and calculated impedance value for use in models.

Appliance	Measured Voltage (V)	Measured Current (A)	Measured Phase Angle (°)	Model Impedance	
				Magnitude (Ω)	Phase Angle (°)
Microwave Oven	236.5	6.54	24.5	36.16	-24.5
Incandescent Light Bulb	238	0.237	0	1004.22	0
Carbon Fluorescent Light Bulb	238	0.08	-50.95	2975.00	50.95
Wireless Router	239	0.053	-58.67	4509.43	58.67
Vacuum Cleaner	234	5.46	14.07	42.86	-14.07
Electric Shower	238	38.57	8.11	6.17	-8.11

4.1.5.2. Multi-Mode Loads

Loads which have similarly constant power consumption properties to those in the “static loads” group but have more than one operating mode where the power consumption may be different were categorised as multi-mode loads. These loads include appliances which have constant power consumption but which also have, for example, a standby mode. The components were categorised and implemented in the same way as the static loads, with the impedance value to use being selected depending on the value of a state variable indicating the operating mode. The operating mode of each appliance can be changed during a simulation using the asynchronous communications mechanism within the package. Upon receiving an asynchronous message, a component compares the message to the names of its operating modes and if a match is found, the operating mode is changed. Table 4-5 below illustrates the measured properties of a variety of the components for each operating mode and the calculated impedance.

Table 4-5: Measured operational characteristics of household appliances with multiple operating modes and calculated impedance for use in models.

Appliance	Operating Mode	Measured Voltage (V)	Measured Current (A)	Measured Phase Angle (°)	Model Impedance	
					Magnitude (Ω)	Phase Angle (°)
Phone Charger	On	237	0.049	-50.95	4836.73	50.95
	Standby	236	0	0	∞	0
LCD TV	On	238	0.495	-25.84	480.81	25.84
	Standby	237	0.059	-85.41	4016.95	85.41
XBOX 360	On	238	0.315	-53.13	755.56	53.13
	Standby	238	0.047	0	5063.83	0
PC	On	230	0.54	-42.27	425.93	42.27
	Sleep	230	0.053	-65.17	4339.62	65.17
	Off	230	0.043	-77.88	5348.84	77.88
TFT Monitor	On	237	0.135	-61.31	1755.56	61.31
	Standby	237	0.05	-85.41	4740.00	85.41

4.1.5.3. Time-Varying Loads

A washing machine was identified as an appliance which could be suitably modelled as a load which varies in a predictable way over time. A number of typical washing machine cycles (40°C normal spin, 40°C short spin, 60°C normal spin, 60°C short spin) were recorded using an AC power analyser and used to create lookup tables within the washing machine component’s logic. The component supports commands using the asynchronous messaging system to indicate a programme being started by the user (START 40N, START 40S, START 60N, START 60S). The instantaneous load value for the component’s model is determined by identifying the time-point within the selected programme that the machine is currently at and selecting the relevant load magnitude and phase angle from the lookup table. If the machine is not currently running a programme then it is modelled as an infinite resistance (i.e. zero power consumption).

A second more generic time-varying load component, named “Time-Varying Load” was included which allows a load pattern to be followed based on an imported table of values. This will allow for the modelling of any time-varying load not included in the standard library of components.

4.1.5.4. Dynamic Loads

During the course of the project, the only “Dynamic Load” device which was implemented was an electric water heater. The motivation behind the implementation of this water heating component was to provide a water-based energy storage device which could be used within domestic demand side management experiments later on in the project. This electrical load is classed as a dynamic load within this discussion as the electricity consumed by the device is related to the internal water temperature within the tank which is related to the end-user consumption of water.

In order to develop a simplified model of a hot water immersion heater, the properties of a commercially available immersion heater were used for reference. The heater that was selected was the ECSd100-580 immersion heater which is manufactured by Dimplex. The properties of this heater are summarised in Table 4-6.

Table 4-6: Properties of the Dimplex ECSd100-580 100 litre 3kW immersion heater.

Height of Tank	0.81m
Diameter of Tank	0.58m
Insulation	6cm Polyurethane Foam
Tank Material	0.5cm Stainless Steel ⁷
Water Storage Volume	100 litres
Heating Element	1x 3kW element
Maximum Water Temperature	90°C
Inlet Water Flow Rate (Minimum)	15 litres / min (Output at Mains Pressure)
Heat Loss	0.9kWh/24 Hours (Estimated)

The first stage in implementing a simplified, generic immersion heater model was to define a constant for the tank’s thermal resistance. As no specific conditions were placed on the manufacturer-specified heat loss in Table 4-6, the assumption was made that within the component logic, the thermal resistance of the tank would be calculated for a set condition of 75°C water temperature (the mid-point between the 90°C maximum and the UK regulation of 60°C minimum storage temperature). The ambient temperature will be taken as

⁷ The tank wall thickness (including insulation) was not explicitly stated in the specification and has therefore been estimated from the other parameters provided, assuming a cylindrical tank shape.

20°C when calculating the thermal resistance of the tank. Based on a configurable heat-loss parameter within the component (kWh/24h), the thermal resistance (°C /W) of the tank material is therefore calculated as shown in (39).

$$\text{Thermal Resistance} = \frac{75 - 20}{\frac{24}{1000} \cdot \text{Heat Loss}} \quad (39)$$

The second stage of developing the water tank model was the simulation of the cooling of water within the tank due to the heat loss through the casing. In order to simplify this calculation, the assumption was made that the water in the tank is always well-stirred and has uniform temperature throughout. The specific heat capacity of water in the tank is taken to be 4180J/kg·°C. The assumption is also made that the tank is always full. The mass of water in the tank is therefore equal to its volume (1 litre of water = 1kg).

Using Newton's Law of Cooling, the temperature decrease in the tank's water during one time-step of a simulation can be calculated using (40) where T_{NEW} is the new temperature of the water in the tank after cooling (°C), T_{OLD} is the temperature of the water in the tank prior to the calculation (°C), T_{AMBIENT} is the ambient temperature (°C), t is the simulation time-step size in seconds, R_{TH} is the thermal resistance of the tank casing (°C /W), C_P is the specific heat capacity of water (J/kg·°C) and m is the mass of water in the tank (kg).

$$T_{\text{new}} = T_{\text{ambient}} + (T_{\text{old}} - T_{\text{ambient}}) \cdot e^{\left(\frac{-t}{R_{\text{TH}} \cdot C_P \cdot m}\right)} \quad (40)$$

The third stage within the model's evaluation is the calculation of the heat loss in the tank due to hot water consumption. For this stage of the calculation, a configurable parameter is supplied to define the temperature at which hot water should be delivered. A monthly table is provided of cold water inlet temperatures. In immersion heater systems, the water in the tank may be significantly higher than the hot water delivery temperature and the required hot water temperature is delivered through the use of a thermostatic mixing valve to mix hot and cold water.

The heater is initially assumed to be supplying no hot water at the beginning of a simulation and water flow is indicated to the heater by other components within a model through the use of the asynchronous messaging capability. Components send messages to the tank in the format “FLOW Q” where Q is a flow rate in litres per minute. On each time-step the component calculates the temperature decrease due to water flow as shown in (41).

$$\text{Mix Percentage} = \frac{T_{\text{DESIRED}} - T_{\text{COLD}}}{T_{\text{HOT}} - T_{\text{COLD}}}$$

$$\text{Hot Percentage} = \begin{cases} 0 & \text{Mix Percentage} < 0 \\ 1 & \text{Mix Percentage} > 1 \\ \text{Mix Percentage} & 0 \leq \text{Mix Percentage} \leq 1 \end{cases}$$

$$\text{Hot Flow Rate (l/min)} = \text{Total Flow Rate} \cdot \text{Hot Percentage} \quad (41)$$

$$\text{Volume of Water Replaced} = \frac{\text{Hot Flow Rate} \cdot \text{Timestep Size (s)}}{60}$$

$$\text{New Temperature} = \left(\frac{\text{Volume Replaced}}{\text{Total Volume}} \right) \cdot T_{\text{COLD}} + \left(1 - \frac{\text{Volume Replaced}}{\text{Total Volume}} \right) \cdot T_{\text{HOT}}$$

The final stage in the evaluation of the heater’s behaviour is the calculation of the temperature increase in the stored water when the heating elements are active. The assumption is again made that the water in the tank is well-stirred with uniform temperature throughout and the assumption that the heating elements always transfer their full rated electrical power as thermal energy when heating the water. The temperature of the water after heating for a single time-step, T_{NEW} (°C) is calculated using the water temperature before the heating calculation T_{OLD} , the power delivered by the heating elements P (W), the specific heat capacity of water C_P (J/kg·°C), the mass of water in the tank m (kg) and the time-step size t (s) as shown in (42).

$$T_{\text{NEW}} = \frac{P \cdot t}{C_P \cdot m} + T_{\text{OLD}} \quad (42)$$

On each time-step of the simulation, the new water temperature of the tank is calculated by first performing the heat-loss calculation, followed by the water-flow calculation and finally

the heating calculation. An electrical connection on the component models the power drawn by the heating elements.

The decision on whether to switch the heating elements on or off is made by comparing the tank temperature to a thermostat temperature setting. The initial value of this setting, along with its dead-band, are configurable parameters on the component. The thermostat value of the water heater may be adjusted during a simulation by sending messages of the format “TEMP X” to the heater’s communication port, where X is the new thermostat setting.

4.2. Communication Components

4.2.1. Scheduled Data Generator

A number of the communication events that take place within a smart grid system (for example, pricing events sent by a utility provider to a smart meter) may, in a particular simulation scenario, occur at “hard-coded” times. These can therefore be modelled by a component which contains a look-up table of messages to be sent at a particular time. The Scheduled Data Generator was included within the package for this purpose. The component has a single communications pin on which messages are sent. A configuration table is included within the component with each row in the table storing a time and a message that should be sent over the communications channel at that time. For entries where the message dispatch time falls between two simulation time-steps, the message is sent on the latter of the two time-steps. The component allows for a multiple messages to be specified for the same time-step.

4.2.2. Repeater

The Repeater component is a communications infrastructure component that allows a simple communications network to be formed between many different components. Each repeater component has 8 communication pins which provide point to point connections between the repeater and the connected component. Any message received on one of the communications ports on the repeater is forwarded to every other port on the repeater. This allows a single component to communicate with up to seven other components.

The repeater component provides no routing capability and therefore the content of messages sent in a network made up of repeaters must contain information that indicates the destination of a message.

4.3. Building Elements

The modelling of building construction was a key requirement in the new smart grid simulation package. To enable this modelling, a set of components for modelling building surfaces, heated spaces, doors and windows was created. Each of these components was based on the models described in the international standards that are used to calculate the heat loss from buildings when developing specifications for heating systems. Additions were made to these standardised models when necessary – either when the standards did not provide a model for a particular building element or when more detail was required than the model presented in the standard was able to provide.

4.3.1. Materials Database

When modelling building elements, the thermal properties of the materials that they are made up from are an important factor. In order to avoid duplication of the thermal properties of building materials across the implementation of multiple modelling components, a database was created containing the thermal properties of a standard set of building materials. A table of materials already exists in the form of the standard BS EN ISO 10456 [126]. This standard contains a tabulated list of common building construction materials along with the density (kg/m^3), design thermal conductivity ($\text{W/m}\cdot\text{K}$), specific heat capacity ($\text{J/kg}\cdot\text{K}$) and wet and dry water vapour resistance factors (μ -value) of each.

The information contained in the ISO standard was compiled into a database and embedded in a DLL library file that can be loaded by components' C# or VB.NET source code. An API provided in the DLL allows the database to be queried for either a list of materials or for one of the thermal properties of a given material.

4.3.2. Surface

The surface component was designed to represent any kind of surface that separates two thermal zones within a building, or separates a zone in a building from the exterior. The roof, exterior walls, interior walls, ceilings and floors can all be modelled using the surface component. The principle of the surface component was derived from the standard BS EN 12831:2003 [89] which describes the calculation of design heat load for buildings. Within this standard, a surface is modelled as an area made up of layers of material, each with a defined thickness and thermal transmittance. The calculations in the standard are solely for steady-state behaviour of the building and therefore only take into account the thermal transmittance of surfaces and not their ability to store heat.

In order to provide a more accurate dynamic model of surface behaviour, the surface component was designed to also take the heat capacity of each material within the surface into account. In order to allow for this calculation, the configurable parameters for the component allow a surface of up to eight layers to be defined in terms of: surface area A (m^2), thickness of each layer l (m) and material for each layer. The materials available within the model are the set of materials defined in the database described in section 4.3.1. Each of these materials has a thermal conductivity Φ ($W/m\cdot K$), density ρ (kg/m^3) and specific heat capacity c_p ($J/kg\cdot K$) which is used in the surface calculations.

During the “reset” phase of the component’s evaluation, a model of the surface is initialised from the specified parameters. This model takes the form shown in Figure 4-15.

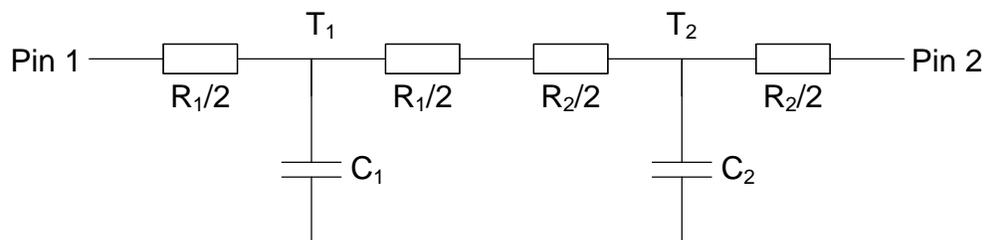


Figure 4-15: Schematic diagram illustrating the model used for a surface component.

Within this model, C_x (J/K) represents the heat capacity of a given layer X. This is calculated from the thickness of the material l (m), area of the surface A (m²), density of the material ρ (kg/m³) and specific heat capacity of the material c_p (J/kg·K) as shown in (43).

$$\begin{aligned} V &= l \cdot A \\ m &= \rho \cdot V \\ C &= c_p \cdot m \end{aligned} \tag{43}$$

R_x (K/W) represents the thermal resistance of a given layer X. This is calculated from the thickness of the material l (m³), area of the surface A (m²) and design thermal conductivity Φ (W/m·K) as shown in (44).

$$\begin{aligned} U &= \frac{\Phi}{l} \\ R &= \frac{1}{U \cdot A} \end{aligned} \tag{44}$$

To simplify the thermal calculation of the surface, rather than model the temperature gradient across each material within the surface, the assumption is taken that the temperature of each material T_x (K) is the temperature at the centre of the material. The material is therefore modelled as shown in Figure 4-15 with half of the thermal resistance on each side of the central temperature.

To initialise the temperature gradient across the surface, the initial temperatures provided in the simulation model are used in conjunction with the thermal resistance of the surface's materials to calculate the temperature of each material within the surface. As an example, consider a scenario where in the schematic in Figure 4-15, material 1 has a thermal resistance of 0.2 K/W and material 2 has a thermal resistance of 0.4 K/W. Also consider that the initial temperature of the zone connected to the pin 1 of the component is 20°C and the initial temperature of the zone connected to pin 2 of the component is 22°C. This would produce a model of the format shown in Figure 4-16.

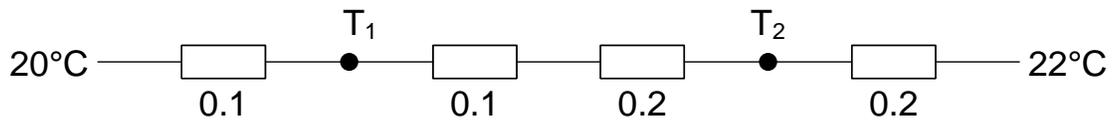


Figure 4-16: Example of initial surface temperature gradient calculation.

In this example, the total thermal resistance is 0.6W/K and the total temperature difference across the surface is 2K. The temperatures at T_1 and T_2 respectively are calculated as a proportion of the total temperature difference across the surface as shown in (45) and (46) respectively.

$$T_1 = 20^\circ\text{C} + 2^\circ\text{C} \cdot \left(\frac{0.1}{0.6}\right) = 20.33^\circ\text{C} \quad (45)$$

$$T_2 = 20^\circ\text{C} + 2^\circ\text{C} \cdot \left(\frac{0.4}{0.6}\right) = 21.33^\circ\text{C} \quad (46)$$

To determine the dynamic characteristics of a material within the surface, the material is considered to be modelled in the form shown in Figure 4-17. In this model, T_1 and T_2 (K) represent the temperatures on either side of the material. These are either the temperatures of neighbouring materials within the surface or the temperatures of the zones that the surface is connected to in the case of a material at one of the edges of the surface. The thermal resistances R_1 and R_2 (K/W) are the sum of half of the material's thermal resistance and either: half the neighbouring material's thermal resistance in the case of an internal connection to another material within the surface; or, the thermal resistance of another component connected to the surface in the case of a material at the edge of the surface. C_{MATERIAL} is the heat capacity of the material (J/K).

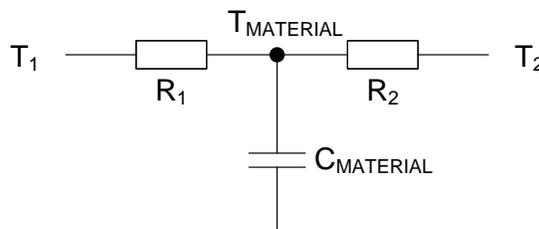


Figure 4-17: Dynamic model of a material within a surface.

The first step in determining the dynamic temperature of a material is to compute the steady state temperature of the material using (47).

$$T_{STEADY} = T_1 + (T_2 - T_1) \cdot \left(\frac{R_1}{R_1 + R_2} \right) \quad (47)$$

The steady state temperature is then used in Newton's cooling equation [127] as shown in (48) to calculate the new material temperature based on the current temperature.

$$T_{MATERIAL(N+1)} = T_{STEADY} + (T_{MATERIAL(N)} - T_{STEADY}) \cdot \exp \left(\frac{-t_{step}}{R_P \cdot C_{MATERIAL}} \right) \quad (48)$$

In this equation, t_{step} (s) is the time-step size used in the simulation and R_P is the parallel combination of the thermal resistances R_1 and R_2 , calculated using (49).

$$R_P = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2}} \quad (49)$$

Once the temperature of each material within a surface has been evaluated for a given time-step, the final equivalent model of the surface can then be derived for that time-step in the format shown in Figure 4-18. In this model T_A and T_B (°C) are the temperatures of the materials at each edge of the surface. R_A and R_B (K/W) are half of the thermal resistance of the material at each edge of the surface.

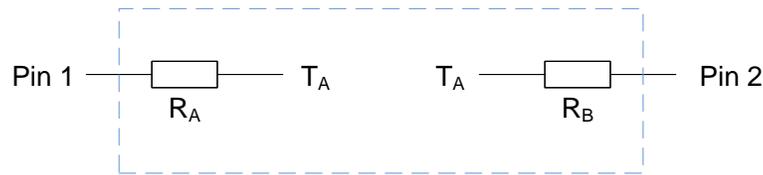


Figure 4-18: Final surface model.

4.3.3. Door

The door model that is currently implemented uses the same logic as the surface component to model a closed door separating two zones within a building or a door to the exterior of the building. Future improvements to the door model will include the calculations from the ISO

10077-1 standard to allow for doors with glazed sections and calculations for the effect of infiltration around the edges of a door. It would also be desirable to include an option for an open door to be modelled.

4.3.4. Window

The window component that was developed is based on the ISO standard 10077-1:2006 [128] which specifies a method for the calculation of the thermal transmittance of doors, windows and shutters. At present, only a calculation of heat conduction using this standard is carried out – the component does not support the solar gain calculations that are carried out in more established window models [56, 129].

4.3.4.1. ISO 10077-1 Window Geometry

The ISO standard defines a standard method of modelling the geometry of a window. This method defines a number of properties of the window which are required in order to perform the thermal transmittance calculation. In the standard, a window is defined as being made up of the frame (which includes any moveable sashes) and a number of glazed areas or opaque panels. Table 4-7 describes the geometrical measurements that are required for glazed areas and opaque panels.

Table 4-7: Description of the measurements of glazed areas or opaque panels that are required for the ISO10077-1 window thermal transmittance calculation.

Parameter	Units	Description
l_g or l_p	m	The perimeter of the glazed area or opaque panel in metres. If the perimeter is different on each side then the larger value should be used.
A_g or A_p	m^2	The area of the glazed area or opaque panel. This is the smallest area visible from either side of the window.

Table 4-8 describes the geometrical measurements that are required for the frame of the window. Each of these measurements is illustrated in the diagram in Figure 4-19.

Table 4-8: Description of the measurements of the frame that are required for the ISO10077-1 window thermal transmittance calculation.

Parameter	Units	Description
$A_{f,i}$	m^2	The area of the internal window frame, including moveable sashes, which is parallel to the glazed or panelled area.
$A_{f,e}$	m^2	The area of the external window frame, including moveable sashes, which is parallel to the glazed or panelled area.
A_f	m^2	The frame area – the maximum of $A_{f,i}$ and $A_{f,e}$.
$A_{f,di}$	m^2	The internal developed frame area – the total area of the internal window frame which is in contact with the air.
$A_{f,de}$	m^2	The external developed frame area – the total area of the external window frame which is in contact with the air.

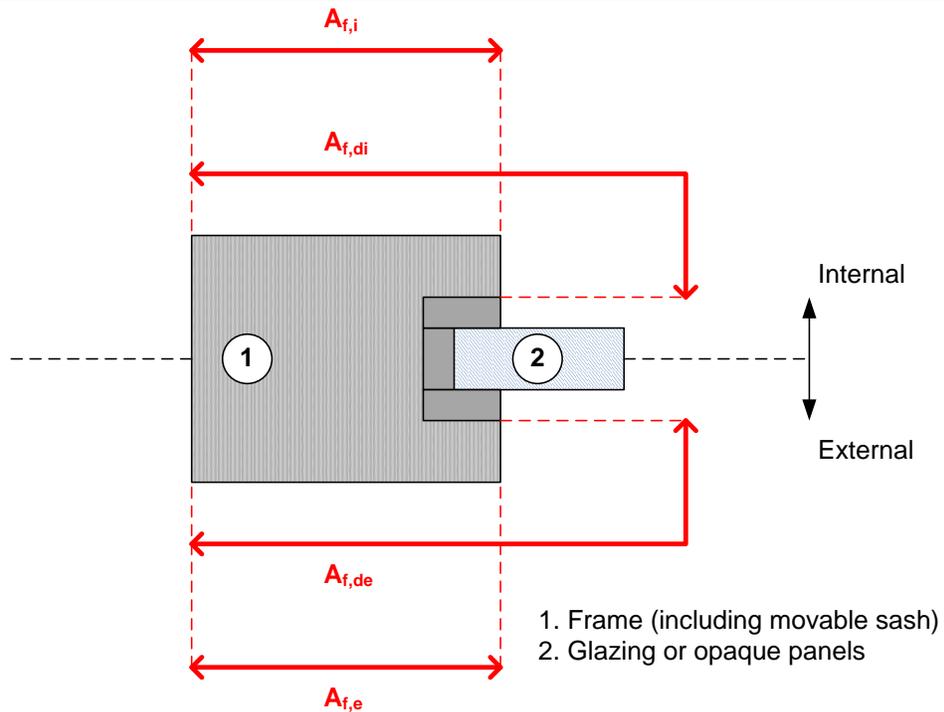


Figure 4-19: Dimensions of a window frame as required by ISO10077-1 for the calculation of the window's thermal transmittance.

4.3.4.2. Software Modelling of Window Geometry

The window geometry used by the thermal transmittance calculation combines a number of properties of the window's geometry into single calculation parameters using assumptions such as including any moveable sashes as part of the frame. In order to provide a method to easily define a window's geometry within a model, a hierarchical method of defining a window as a combination of the frame, moveable sashes and glazed areas or opaque panels

was used. An illustration of this hierarchical window model is shown in Figure 4-20. In order to simplify the specification of a window model, the following assumptions are made: all frames, glazed areas and opaque panels are rectangular; the faces of the frame and all moveable sashes are parallel or perpendicular to the glazing of the window; and the frame and all moveable sashes are constructed from the same material.

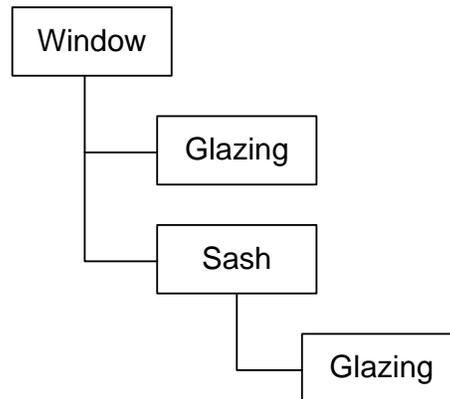


Figure 4-20: Hierarchical model of a window.

The root component in this model represents the frame of the window. It will be defined by its width, height and thickness, all in metres, and the material that the window is constructed from. The list of available materials will be restricted to polyurethane, UPVC, hardwood and softwood since standard values for each of these materials are included in the ISO standard. The window component will contain references to each of the glazed areas, opaque panels and moveable sashes included within it.

A moveable sash⁸ will be defined as illustrated in Figure 4-21. The inner and outer dimensions are defined to allow for overlap with the frame on one side as shown. The thickness of the sash in metres will be defined as well as a dimension D_{OPEN} which specifies the distance that the sash protrudes from the frame on the opening side of the window. The position of the sash on the window is defined in terms of the X and Y distances in metres of the top-left corner of the sash from the top-left corner of the window frame.

⁸ This model is only valid for hinged sash windows. A separate model would be required for sliding sash windows.

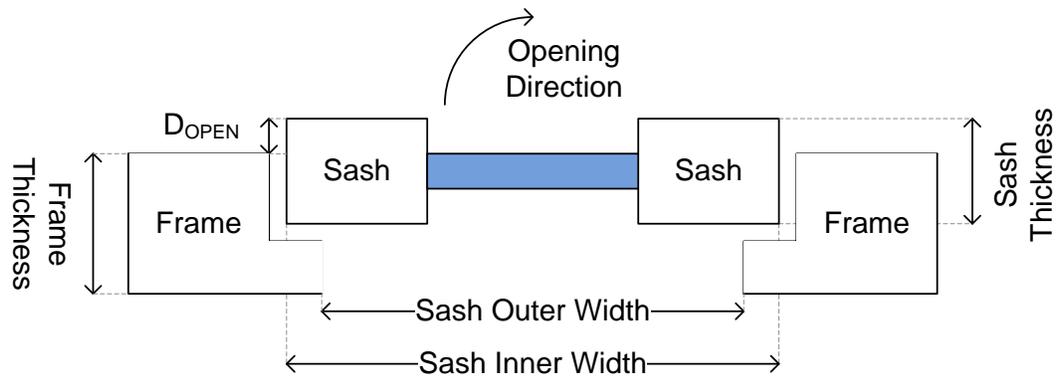


Figure 4-21: Illustration of how a moveable sash is defined within the window component.

The final components within the hierarchical window model are the glazed or opaque panelled areas on a window. Similarly to moveable sashes, the position of these will be expressed as an offset in metres from the top-left corner of the frame or moveable sash that contains the glazed or panelled area. The thermal properties of an opaque panel are defined by selecting the material that the panel is made of from the materials database described in section 4.3.1. The thermal properties of single glazing are specified by defining the thickness of the glazing in metres. Calculation of the thermal properties of double glazing is specified in terms of three properties: glass type, normal emissivity and gas between glazing layers. A lookup table of glass type, emissivity and gas is provided in the ISO standard, along with the thermal conductivity for each combination of options. To simplify the calculation of the window's thermal properties, only the combinations provided in the standard are supported in the hierarchical window model.

The window hierarchical model is stored within a window component as the "Custom Configuration Dialog" type parameter. The parameter displays the dialog shown in Figure 4-22 to allow users to define the geometry and thermal properties of the window. These properties are then stored in the class structure illustrated in Appendix B for use by the window model.

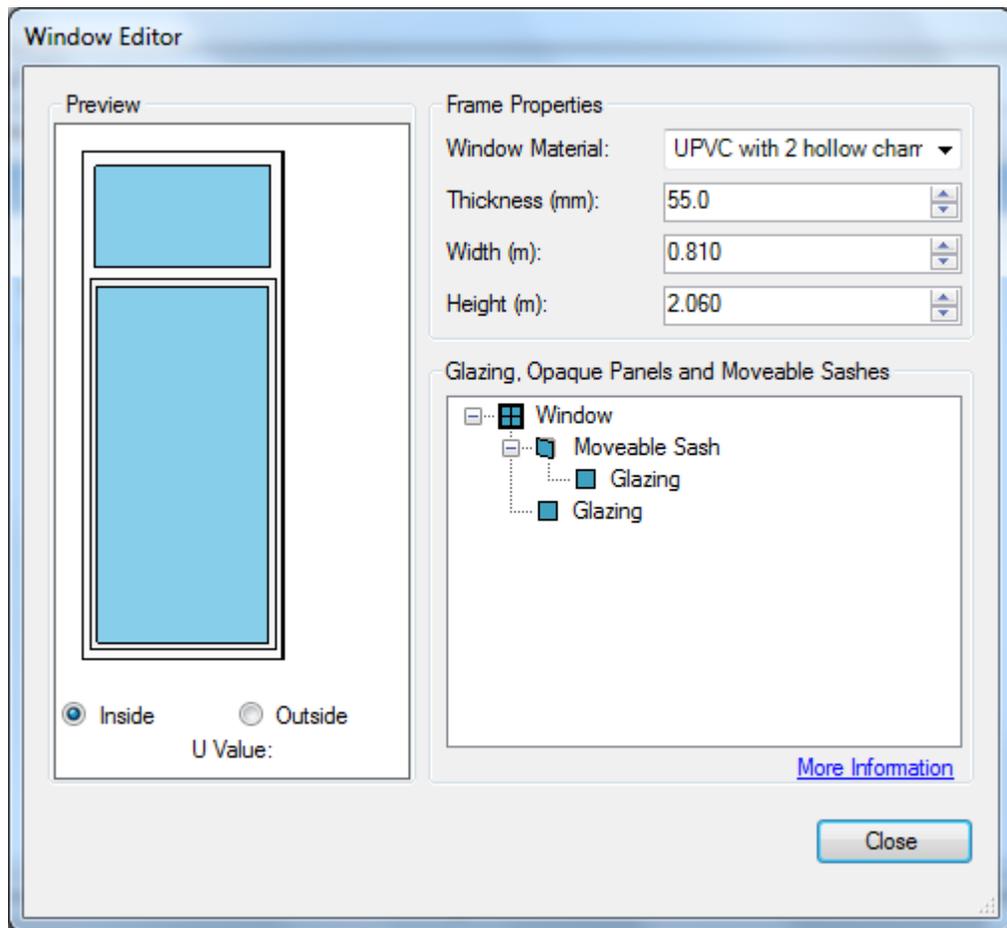


Figure 4-22: Custom configuration dialog included within the “Window” component to allow for the definition of window properties.

4.3.4.3. Translating Hierarchical Model to ISO Standard Model

The following steps are taken by the Window component logic to translate the hierarchical window model into an ISO standard model:

1. Extract A_g and l_g for each glazed area on the window directly each glazing definition.
2. Extract A_p and l_p for each opaque panel directly from each panel definition.
3. Calculate $A_{f,i}$, $A_{f,e}$ and hence A_f by computing the area of the frame including moveable sashes which does not contain glazing or opaque panels on each side of the window.
4. To calculate $A_{f,di}$ and $A_{f,de}$, it will be assumed that all glazed areas or moveable sashes are exactly at the mid-point of the sash or frame in which they are installed. The

exposed thickness of frame perpendicular to the glazing or opaque panel will therefore be calculated using $l_{\text{EXPOSED}} = ([\text{FRAME THICKNESS}] - [\text{GLAZING OR PANEL THICKNESS}]) / 2$. This can then be used in conjunction with the glazing or panel's width or height to calculate the exposed areas. The internal and external dimensions of each moveable sash will be used to calculate the areas of the sash that are perpendicular to the frame for use in this calculation.

4.3.4.4. Thermal Transmittance Calculation

The ISO standard defines the thermal transmittance of a window U_W ($\text{W}/\text{m}^2\cdot\text{K}$) using equation (50). The parameters of this equation are as described in section 4.3.4.1, as well as the following parameters: U_g ($\text{W}/\text{m}^2\cdot\text{K}$) – thermal transmittance of a glazed area; U_p ($\text{W}/\text{m}^2\cdot\text{K}$) – thermal transmittance of an opaque panel; U_f ($\text{W}/\text{m}^2\cdot\text{K}$) – thermal transmittance of frame material; φ_g ($\text{W}/\text{m}\cdot\text{K}$) – linear thermal transmittance of a glazed area; φ_p ($\text{W}/\text{m}\cdot\text{K}$) – linear thermal transmittance of an opaque panel. The linear thermal transmittance takes into account the combined thermal effect of the glazing, spacers and frame in double or triple glazed windows. Standard values for this parameter are provided for different types of frame. The value can be set to zero when single glazing is used. The standard also specifies that a value of zero can be used when opaque panels with thermal conductivity of less than $0.5\text{W}/\text{m}\cdot\text{K}$ are used. The assumption that this is always the case is used to simplify the window model.

$$U_W = \frac{\sum A_g \cdot U_g + \sum A_p \cdot U_p + \sum A_f \cdot U_f + \sum l_g \cdot \varphi_g + \sum l_p \cdot \varphi_p}{\sum A_g + \sum A_f + \sum A_p} \quad (50)$$

The ISO standard provides tables of values of U_f for various frame materials and tables of U_g for double and triple glazing of various types. Lookup tables of these values are used within the component logic to compute these values. For single glazing, equation (51) is used to calculate U_g where d is the thickness of the glazing. This equation assumes non-laminated, vertically positioned soda lime glass.

$$U_{g,single} = \frac{1}{0.17 + d} \quad (51)$$

4.3.4.5. Window Model Definition

Once the thermal transmittance of the window has been determined using equation (50), this is used in conjunction with the total area of the window A_w (m^2) to obtain the linear thermal resistance R_w (K/W) of the window as shown in (52).

$$R_w = \frac{1}{U_w \cdot A_w} \quad (52)$$

The calculated linear thermal resistance is then used as shown in Figure 4-23 to define a model of the window that is compatible with the simulation package. In this model R_1 and R_2 are the thermal resistances of the pins connected to pin 1 and pin 2 of the window respectively and T_1 and T_2 are the temperatures defined for the pins connected to pins 1 and 2 of the window.

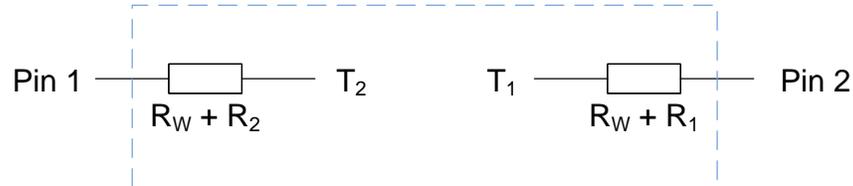


Figure 4-23: Schematic illustrating equivalent thermal resistance model for the window component.

4.3.5. Room

The room component is designed to model a room or other space within a building (for example a loft or basement space). The component models the temperature of the air within the room based on a given initial room temperature and room volume. The assumption is made that the air within the room is well-stirred and has a uniform temperature throughout. Four variants of the room component are provided with 4, 8, 16 and 24 external connections, although the internal logic is identical for each.

The external connections on a room component are used to connect to the surfaces, doors and windows which join the room to the surrounding areas as well as components which model heating or ventilation sources. Each of these connected components is represented by a temperature source in series with a thermal resistance, as described in section 3.3.2.

In order to model the thermal characteristics of the room, the electrical circuit analogy shown in Figure 4-24 can be used where the capacitance of the capacitor C_{ROOM} represents the heat capacity of the air within the room in J/K; T_{ROOM} represents the instantaneous temperature of the air in the room in °C. T_{STEADY} represents the steady-state temperature of the room in °C – this is the temperature that the air in the room will eventually reach should the present conditions of the room’s surroundings, heating and ventilation remain the same. θ_{RATE} is a thermal resistance (K/W) which governs the rate at which the room will heat up or cool down to its steady state temperature.

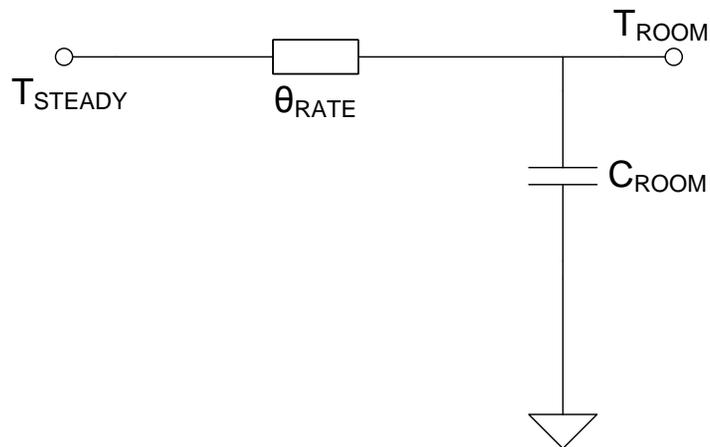


Figure 4-24: Electrical analogy for the thermal model of a room.

The steady state temperature T_{STEADY} and heating rate resistance θ_{RATE} model the net power flow into the room from all of the other thermal components which are connected to the room component. To calculate these values, assume that if there are N other components connected to the room, there are two vectors which contain the temperatures and thermal resistances of the connected components $T = (T_1, T_2, \dots, T_N)$ and $\theta = (\theta_1, \theta_2, \dots, \theta_N)$. Using these values, the steady state temperature of the room can be calculated using (53) and the charge rate resistance can be calculated using (54).

$$T_{STEADY} = \frac{\sum_{n=1}^N \frac{T_n}{\theta_n} \prod_{m=1}^N \theta_m}{\sum_{n=1}^N \frac{1}{\theta_n} \prod_{m=1}^N \theta_m} \quad (53)$$

$$\theta_{RATE} = \frac{1}{\sum_{n=1}^N \frac{1}{\theta_n}} \quad (54)$$

The steady state characteristics of the room component are calculated by its model on each time-step of the simulation. In a similar manner as used previously, the steady-state temperature is used in conjunction with Newton's cooling equation to calculate the new temperature for the next time-step as shown in (55). In this equation, t_{STEP} is the simulation time-step size in seconds.

$$T_{NEW} = T_{STEADY} + (T_{ROOM} - T_{STEADY}) \cdot \exp\left(\frac{-t_{step}}{\theta_{RATE} \cdot C_{ROOM}}\right) \quad (55)$$

In calculating the transient characteristics of the room, the heat capacity of the air in the room C_{ROOM} is required. The heat capacity of the air in the room is calculated by first calculating the mass of air in the room, m_{AIR} (kg) using (56) where ρ_{AIR} is the density of air and V is the volume of air in the room (m^3). This is then used in (57) to calculate the heat capacity of the air in the room C_{AIR} (J/K) using a constant value for the specific heat capacity of air $C_{P,AIR}$ (1012 J/kg K).

$$m_{AIR} = \rho_{AIR} \cdot V \quad (56)$$

$$C_{AIR} = C_{P,AIR} \cdot m_{AIR} \quad (57)$$

4.3.6. Static Room

While performing building model validation studies, it became apparent that it may in some cases be desirable to simulate the behaviour of a small part of a building to prove that the simulated behaviour matches experimental readings. To do this, the simulation package would need to be aware of the properties of the areas surrounding the areas being simulated

in order to produce results that are accurate to the experimental readings. In order to allow for this, the “Static Room” component was developed. The component acts as an area within the building that has a fixed temperature that is set based on values in a lookup table, rather than dynamically calculated. This allows the temperature of the room to follow experimental readings that have been imported into the package.

4.4. Heating and Ventilation

4.4.1. Natural Ventilation Component

Natural ventilation of a heated space within a building is one of the many factors which contribute to the loss of heat from a room. The European Standard EN 12831 [89] provides a method for calculating the heat loss through natural ventilation of a space. In this standard, heat loss is given by (58) where Φ is the heat loss (W), U_V is the ventilation heat loss coefficient (W/K), T_I is the internal temperature of the space (K) and T_E is the outside temperature (K).

$$\Phi = U_V(T_I - T_E) \quad (58)$$

The heat loss coefficient U_V is given by (59) where \dot{V} is the air flow rate of the heated space (m^3/s), ρ is the density of air (kg/m^3) and c_p is the specific heat capacity of air ($\text{kJ}/\text{kg}\cdot\text{K}$). Using the assumption of constant values for ρ and c_p , the standard simplifies (59) to (60), where \dot{V} is now expressed in m^3/h .

$$U_V = \dot{V} \cdot \rho \cdot c_p \quad (59)$$

$$U_V = 0.34 \cdot \dot{V} \quad (60)$$

The value of \dot{V} is taken as the maximum of \dot{V}_{INF} and \dot{V}_{MIN} where \dot{V}_{INF} is the air flow rate due to infiltration through the building fabric. \dot{V}_{MIN} is the minimum design air flow rate that is used for hygiene reasons within buildings and varies depending on the type of room. The infiltration air flow rate \dot{V}_{INF} is given by (61) where V_R is the volume of the room (m^3), n_{50} is the air exchange per hour that occurs due to a 50Pa difference between the inside and outside air pressure which is dependent on the air-tightness of the building, e_i is a shielding

coefficient which is dependent on the shielding provided by the building's surroundings and the number of exposed openings in the room and ϵ_i is a height correction factor which is dependent only on the building's height. Typical values of n_{50} , e_i and ϵ_i are given in the standard.

$$\dot{V}_{INF} = 2 \cdot V_R \cdot n_{50} \cdot e_i \cdot \epsilon_i \quad (61)$$

The hygiene air flow rate is given by (62) where V_R is the volume of the room (m^3) and n_{min} is the minimum number of air exchanges per hour required for hygiene reasons for that particular type of room. Typical values of n_{min} are given within the standard for different types of room.

$$\dot{V}_{MIN} = n_{min} \cdot V_R \quad (62)$$

The Natural Ventilation component that is provided within the simulation package has a number of user configurable parameters which allow is to carry out the calculations described above. These are: Volume of Room (m^3); Type of Room (Kitchen, Bathroom, Office, Meeting Room, Others); Building Air Tightness (0-100%); Building Shielding (Light, Medium, Heavy); Number of Exposed Openings in Room; Height of Room above Ground Level (m).

These configurable parameters are used in conjunction with the lookup tables provided in the standard document to determine the heat loss coefficient U_V for the room. The inverse of this value in (W/K) is then taken to produce a thermal resistance θ_V in K/W ($^{\circ}C/W$).

The component has two pins, one for a connection to the room and another for a connection to the ambient (outside) temperature. The model of the component behaves as a series resistance with value θ_V between the room temperature and outside temperature, modelling the resulting heat loss as described in (58).

4.4.2. Radiator

A basic model of a radiator for a water-based heating system was developed, primarily for a validation study that was carried out on the package using a home with a gas-fired heating system. The main requirement of this component was not to model the water temperatures

within the heating system but rather to model the heating effect that a radiator has on room temperature. A radiator manufacturer's catalogue [130] was used to obtain heat output specifications for different types of radiator. Within this catalogue, radiator heat outputs in watts were specified for a selection of different radiator constructions and sizes. Radiator sizes were specified in millimetres in terms of length and height. The different types of construction documented in the catalogue were: single panel; double panel with fins on one panel; double panel with two sets of fins; triple panel with three sets of fins. The heat output for each type of heater is specified in watts at a temperature difference (ΔT) of 60°C between the water temperature in the radiator and the room temperature. Correction factors are provided as fractions of the $\Delta T=60^{\circ}\text{C}$ heat output for other values of ΔT .

The radiator model that was implemented provides the list of radiators defined in the catalogue as a configurable "Radiator Type" drop down list parameter. A second parameter is provided to specify the initial temperature of the water within the radiator. In order to simulate dynamic adjustment of the heating system output throughout a simulation, the component supports messages using the asynchronous messaging system in the package in the format "TEMP X" where X is the average water temperature in $^{\circ}\text{C}$ across the radiator. This information can be imported into the package using physical measurements from a home.

The radiator has a single heat transfer connection to a room which has a temperature T ($^{\circ}\text{C}$) and thermal resistance R_{TH} ($^{\circ}\text{C}/\text{W}$). The temperature T is set to be the average water temperature in the radiator. The value of R_{TH} is calculated based on the radiator power output in the manufacturer's data lookup table.

The process of calculating R_{TH} begins by calculating the temperature difference between the radiator water and the room. This temperature difference is used to determine a correction factor for the $\Delta T=60^{\circ}\text{C}$ power output quoted by the manufacturer. This correction factor is determined by creating a piecewise linear model of the correction factor based on temperature using the manufacturer supplied correction factor data. Once the correction factor has been established, the radiator power output for the present ΔT value can be calculated by multiplying the $\Delta T=60^{\circ}\text{C}$ output by the correction factor. The thermal

resistance value R_{TH} is then calculated by dividing the temperature of the water in the radiator by the calculated power output for the given ΔT .

4.4.3. Electric Heater

An ideal electric heater component was implemented for use in models to study the smart control of heating appliances. This heating component has the following features: adjustable rating; 100% efficiency – all electrical input converted to heat output; configurable thermostat dead-band; and thermostat setting adjustable using communication messages.

The component has three connections – a heat transfer connection to the room being heated, an electrical connection to the home's mains supply and an optional communications link on which thermostat temperature setting messages can be received. The logic within the heater is programmed such that the heater operates with its element at 150°C when in its heating mode, providing its full output power when the room temperature is 0°C. The heat output of the heater decreases linearly with increasing room temperature. The electrical load resistance of the heater is determined by dividing the square of the heater's nominal voltage by the instantaneous heat output power. This simplified model assumes uniform element resistance over the entire operating temperature range of the heater.

The control algorithm built into the heater uses a combination of the heater thermostat setting values and dead-band value to determine whether the heating elements are switched on or off. If the room temperature falls below (Thermostat Setting - Dead-band / 2) then the heating element is turned on; if the room temperature rises above (Thermostat Setting + Dead-band / 2) then the heating element is turned off. The heater consumes no power and has no heat output when the element is turned off.

Basic communication support is built into the heater. When a message in the format "TEMP X" is received, the thermostat setting on the heater is set to the value X (°C). While this heating component is a relatively simplistic implementation of an electric heater, it is sufficient to allow for investigations into control over heating appliances.

4.5. Weather

4.5.1. Ambient Temperature Pattern

The Ambient Temperature Pattern component allows the ambient temperature used in a model to be varied during the course of a simulation. The component provides a configurable table parameter in which each row contains a time and a temperature to be set at the given time. In the case that a time falls between two simulation time-steps, the temperature value is set of the latter of the two time-steps – consistent with the behaviour of other components. Using this logic, it may be possible that more than one temperature entry exists for each time-step. In this case, the temperature point with the latest time value is used during that time-step.

Using this component, ambient temperature patterns can be set using historical or predicted weather data to perform simulations which are valid in the context of a particular area or can be set to specifically test the temperature response of a particular simulation component.

4.5.2. Random Ambient Temperature

In contrast to the Ambient Temperature Pattern component, the Random Ambient Temperature component uses an initial starting temperature and then varies the temperature randomly over the course of a simulation. A set of configurable properties define the way in which the temperature should be varied.

The user-specified properties that govern the random variation in temperature are the maximum allowed temperature (T_{MAX}), minimum allowed temperature (T_{MIN}) and the maximum variation in temperature (T_{MAXVAR}) on each time-step. Additionally, the user may also specify an integer seed value for the pseudo-random number generator used to ensure that the random temperature variation carried out is the same on each run of the simulation.

Given the user properties defined above and the previous ambient temperature T_{OLD} , the new ambient temperature T_{NEW} is generated using the algorithm shown in (63).

$$\begin{aligned}
&\text{Sign} = \text{Random selection from } [-1,1] \\
&\text{Random} = \text{Random decimal value between } 0 \text{ and } 1 \\
&T_{\text{NEW}} = T_{\text{OLD}} + \text{Sign} * (T_{\text{MAXVAR}} * \text{Random}) \\
&\text{If } T_{\text{NEW}} > T_{\text{MAX}} \text{ Then } T_{\text{NEW}} = T_{\text{MAX}} \\
&\text{If } T_{\text{NEW}} < T_{\text{MIN}} \text{ Then } T_{\text{NEW}} = T_{\text{MIN}}
\end{aligned}
\tag{63}$$

4.5.3. Solar Irradiation Pattern

The Solar Irradiation Pattern component is responsible for updating a global parameter within the simulation to a representative solar irradiation level in W/m^2 for the current season and time of day. The component contains a configurable look-up table of solar irradiation levels containing 24 rows representing each hour of the day. Within each row, there are twelve columns containing a representative solar irradiation value for that time of day on each month of the year. A second configurable parameter specifies the name of the global parameter that the component should store the current solar irradiation level in.

The component uses the simulated date and time during each time-step to determine the correct solar irradiation level within the lookup table and stores the selected value in the specified global parameter. This approach means that other components (for example a solar panel) which rely on the solar irradiation level can use the global parameter setting rather than each component having to use its own lookup table. Representative solar irradiation levels for various locations around the world are provided in the weather files distributed by the EnergyPlus project [131].

4.6. Other Components

4.6.1. Scheduled Asynchronous Messaging

In section 4.2.1, the Scheduled Data Generator Component was discussed. This component sends a message using the simulation package's built in data communications system based on a schedule table containing messages and the associated time at which each message should be sent. A similar component has been implemented to allow messages to be sent according to a schedule using the package's asynchronous messaging system which is designed to model user actions rather than data communications. The Scheduled Asynchronous Messaging component has no pins and instead provides a second configurable

parameter which specifies the instance name of the component that scheduled messages should be sent to.

4.6.2. List and Table Parameter Watch

The result recording mechanism within the simulation package, described in section 3.5.1, only allows for the recording of the values of scalar parameters within components. Occasionally, it may be more suitable to record run-time results in lists or tables within components and therefore it may also necessary to record certain values from lists or tables as a simulation runs for post-simulation analysis.

The List and Table Parameter watch component enables the recording of numeric values from lists or tables by reading values from specific elements of a list or table and storing them in scalar parameters. The built-in result recording mechanism can then be used to record the values of the scalar parameters and present them in graphs or tables. The component contains two configurable parameters which specify the instance name of the component containing the list or table to watch and the name of the list or table parameter within the component to watch. Ten read-only decimal parameters are provided within the component named "1-10" to which list or table values can be mapped. A configurable table parameter is also included within the component which contains 10 rows to store the mapping of list or table elements to the ten scalar parameters within the component. Each row has two columns in which the row and column number of up to 10 list or table elements to be watched are entered. In the case of lists, the column number is entered.

During the course of a simulation, the component examines the associated list or table to determine the value of the mapped elements and stores these values in the associated read-only parameter. If any of the mapped row or column indices are outwith the bounds of the list or table or do not contain numeric data then the value is recorded as NaN (not a number).

4.7. Summary

A selection of components including electrical appliances and infrastructure, building construction elements, heating appliances and communication systems has been presented in this chapter. While the set of components that has been developed is far from exhaustive, and many components have room for improvement in the detail of their models, the library provides a good starting point for the evaluation of the package. In the next chapter, the testing of the package and the validation of a number of the components that have been developed is described. This is followed by a case study in Chapter 6 which makes use of the library of components that has been implemented in this chapter.

Chapter 5

Testing & Validation

Testing and validation were an essential part of the development of the new domestic smart grid simulation package to ensure that the results obtained from the package were accurate and reliable. This chapter describes the testing that took place during the development of the package to test the correctness of the simulation results and functionality of the application. This is followed by the description of three experimental validation studies that took place in which rooms within buildings were modelled and physical measurements were taken of temperature and electrical power consumption within the rooms. These tests measured the performance of the simulator against real-world results.

5.1. Unit Testing

Throughout the development of the software, the Microsoft Visual Studio Unit Testing Framework was used to create unit tests. These unit tests check the functionality of the individual methods which make up each class in the software. Unit tests operate by providing test inputs to a method, executing the method and then checking that the outputs from the method match the output expected for the given test inputs.

Unit tests were extremely useful during the development of the software as they provide a method of constantly checking that changes to the implementation of the software have not broken its functionality. The unit test coverage of the software was estimated to be around 60% of all of the source code. This was due to a large part of the software's code base being made up of graphical user interface components for which test automation is difficult. To supplement the test-driven development approach, automated functional tests as well as manual inspections of the graphical user interface were undertaken.

5.2. Automated Functional Test Program

Automated functional tests of the software were performed to ensure that the simulation results produced by the completed package were theoretically correct. Testing of simulation results was not possible using the unit test framework as the component models used within simulations are loaded from XML files and compiled at run time. Therefore, in order to automate the testing of generated simulation results, a small test application was written in which test cases could be created and then run.

To enable the creation of an automated test suite for the application, a command-line version of the simulation package was created. The command-line tool, which is a cut-down version of the full simulation package, takes two input parameters. These are the file name of a system model which has been created using the full version of the software and the name of a CSV spreadsheet file in which to store the simulation results.

The automated test tool has a user interface to allow a test case to be specified which is shown in Figure 5-1. A test case consists of a system model file name and a set of expected result values to check against the simulation results. Each test case is saved in an XML file. Additionally, multiple test case filenames may be specified in a text file to run a batch of tests sequentially.

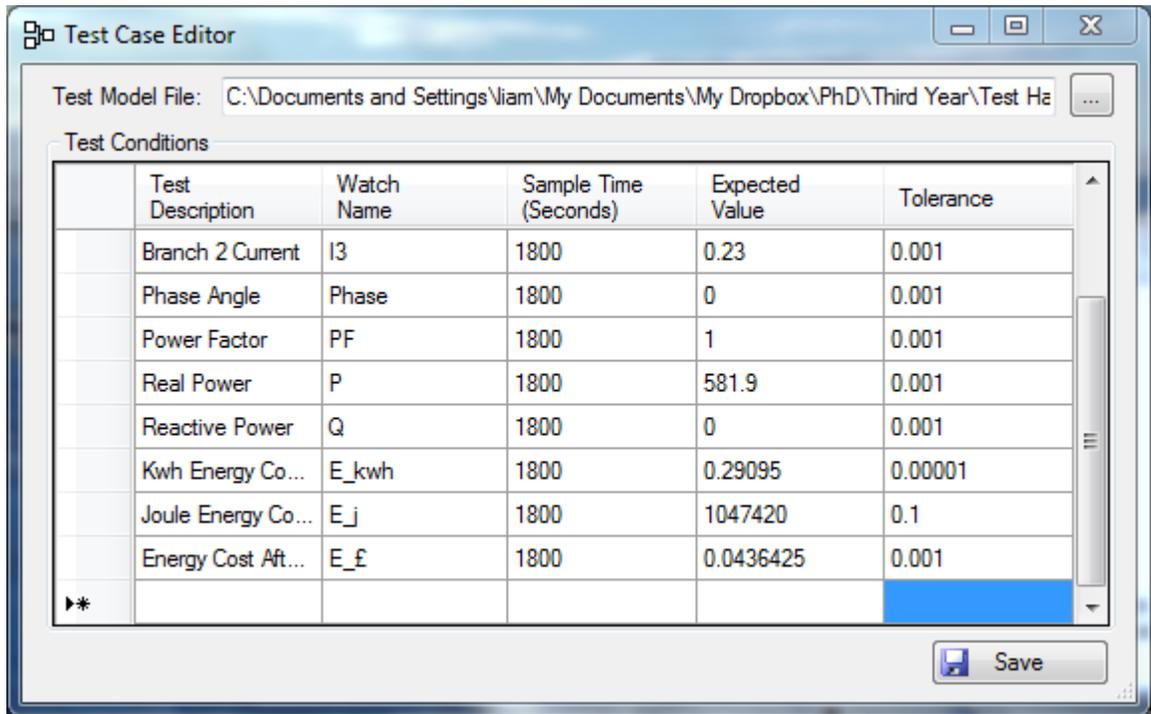


Figure 5-1: Automated test tool test case editor.

To run a test, the tool uses the command-line version of the simulation package to run the simulation defined in the test case. The results spreadsheet produced by the command-line simulator is then analysed to ensure that the results match the expected results entered in the test case. An example of a test run is shown in Figure 5-2.

The majority of the library components described in Chapter 4 were tested using the automated functional test program during development. The test program was also used to validate the electrical and temperature node components which were built into the package. During the software's development, on many occasions the test suite highlighted side-effects of code changes which caused the package to produce erroneous results. The stringent test process that was used during development provides a high level of confidence in the results that are produced by the package.

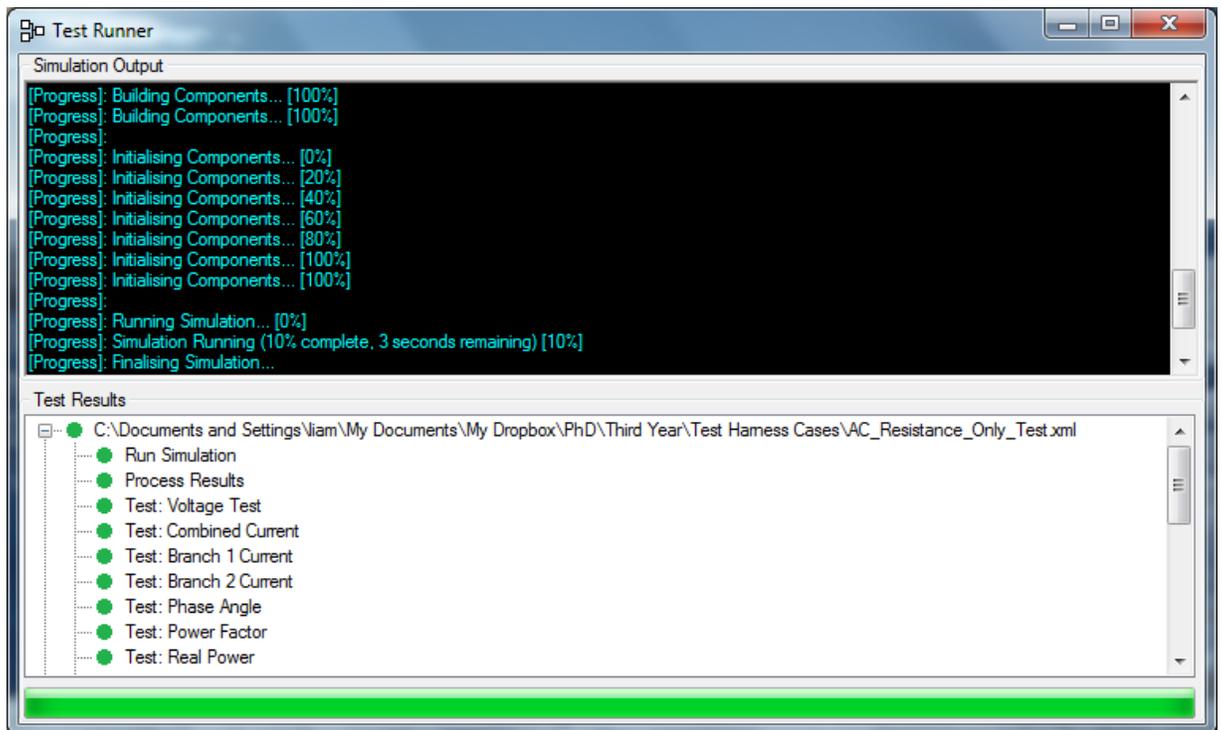


Figure 5-2: Automated test runner interface.

5.3. Graphical User Interface Testing

Automated testing of graphical user interfaces is problematic due to the volume of commands that may be obtained from a user when performing even the simplest of operations. For example, when using the graphical model editor, the same system model may be created multiple times within the software by performing a different sequence of actions each time. This is due to the different order of creation of model parts and also because the user may position components in different locations on the screen, despite the underlying logic model being identical.

Testing of the graphical user interface therefore took a more informal approach whereby when a new user interface feature was implemented or an existing feature modified, the application was run and the feature was manually tested. This approach was more useful in some scenarios than others. For example, dialogs which were used to edit simulation settings or parameter values were easy to test because data could be entered in the dialog and the dialog could then be closed and re-opened to confirm that the data entered were accepted

correctly by the dialog and saved to the underlying data store. Other more complex graphical user interface elements were also tested in this way – for example, the system model editing canvas and the graphical component editor. The system model editor was tested by creating a model, saving the model to a file and re-loading the model to ensure that the underlying representation of the model correctly represents the user’s input. While this approach generally worked adequately for these more complicated user interface components, bugs were occasionally discovered during the use of the package which were not immediately apparent during testing. The majority of these bugs were focussed around usability problems more than the ability of the package to produce models and obtain accurate simulation results.

It was of particular benefit from a testing perspective that the development of the modelling package was tightly coupled to the production of results for this project. This meant that the package was being actively used as it was developed. This allowed for first-hand experience of the types of bugs in software that are easily overlooked during a formal testing process but are picked up at the end-use stage.

5.4. Experimental Validation Studies – Thermal Models

5.4.1. Experimental Methodology

The main purpose of the thermal model experimental validation studies that were carried out was to verify that the room temperatures obtained from a simulated building model were correlated with the temperatures obtained through measurements of a modelled room. Three key aspects of the experimental method were therefore the measurement of room temperatures, the measurement of the behaviour of heating devices within the rooms and the accurate characterisation of the construction of the rooms and their surroundings for use in developing simulation models.

Within the simulation package, a well-stirred air model is used to model the air temperature within a room. This model makes the assumption that the air temperature within a room is completely uniform. In reality, this is not the case. Therefore, the experiments that were

carried out relied on recording average room temperatures. To provide a good representation of average room temperatures, multiple temperature readings were taken in order to record an average temperature of the hot (for example, near heating appliances) and cold (for example, near doors and windows) areas of the room.

In order to take a number of different temperature readings from the rooms while minimising disruption to the homeowners, *Lasca Electronics EL-USB-1* sensors were used. These sensors are small, battery powered units which can be connected to a USB port for configuration and then operate in a standalone mode for logging of temperatures. The sensors can then be reconnected over USB when the temperature logging is complete to download the data. The sensors have an accuracy of 1°C and a resolution of 0.5°C. The range of -35°C to 80°C makes them suitable for recording room temperatures. Prior to being used in the experiment, the correlation of the values read by different sensors was verified by recording data from co-located sensors for 24 hours at ten second intervals. The temperatures recorded were found to be within the manufacturer's quoted accuracy of 1°C. Twenty of these sensors were acquired for use in the thermal validation experiments.

Although utility software is provided with the *EL-USB-1* units to configure the logging and to download datasets, this software has the restriction of only being able to configure a single unit at a time. Therefore, a custom-built utility application was written for these experiments to allow multiple devices to be programmed with the same configuration simultaneously. The application also allows for the temperature data to be downloaded from multiple units simultaneously upon completion of an experiment.

The units provide the useful capability of assigning a textual label within the configuration parameters which is stored along with any data downloaded from the unit. This label was set to a unique identification number for each sensor, which was also physically marked on the sensor. When positioning sensors for data collection, the location of each sensor number was recorded to allow the downloaded data to be associated with the sensor's location for analysis.

The approach taken to record the behaviour of radiators within a room was to measure the temperature of the water inlet and outlet pipes of the radiator. An average water temperature could then be established from these two measurements to use as input to the radiator model within a simulation. Due to the higher temperatures involved and the need for a direct contact with the radiator pipes for measurement, thermocouples were deemed more suitable for obtaining these measurements than the EL-USB-1 loggers. J-Type thermocouples were fastened to the radiator pipes using cable ties and heat transfer compound was applied to the tip of each thermocouple to ensure good thermal conductivity between the pipe and the thermocouple junction. An Agilent data acquisition unit was used to log the thermocouple temperatures every minute. The data from this unit was saved onto a USB drive for analysis on a PC.

The characterisation of the construction of rooms for use in simulations was carried out using a survey process. This process involved obtaining the physical dimensions of the rooms, as well as a list of the materials used in the construction of the floors, ceilings, walls, doors and windows at the boundaries of the room.

In the first home that was studied, described in section 5.4.2, no original plans of the building were available and therefore measurements were taken to establish the geometry of the room. The materials used in the construction of the building were obtained through a visual survey, with the help of the homeowner who had detailed knowledge of the building's construction due to a recent refurbishment of the room being studied. In the second home, described in section 5.4.3, original plans were available describing the building geometry and the materials used in the construction. However, a visual survey of the doors, windows, flooring and decoration was carried out to complete the materials list. In both cases, *Google Sketchup* 3D models were created of the areas of interest within the buildings for reference use during the creation of the simulation models.

5.4.2. Test Room Experiment – Heat Loss Response

The first of the thermal model experimental validation studies carried out on the package was designed to assess how well the package's thermal modelling capabilities could model

the thermal behaviour of a room within a building without the presence of any forced heating or ventilation. This study defines the baseline error rate of the package which can then be taken into consideration when performing studies on heating and cooling systems.

The decision was taken to perform the study on a single room rather than a whole building: this allowed a room of the building to be chosen where the construction was well known and the geometry of the room and its surroundings were easily measurable. It also reduced the amount of measurement equipment that was required to gather data for the study.

The home used for this study was an early 20th century detached sandstone house. The home was chosen because it was in the process of being refurbished so the construction of the building was well-known. The dining room within the home was chosen for use in the study. During the visit to the home, detailed measurements were taken of the geometry of the selected room, doors and windows. A survey was also performed, with the assistance of the homeowner, of the materials used in the construction of the building around the selected room. This information was used to create an accurate 3D reference model of the room, shown in Figure 5-3, for use when creating the software model of the room. The 3D model also illustrates the areas bordering the room being measured. The results of the materials survey are shown in Table 5-1.

Table 5-1: Results of room materials survey for first test home.

Building Element	Material	Thickness (m)	Area (m²)
South Wall Doors	500kg/m ³ Timber	0.04	1.8146
East Wall Door	500kg/m ³ Timber	0.04	1.8990
Fireplace	Limestone	0.18	1.672
West Wall Behind Fireplace	Sandstone	0.3	1.672
	Air Gap	0.0508	
	Sandstone	0.3	
West Wall	1000kg/m ³ Plaster	0.0127	14.098
	450kg/m ³ Timber	0.0064	
	Sandstone	0.3	
	Air Gap	0.0508	
	Sandstone	0.3	
South Wall	1000kg/m ³ Plaster	0.0127	9.064
	Clay Brick	0.1145	
	1000kg/m ³ Plaster	0.0127	

Building Element	Material	Thickness (m)	Area (m ²)
East Wall	1000kg/m ³ Plaster	0.0127	13.855
	Clay Brick	0.1145	
	1000kg/m ³ Plaster	0.0127	
North Wall Around Windows	500kg/m ³ Timber	0.016	7.147
	Air Gap	0.0508	
	Sandstone	0.3	
	Air Gap	0.0508	
	Sandstone	0.3	
Remainder of North Wall	1000kg/m ³ Plaster	0.0127	6.585
	450kg/m ³ Timber	0.0064	
	Sandstone	0.3	
	Air Gap	0.0508	
	Sandstone	0.3	
Floor	Oak	0.022	24.108
	Pine	0.028	
Ceiling	1000kg/m ³ Plaster	0.0127	23.177
	Air Gap	0.045	
	450kg/m ³ Timber	0.015	
	Cinders	0.04	
	Air Gap	0.045	
	450kg/m ³ Timber	0.028	
	Underlay	0.005	
	Carpet	0.015	

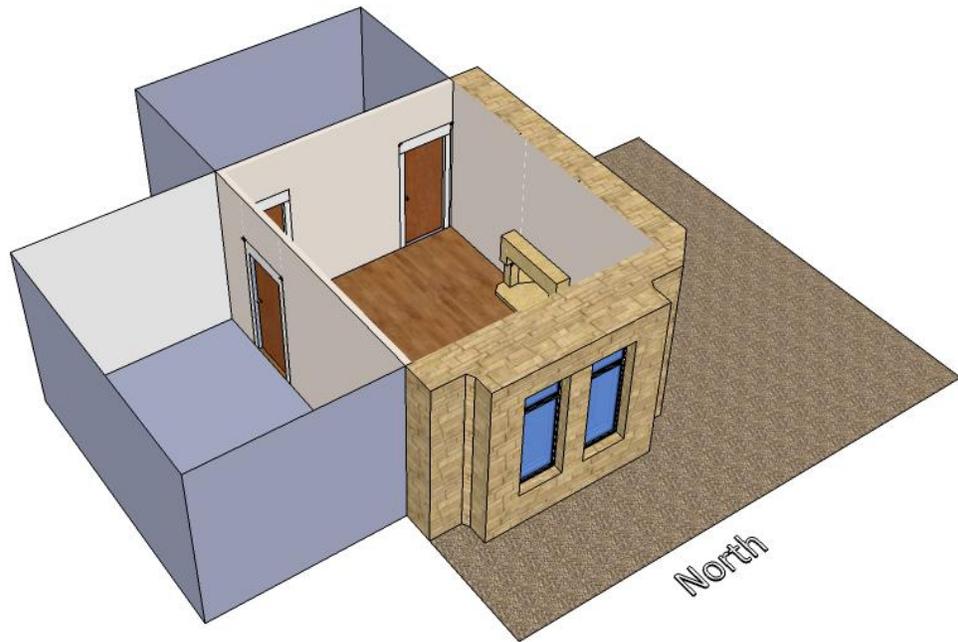


Figure 5-3: 3D Model of dining room used in case study showing neighbouring rooms.

Measurements of the thermal behaviour of the room were taken using the EL-USB-1 standalone temperature sensors, sampling once per minute. A number of sensors were placed at different points around the room to take an average temperature for the room. Sensors were also placed in the neighbouring rooms and the room on the floor above. A sensor was placed outside the building to measure the exterior temperature. These temperature sensors were allowed to run for five days in order to collect a suitable amount of information for use in the study.

A software model was created of the test room and its surroundings. This software model, an illustration of which is provided in Appendix B, was created by modelling the construction of the measured room and its surrounding floor, walls and ceilings using the library of components described in Chapter 4. Surrounding spaces were modelled using the “Static Room” component described in section 4.3.6 using a lookup table of temperatures measured during the experiment. The dining room was modelling using the dynamically evaluated room component, described in section 4.3.5.

The first experiment that was run involved simulating the thermal behaviour of the dining room over a 24 hour period within the 5 day recorded data period. The graph in Figure 5-4 illustrates the simulated room temperature compared with the measured value from the temperature sensors. Figure 5-5 illustrates the percentage error in the simulated temperature. During this simulation run, the average error in temperature was 1.8% and the maximum error was 3.7%. This shows that the simulated temperature is in agreement with the measured value.

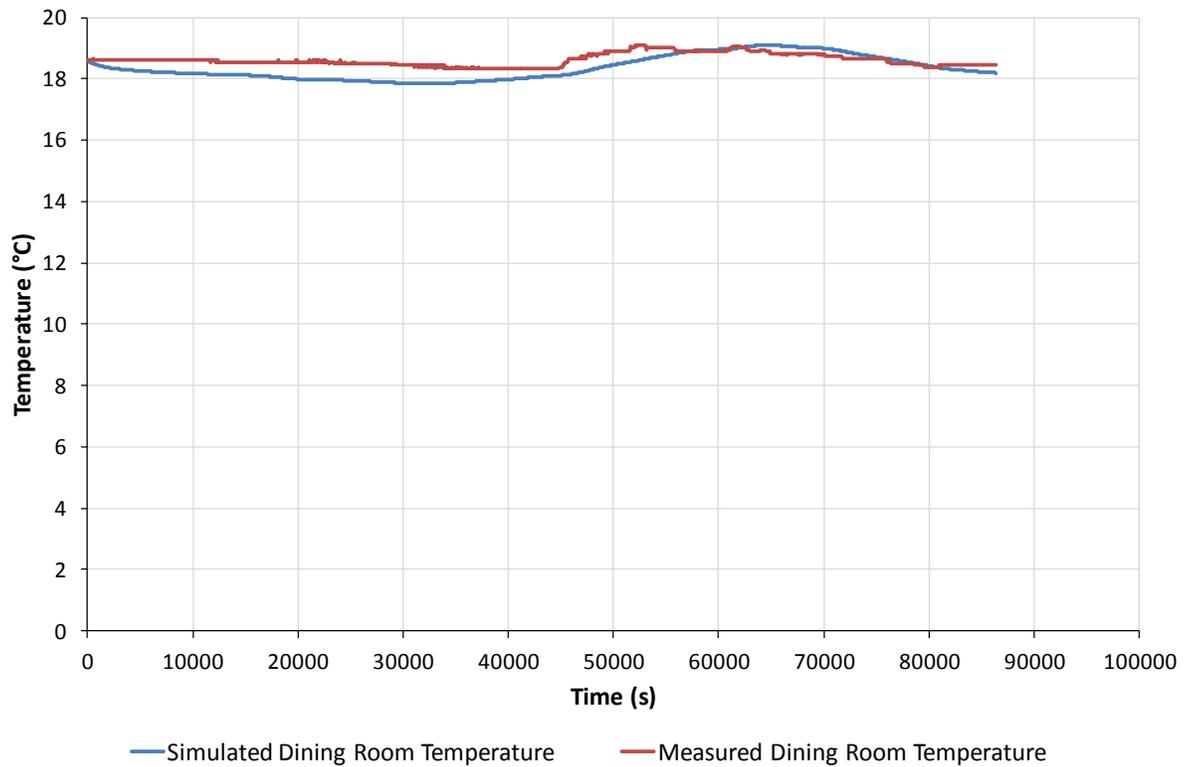


Figure 5-4: Results of 24 hour simulation of dining room temperature compared to measured room temperature.

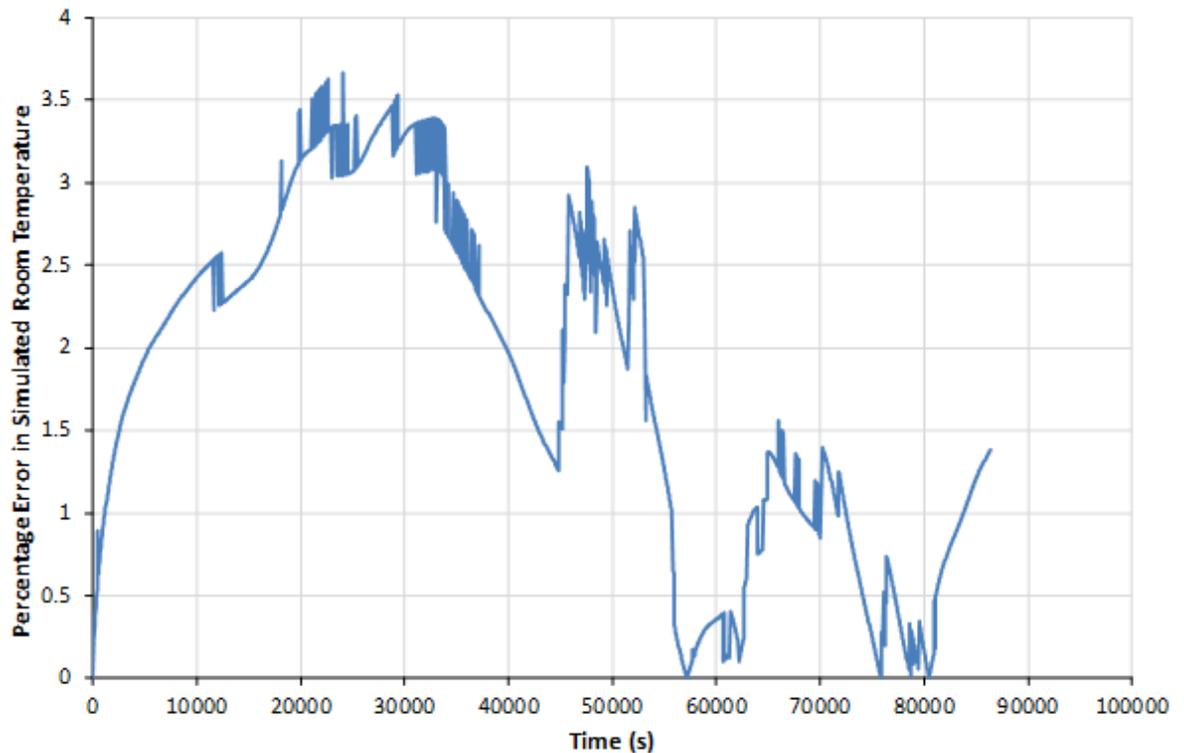


Figure 5-5: Percentage error in simulated room temperature during 24-hour study.

During the 24 hour study, the measured room temperature was relatively constant and therefore a second study was carried out on a more interesting period of the collected data where the temperature rose sharply for a short period of time due to incident sunlight on the north face of the building. In this second study, the thermal behaviour of the room was simulated for 48 hours. The simulated and measured temperatures are illustrated in Figure 5-6 and the percentage error in the simulated temperature is shown in Figure 5-7. The average error during this study was 2.3% and the maximum error was 17.2%.

The maximum error occurred during the brief period of sunlight which raised the room temperature significantly. While the simulation was able to follow the room temperature with good accuracy during the majority of the study, it was unable to model the increase in temperature caused by the sunlight. The main reason for the large error during this period is that the window model provided in the simulation package, described in section 4.3.4, does not model solar gain through the window, it only models heat conduction through the window. The future accuracy of thermal simulations of buildings with windows could be

improved by incorporating a model of incident sunlight on buildings in addition to a window model which supports heat transfer due to solar irradiation. Despite this inaccuracy in the model, the results of both the 24 hour and 48 hour studies show that the package is capable of producing thermal simulation results which agree well with measured experimental values. In both cases, the average error in the results was less than 5%.

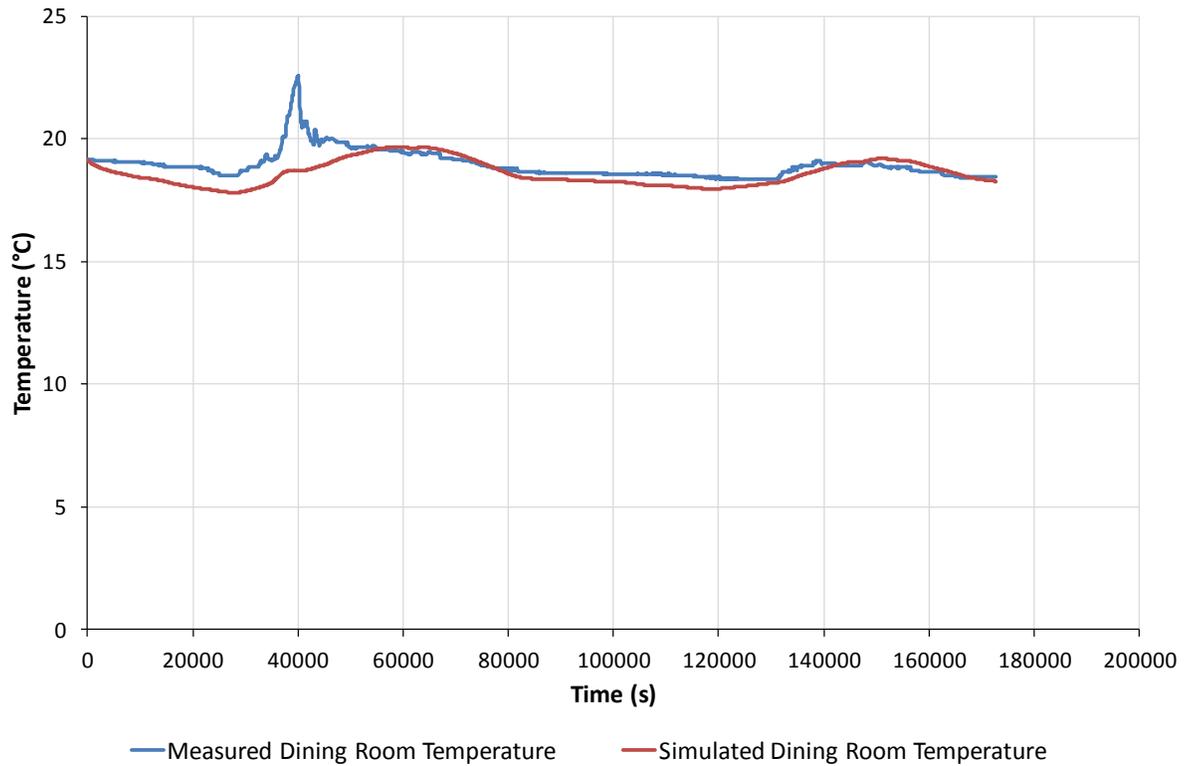


Figure 5-6: Results of 48 hour simulation of dining room temperature compared to measured temperature.

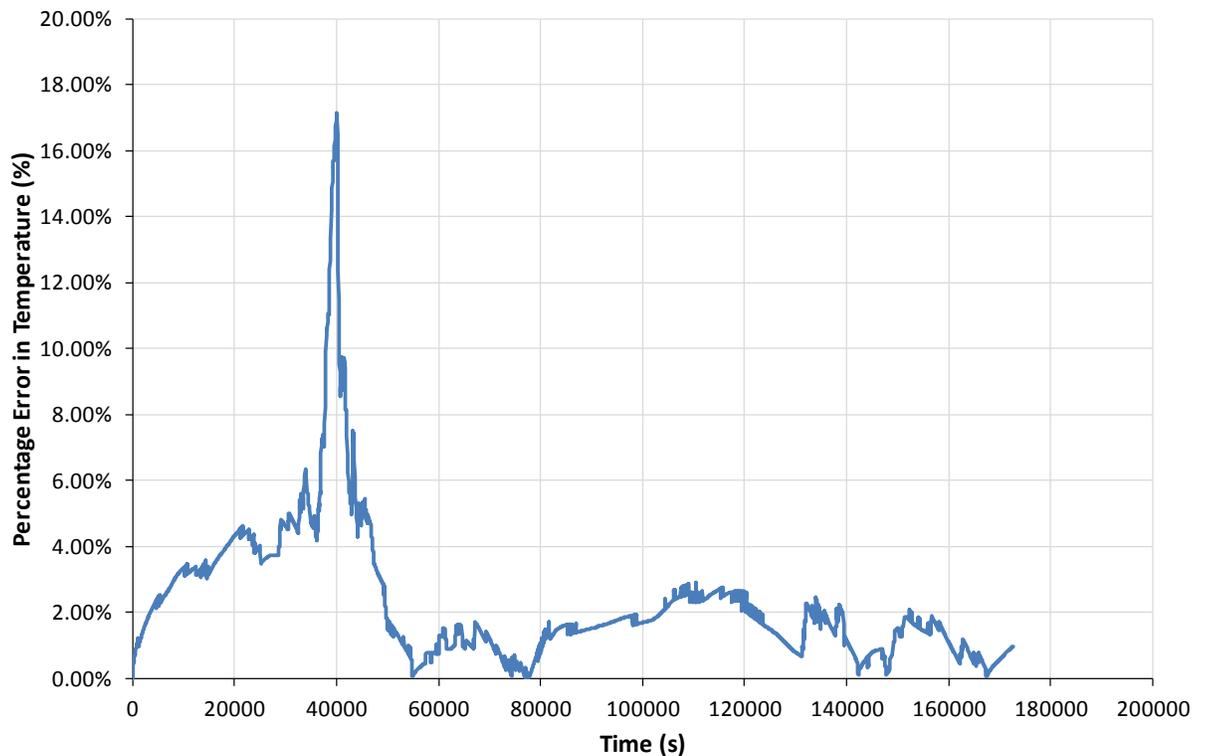


Figure 5-7: Percentage error in simulated temperature during the 48-hour study.

5.4.3. Test Room Experiment – Heated Room Response

A second validation study was carried out on the thermal modelling capabilities of the package. This time, as well as verifying the behaviour of the building element components provided in the package, the radiator component described in section 4.4.2 was also considered. In this case, a modern home (built around 1990) with gas-fired central heating was studied.

The development of a model of the home was made significantly easier in this case due to the availability of the original plans for the building. The 3D model shown in Figure 5-8 was developed from the floor plans in order to provide a reference model of the building geometry.

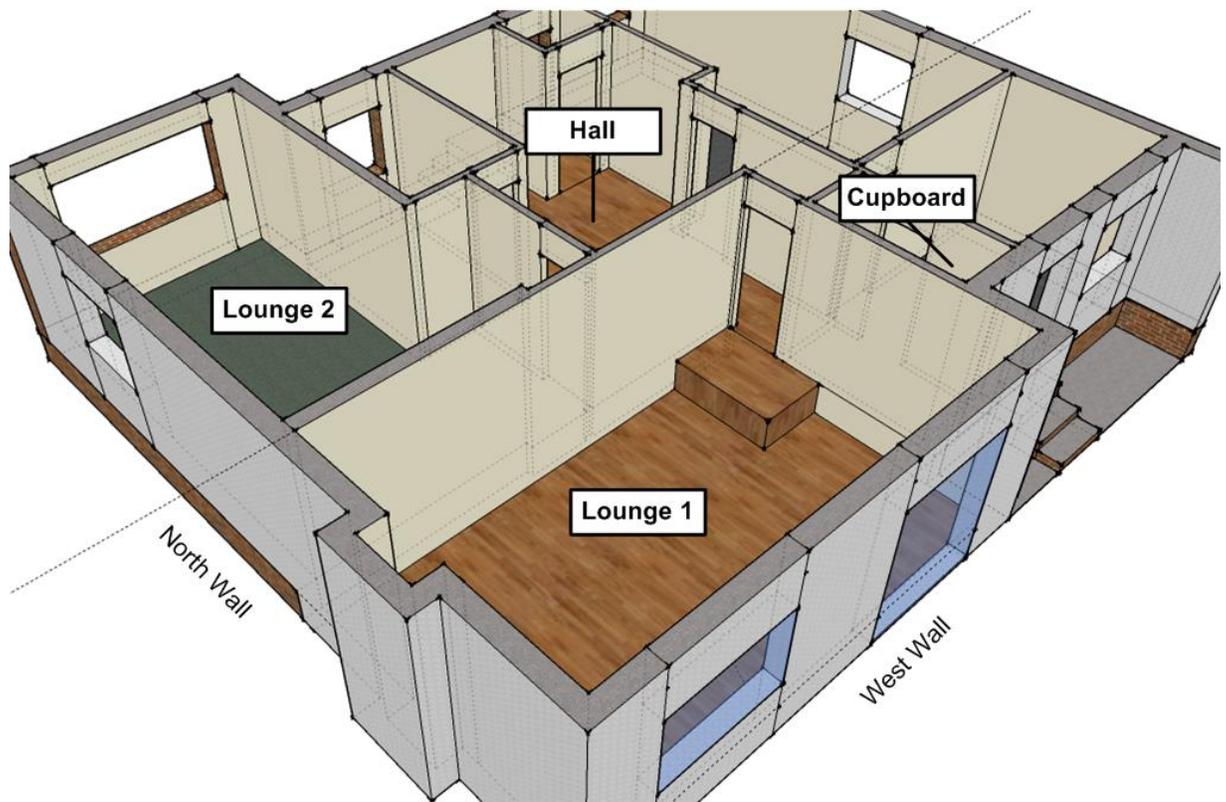


Figure 5-8: 3D model that was developed as a reference geometry model for the home used in the second case study.

The particular room that was studied in this case was the room marked as “Lounge 1” on the model in Figure 5-8. A materials survey of the construction of the building around the room was carried out, the results of which are shown in Table 5-2. It should be noted that in the selected room, the floor height is lower than the rest of the ground floor of the building, leading to a number of different material cross-sections making up each interior partition. Additionally, a portion of the wall between the lounges at the front and rear of the house is of external wall construction because “Lounge 2” is a garage conversion.

Table 5-2: Survey of the materials used in the building construction around the room marked "Lounge 1" in Figure 5-8.

Building Element	Material	Thickness (m)	Area (m²)
West Wall			
Exterior Wall	Cement Finish	0.015	13.756
	Concrete Block	0.1	
	Cavity	0.05	
	Insulation	0.075	
	Concrete Block	0.1	
	Plasterboard	0.015	
North Wall			
Exterior Wall	Cement Finish	0.015	11.745
	Concrete Block	0.1	
	Cavity	0.05	
	Insulation	0.075	
	Concrete Block	0.1	
	Plasterboard	0.015	
East Wall			
Internal Partition to Lounge 2 (Exterior Wall Standard)	Plasterboard	0.015	7.38
	Concrete Block	0.1	
	Insulation	0.075	
	Cavity	0.05	
	Concrete Block	0.1	
	Plasterboard	0.015	
Internal Partition to Lounge 2 (Interior Partition Standard)	Plasterboard	0.015	2.1936
	Insulation	0.025	
	Concrete Block	0.1	
	Insulation	0.025	
	Plasterboard	0.015	
Internal Partition to Hall	Plasterboard	0.015	5.76
	Insulation	0.025	
	Concrete Block	0.1	
	Insulation	0.025	
	Plasterboard	0.015	
Partition to Under-Floor Space	Plasterboard	0.015	3.328
	Insulation	0.025	
	Concrete Block	0.1	

Building Element	Material	Thickness (m)	Area (m ²)
South Wall			
Door	Solid Pine	0.035	1.509
Partition to Hall	Plasterboard Insulation Concrete Block Insulation Plasterboard	0.015 0.025 0.1 0.025 0.015	0.32
Partition to Cupboard	Plasterboard Insulation Concrete Block Insulation Plasterboard	0.015 0.025 0.1 0.025 0.015	5.04
Partition to Under-Floor Space	Plasterboard Insulation Concrete Block	0.015 0.025 0.1	1.65
Exterior Wall	Cement Finish Concrete Block Cavity Insulation Concrete Block Plasterboard	0.015 0.1 0.05 0.075 0.1 0.015	2.175
Floor			
Floor	Oak Flooring Underlay Particleboard	0.015 0.005 0.018	26.325
Ceiling			
Ceiling to Bedroom 1	Plasterboard Insulation Air Gap Particleboard Underlay Carpet	0.015 0.1 0.1 0.018 0.005 0.008	16.15
Ceiling to Bedroom 2	Plasterboard Insulation Air Gap Particleboard Underlay Carpet	0.015 0.1 0.1 0.018 0.005 0.008	9.72

The total volume of air in the room is 76.3795m³.

Experimental measurements of the room temperature, the temperature of surrounding rooms and the exterior temperature were taken over the course of 16 hours using the same portable data logging devices as in the previous experiment.

There are two radiators installed in the room – both double-panel, double-fin 900x600mm heaters with a rated output of 2.06kW at a temperature difference of 60°C. In order to record the approximate power output of these radiators for use in the model, thermocouples were attached to the water inlet and outlet pipes and temperature data was logged each second using an Agilent data acquisition unit.

Upon completion of the experimental phase of the study, a model of the room was created in a similar manner to the previous experiment, modelling the surrounding rooms using the “static room” component which uses the temperature data collected in the experiment. The exterior temperature is modelled to follow the temperature data collected in the experimental phase of this study. The heat output of the radiators in the room is simulated using the *Radiator* component described in section 4.4.2, with the average of the measured radiator inlet and outlet temperatures being used for the radiator water temperature in the component. The full model of the room is illustrated in Appendix C.2.

Figure 5-9 below illustrates the simulated room temperature compared to the actual room temperature measured in the experimental phase of the study. Figure 5-10 illustrates the percentage error in the simulated temperature compared to the measured temperature over the course of the simulation.

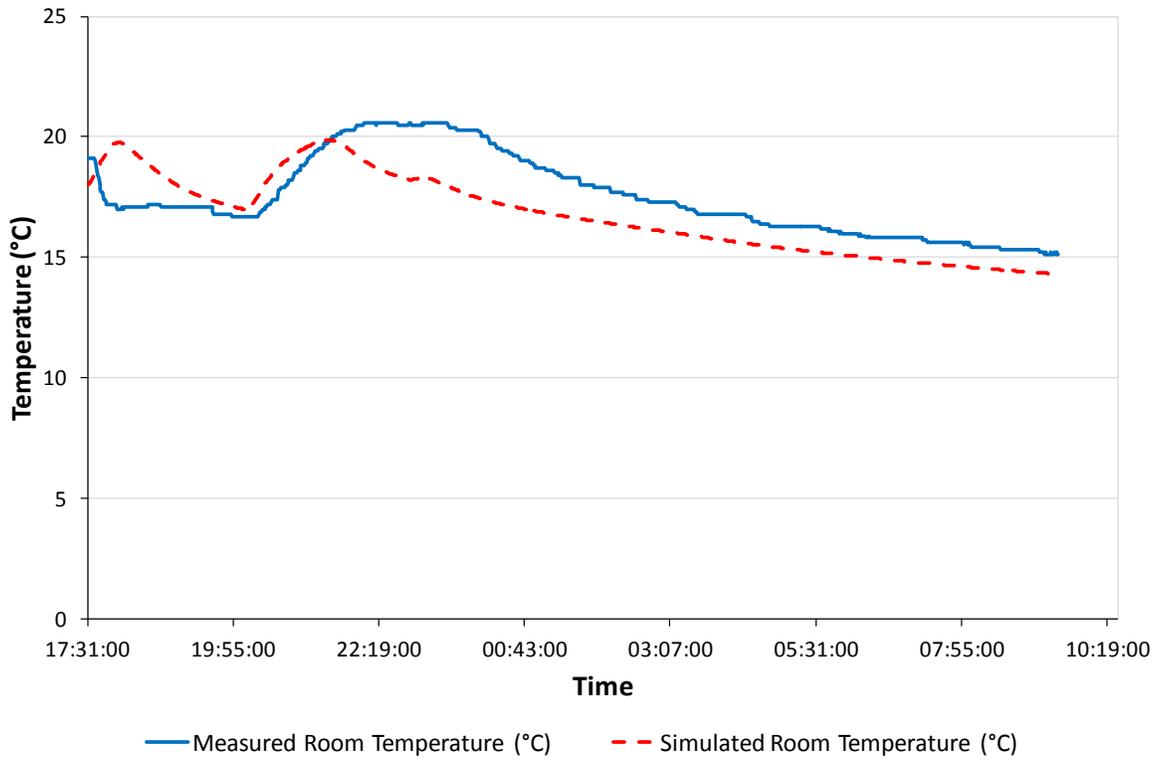


Figure 5-9: Comparison of measured and simulated room temperature in the second case study.

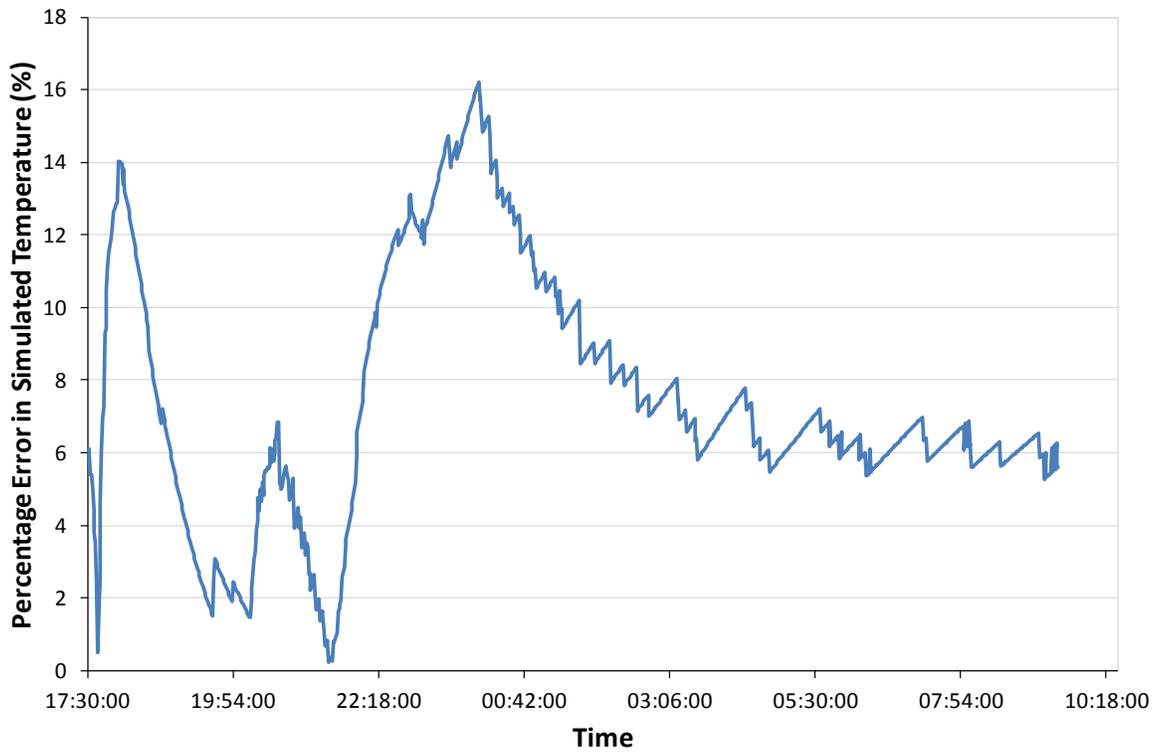


Figure 5-10: Percentage error in simulated temperature in the second case study.

The results in Figure 5-9 and Figure 5-10 show that there is a greater error in the simulated values for this case study than in the first case study that did not involve room heating. The maximum error encountered during the simulation was 16.2%, with an average error of 7.5% - around double the error in the previous case study.

The larger error is related to an inaccuracy in the radiator model from section 4.4.2 that was used in the simulations, along with errors introduced by the water temperature data that was recorded from the home and used in the simulated radiator models. The largest error contribution was introduced by water temperature measurement used in the experimental phase of the study. Water temperature measurements were taken using thermocouples attached to the inlet and outlet pipes on the radiators. These measurements can provide an accurate reflection of the water temperature in the radiator under steady-state conditions when the heating system has been running for some time. However, they are not an accurate representation of the water temperature in the radiator under the following conditions: when the radiator is initially cold and water begins to flow in the heating system; when the thermostatic valves close on the radiator and water stops flowing through the heater; and when the heating system pump stops and water stops flowing.

The contribution of this error can be observed at both of the peaks in the simulated room temperature in Figure 5-9. At the first peak, the measured water temperature from the radiators has momentarily risen due to water flowing in the heating system. In the simulated data this has caused the room temperature to momentarily heat up. However, in the measured data, it can be observed that despite the water temperature in the heating system rising, the system was not running long enough for the radiator's water temperature to rise to a level that caused the room to begin heating up significantly. At the second peak in the simulated room temperature, the simulated room temperature is shown to rise more rapidly than the measured temperature. Again, this can be attributed to the temperature in the radiator inlet and outlet pipes rising and falling more rapidly than the overall temperature of the water within the radiator and hence the simulation failing to accurately reflect the time taken for the radiator to heat up and the time taken for the radiator to cool down once the heating system is switched off (or the thermostatic valve closes).

Despite the error introduced by the inaccuracies in the modelling of radiators in this particular model, it can be observed that the general trend of the simulated and measured temperatures shown in Figure 5-9 is correlated and that improvement of the radiator modelling would enhance the accuracy of the results obtained.

5.5. Experimental Validation Study – Electrical Circuit

A short validation study was carried out on the electrical modelling capabilities of the package. The two main objectives of this validation study were to verify the accuracy of the representative models of household electrical appliances that were described in section 4.1.5 and to verify the accuracy of a simulated electrical circuit involving a collection of household appliances.

A set of household appliances – an LCD TV, a BT Vision Set-Top Box and a lamp with a CFL bulb – were connected to the domestic mains supply through a power analyser unit (described in Appendix D). The power analyser unit provided measurements of current, power, power factor and phase angle for each connected appliance. A measurement of the mains voltage was provided in addition to overall readings for the power and power factor of all of the connected appliances combined. The power analyser was connected to a laptop computer which logged measurements at one second intervals between 17:52 and 22:28 of a typical evening's usage of the appliances. As a supplement to this automated logging of the physical measurements, a diary was kept of the times that the television was in the "On" and "Standby" modes for use in the simulation of the system.

A simulation model equivalent to the experimental setup was developed, a schematic diagram of which is provided in Appendix C.3. The times logged in the TV use diary were set in a scheduler component within the model so that the modelled system matched the behaviour of the real system. Additionally, the results obtained during the physical measurement phase of the validation experiment were imported into the software package so that they could be displayed with the simulated results inside the package for comparison.

Figure 5-11 illustrates the comparison of the measured power of the appliances with the simulated values of each appliance’s measured power. Upon initial visual inspection of this graph, it appears that the modelling package has produced results that model the real behaviour of the appliances with acceptable accuracy. However, upon analysis of the errors within the measured power values, it became apparent that the BT Vision Box and TV had an average error of 9.8% and 9.2% respectively in the simulated value when compared to the measured value. The electric lamp had a lower error at only 2%. The error in these values was attributed to the fact that each appliance was modelled as a constant impedance, rather than an impedance which dynamically varied. Such a model has shown to be acceptable for the electric lamp – a relatively simple appliance in which the power consumption rarely varies. However, the BT Vision Box and TV are more complex appliances in which the power consumption dynamically varies throughout their use to a larger extent and therefore the lumped impedance model provides a poorer fit to the dynamic behaviour of the appliance.

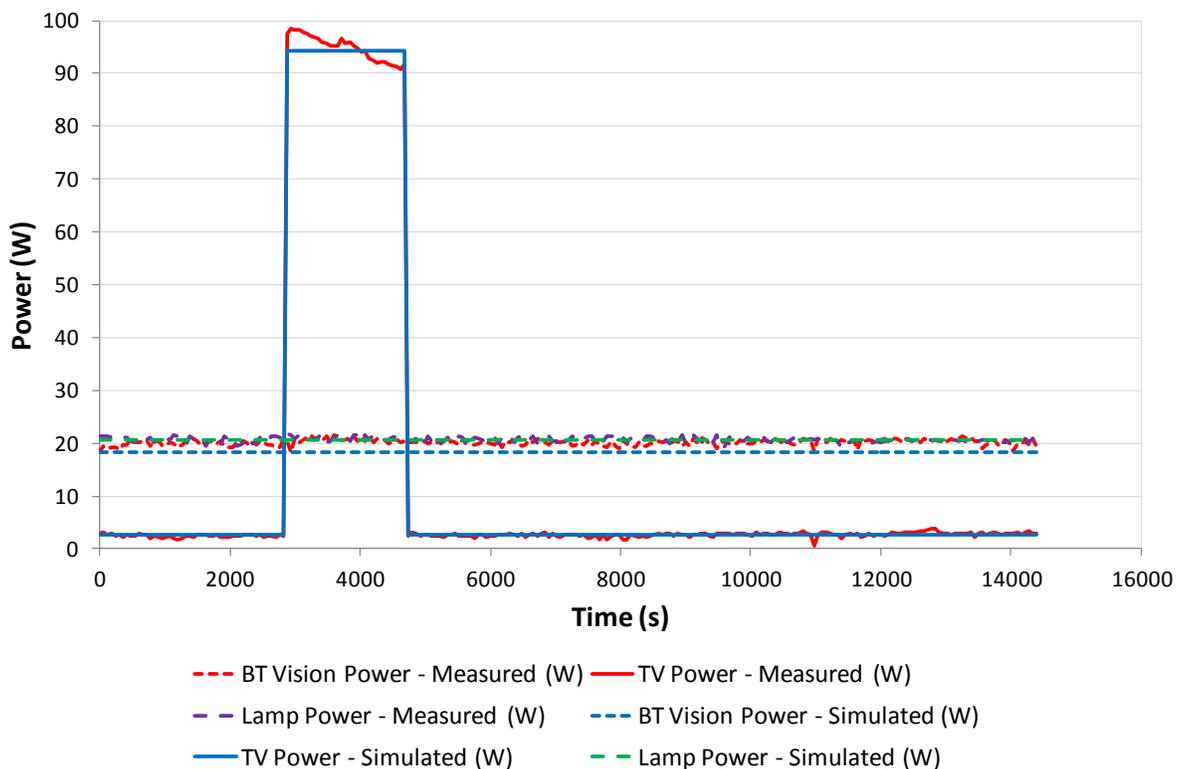


Figure 5-11: Comparison of physical measurements of domestic appliance power consumption with simulated power consumption.

In addition to comparing the individual appliance power consumption, a comparison was also carried out between the total measured power of the electrical circuit formed by the three appliances compared to the simulated value of overall power consumption. Again, upon visual inspection of Figure 5-12, the simulated and measured power consumption appears to be well correlated. After an analysis of the error in the values, the average error between the simulated and measured value of total power consumption was 4.16%. It can be concluded from this result that while constant impedance models of domestic appliances may be less desirable for certain types of appliance, the larger errors observed when examining a home model in detail become less significant when considering the power consumption of the home as a whole.

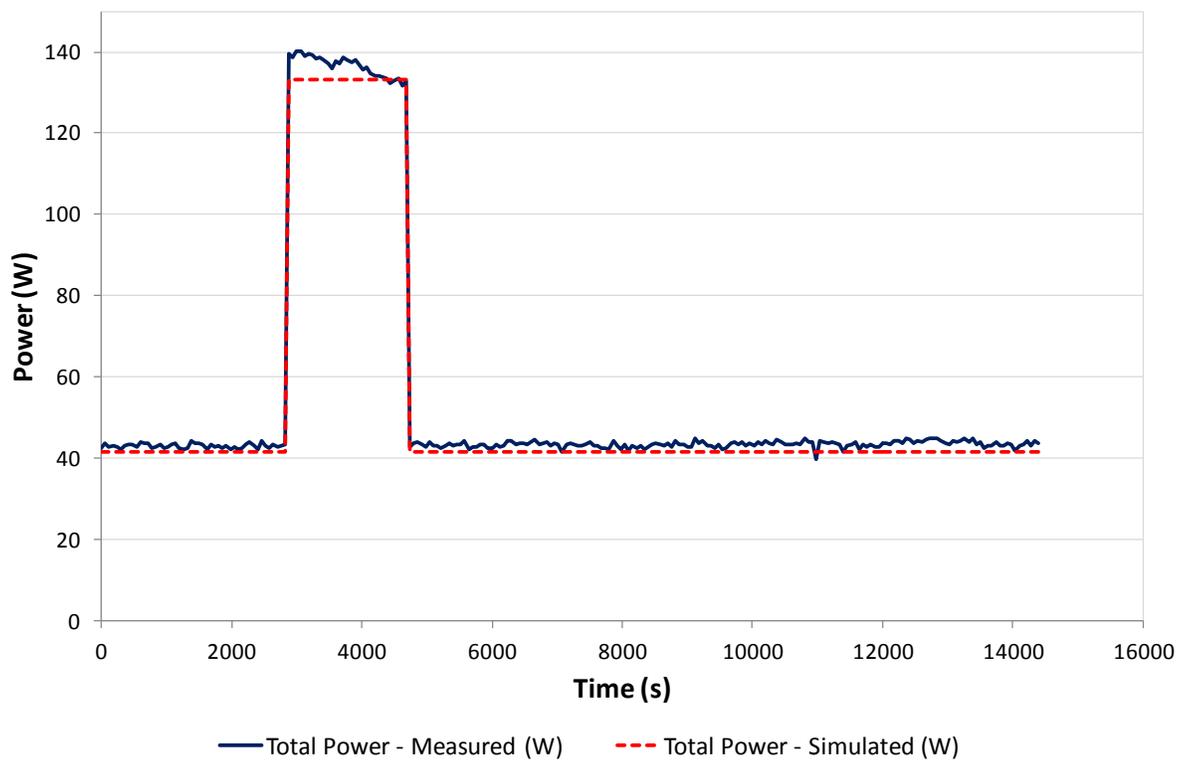


Figure 5-12: Comparison between the simulated and measured values of the total power consumption of all appliances.

The simulated and measured values of the difference in phase angle between voltage and current for each appliance were also compared in this validation experiment. The simulation of these values was considered more successful than the total power consumption as

average error in phase angle for the BT Vision Box, TV and Lamp were 1.78%, 0.5% and 5% respectively. These lower errors were attributed to the fact that although the power consumption of each appliance varied during its operation unlike the constant power model used in the simulations, the phase angle of each appliance remained relatively constant throughout the operation.

Dynamic variation in the power consumption of devices that is not accurately reflected in the simulation models has already been described as one source of error in this validation experiment. However, for completeness it is important to consider the other sources of error in the experiment. The experimental apparatus used to obtain the physical measurements for this comparison consisted of a custom-made power analyser unit which used a microcontroller in conjunction with off-the-shelf sensors to obtain voltage, current and phase angle measurements and used these measurements to calculate power and power factor. Taking into account the quoted errors of the sensor and amplifier components used in the system, along with the quantisation error in the microcontroller's ADC, the error in this unit is estimated to be no more than 7%. Electrical noise should also be considered as a potential source of error in the measurements when using this apparatus: the unit lacks the shielding provided in professional-grade measurement equipment.

The RMS voltage of the electrical grid connection used in the simulation model is a source of error for the comparison. The simulated voltage was fixed at 233 – the average value of the measured voltage. The actual measured voltage fluctuated between 231V and 240V throughout the experiment. The average error in the simulated voltage compared to the measured value was 1.4%.

5.6. Summary of Testing and Validation

Three validation experiments that were carried out on the simulation package have been presented in this chapter. These experiments have shown good agreement between the simulated results and the actual measured values with varying but small absolute errors.

In the first experiment – the analysis of the thermal properties of a room with no forced heating or cooling system – the simulated results were shown to agree well with the experimental values. In the first comparison, the average error in the simulated results was 1.8% with a maximum error of 3.7%. However, in a second comparison during this experiment, a weakness was found in the thermal modelling components' lack of ability to model heat gain due to solar irradiation. This produced a transient error of 17.2% in the simulated temperature value during a brief period of intense sunlight through the window at the front of the room. However, the average error during this experiment was still low at 1.8%. These results show that while there are some shortcomings in the implementation of the surface and window models provided in the package, it provides a good representative model of a room's thermal properties.

In the second experiment – the analysis of the thermal properties of a room with forced heating – the simulated results differ significantly from the measured room temperature values. The average error in the simulated results was 7.5% with a maximum error of 16.2%. However, the source of this error was identified as a problem with the collection of radiator temperature data which resulted in the simulation under-estimating the time taken for water to heat up and cool down within the radiator itself. If the periods when the radiators were in operation are excluded from this study then the maximum error in the simulated temperature falls to around 7%. This illustrates that an improvement of the modelling of the radiators in the room would yield significantly improved rates of error in the simulation. The incorporation of flow rate sensors in the experimental measurements would have also contributed to a more accurate set of inputs to the validation model. However, in an existing domestic system, these sensors cannot be easily fitted on a temporary basis and therefore the inlet and outlet pipe temperature measurements were averaged as an approximation to the water temperature within the radiator.

In the third experiment – the comparison of simulated domestic electrical appliances to measured values from real appliances – a number of deficiencies in the electrical models provided in the package were highlighted. The initial assumption was made during the development of the electrical models that appliances have a constant power output

throughout their operation while experimental measurements of the appliances have shown that this is not the case. This initial assumption resulted in average errors of nearly 10% in some of the individual appliance measurements. However, when considering the total power consumption of all appliances together the individual errors become less significant and an average error of around 4% was achieved. In the appliances considered, phase angle was shown to vary less than power consumption and therefore the constant impedance models provided a maximum average error of 5% in an individual appliance's phase angle. These results illustrate that the simulation package provides good accuracy for the simulation of the power consumption of a home as a whole, but further work would be required to refine appliance models if a more detailed analysis of individual parts of a home's electrical system is to be undertaken.

The validation results presented in this chapter show that the new simulation package provides good accuracy in its initial results, with a number of the comparisons yielding an error of 5% or less. Areas for enhanced accuracy of the modelling elements included in the package have also been identified. These initial results establish the necessary confidence level that the package is able to provide sufficiently accurate results for the detailed simulation studies which are presented in Chapter 6.

Chapter 6

Case Study

Chapters 3 - 5 describe the design, implementation and testing of a new software package for the modelling of intelligent domestic energy control systems. The package provides the benefit of allowing for the modelling of the existing heat and electrical energy systems within a home and allowing for the prototyping of new smart energy control systems. This case study illustrates one use case of the software – the modelling of an integrated smart home control system.

6.1. Home Energy System Overview

Chapter 2 discussed how electricity networks are currently being upgraded to permit more renewable methods of generation, including microgeneration devices in domestic and business premises.

One of the associated problems with the increased deployment of renewable generators throughout electricity networks is the fluctuating output provided by wind or solar generators. These generators may produce surplus electricity when it is not required by end-users and conversely may also not produce enough electricity at the peak times when it is most needed. Demand side management and energy storage were identified as two methods which could be used at the domestic level in order to allow consumption to more closely match generation.

The software system presented in this case study combines a number of elements of research work carried out during this project, most notably the development of a maximum power point tracking algorithm for small-scale photovoltaic systems [2] and the adaption of a smaller case study involving controllable loads and critical-peak-pricing tariffs [1]. The

system, illustrated in Figure 6-1, includes a roof-mounted photovoltaic panel, a battery storage system and a hot water storage system.

The photovoltaic system uses a new maximum power point tracking algorithm developed to maximise the power output of the PV panel. The battery can be used to store energy from either the microgeneration system or directly from the grid. An electric water heater provides a thermal energy storage option for the system whereby water can be heated during off-peak times and used when required within the home. All other uncontrollable electrical load within the home is modelled as a time-varying lumped load component. The smart domestic energy controller within the system is capable of reading the energy consumption data from the smart meter and using these data to issue control actions to the in-home appliances.

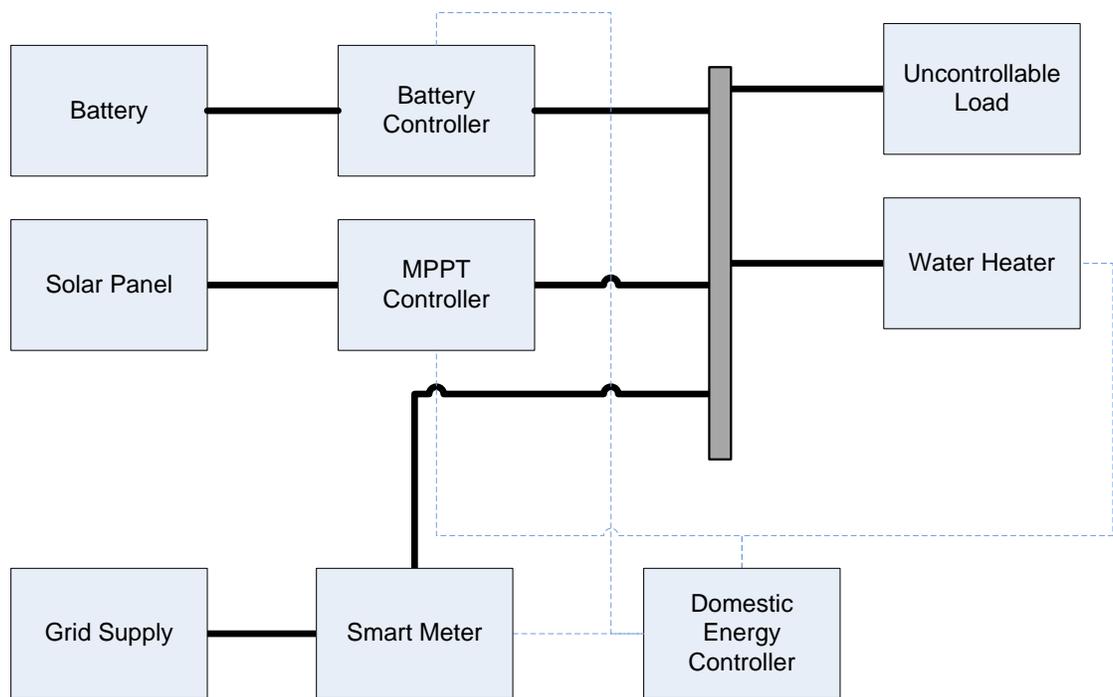


Figure 6-1: Domestic energy system modelled in this case study. Solid lines indicate electrical connections while dashed lines indicate communication links.

6.2. Maximum Power Point Tracking

6.2.1. Background

In section 4.1.2, the implementation of a generic solar panel model for use in simulations was described. In the discussion of this model, the non-linear relationship between voltage and current in a solar panel (64) was described. An example of the current-voltage relationship for varying solar irradiation levels is shown in Figure 6-2. On this illustration, the point on each curve that yields the maximum power output from the solar panel is indicated. This point is known as the maximum power point.

$$I(S, T, V) = I_{PH}(S) - I_{RS}(T) \left(\exp\left(\frac{V}{m \cdot V_T(T)}\right) - 1 \right) \quad (64)$$

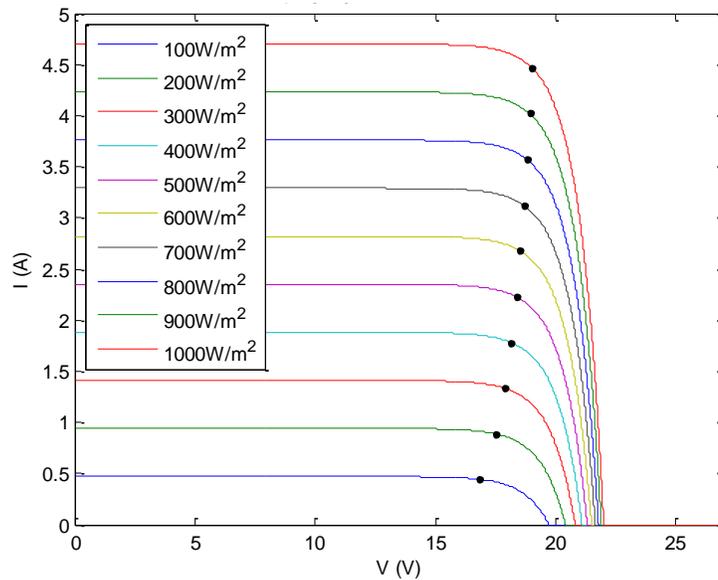


Figure 6-2: Illustration of the current-voltage relationship of a solar panel for varying solar irradiation levels. Maximum power points are indicated on each curve.

In order to obtain the maximum power output from a solar panel, the load resistance R_{LOAD} (Ω) has to be carefully chosen to match the case described in (65) where V_{MPPT} (V) and I_{MPPT} (A) are the respective voltage and current at the solar panel's maximum power point.

$$R_{LOAD} = \frac{V_{MPPT}}{I_{MPPT}} \quad (65)$$

In reality, a solar panel connected directly to the electrical load that it is supplying is unlikely to yield its maximum power output. In order to address this issue, power converters with maximum power point tracking systems are commonly used to interface photovoltaic generation systems to the electrical grid. These algorithms use switched mode power supplies to constantly vary the load on the solar panel in an attempt to yield more power from the panel – in effect matching the output impedance of the panel to the input impedance presented by the converter and thereby satisfying the maximum power transfer theorem requirement for matched source and load impedance.

Two broad classes of maximum power point tracking algorithm are available: model-based approaches, and perturbation and observation approaches. Model-based approaches use the model of the solar cell from (64) to accurately calculate and set the maximum power point. A model-based approach is presented in [20] which uses manufacturer-supplied data in addition to measurements of the solar irradiation and cell temperature to calculate the maximum power point. Model-based approaches offer the benefit of being able to very accurately track the maximum power point regardless of how quickly irradiation changes but are highly dependent on the particular solar panel in use and require extra hardware to measure irradiation and temperature.

Perturbation and observation based algorithms provide a more generic approach to maximum power point tracking and rely only on measured voltage and current as inputs to the algorithm. The most basic form of perturbation and observation algorithm operates by making an adjustment to the operating voltage of a photovoltaic system⁹ and observing whether this yields an increase or decrease in the power output of the system. If a decrease is observed, the algorithm adjusts the voltage in the opposite direction. If an increase is observed, then the algorithm continues to make voltage adjustments in the same direction. The overall effect of the algorithm is that the output power of the panel converges to close to the maximum power point and oscillates around the maximum point when it is found.

⁹ In practice, changes to the operating voltage of a photovoltaic system are made by adjusting the load on the solar panel through the use of an electronically-controlled switched mode power supply.

The hardware required to implement this type of algorithm consists of voltage and current transducers, a switched mode power supply to vary the load on the panel and an electronic control system such as a PLC, FPGA or microcontroller on which to implement the algorithm. Optimisations such as varying the perturbation size [24] or varying the sampling speed [25] are trivial to implement, requiring only a software change on the control system.

The Incremental Conductance algorithm [23] is an improvement on the basic perturbation and observation algorithm and reduces power losses due to oscillation around the maximum power point and tracking in the wrong direction during rapid changes in atmospheric conditions. These improvements stem from the use of a technique which controls the perturbation direction based on the photovoltaic system's power-voltage curve.

The maximum power point represents a peak in the power-voltage curve of a solar panel, hence at this point $dP/dV = 0$. At any point on the curve to the left of the maximum, $dP/dV > 0$. To the right of the maximum, $dP/dV < 0$. Determining the sign of the slope represented by dP/dV allows a decision to be made on how to adjust the voltage. The voltage should be increased if $dP/dV > 0$ and decreased if $dP/dV < 0$, to move closer to the maximum power point. If $dP/dV = 0$ then no change is required: the photovoltaic system is operating at its maximum power point.

In order to compute dP/dV using only voltage and current measurements, equation (66) can be used [23]. Voltage and current measurements can be taken directly from transducers while approximations for dV and dI can be made using (67) and (68) respectively. V_N and I_N are the voltage and current readings taken during an iteration of the algorithm and V_B and I_B are the voltage and current readings taken on the previous iteration of the algorithm.

$$\frac{dP}{dV} = \frac{d(I \cdot V)}{dV} = I + \frac{V \cdot dI}{dV} \quad (66)$$

$$dV = (V_N - V_B) \quad (67)$$

$$dI = (I_N - I_B) \quad (68)$$

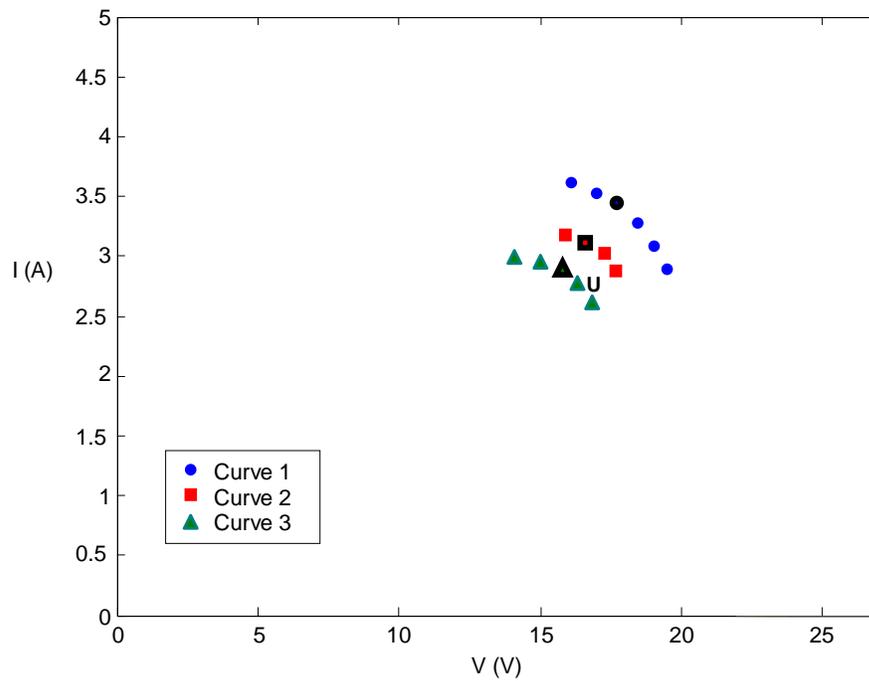
The addition of the curve-tracking technique ensures that the maximum power point is tracked correctly, even under rapidly changing atmospheric conditions. Additionally, by performing no voltage adjustments when the maximum power point is reached, losses caused by oscillating around the maximum are eliminated. The hardware requirements of this algorithm are the same as those of the basic perturbation and observation algorithm and this makes it an attractive replacement. Results from the literature ([23], [132]) have shown that this algorithm provides significant improvements in power yield over the basic perturbation and observation algorithm.

6.2.2. Newly Developed Algorithm

While the Incremental Conductance algorithm addresses some of the shortcomings of basic perturbation and observation algorithms, a particular situation in which it continues to offer reduced efficiency is in its tracking stage when the operating point is moving between two significantly different maximum power points (e.g. during partial cloud cover). Perturbation and observation algorithms, including the incremental conductance algorithm are limited in their tracking speed as a consequence of the fixed-size adjustments to the operating voltage on each iteration. This limitation provided the motivation for the development of a new algorithm to improve the tracking speed of perturbation and observation based algorithms.

6.2.2.1. Description of the New Algorithm

To introduce this new algorithm, consider the current-voltage plot shown in Figure 6-3. This plot provides a hypothetical example of different situations in which a perturbation and observation algorithm has been used to determine the maximum power point of a system. The graph illustrates the sets of current-voltage points that were recorded when the algorithm tracked from one maximum power point to another and the maximum power point on each occasion. The benefit that the new algorithm aims to introduce is to quickly find the maximum power point for an unknown curve for which a single data value has been recorded.



curve is defined and all of the recorded data points, the maximum power point and the maximum power point voltage are stored and associated with that curve. If the maximum power point does not lie within $\pm\Delta P$ of another curve's maximum power point then all of the data points recorded during the tracking phase are associated with the existing curve.

Once the new maximum power point has been found, and the associated information recorded, the algorithm enters a waiting state where the power output of the solar panel is monitored until it changes by more than a specified value P_{DRIFT} . Under this circumstance, the classification and tracking process begins again. A full formal definition of the algorithm is provided in the activity diagram in Figure 6-4.

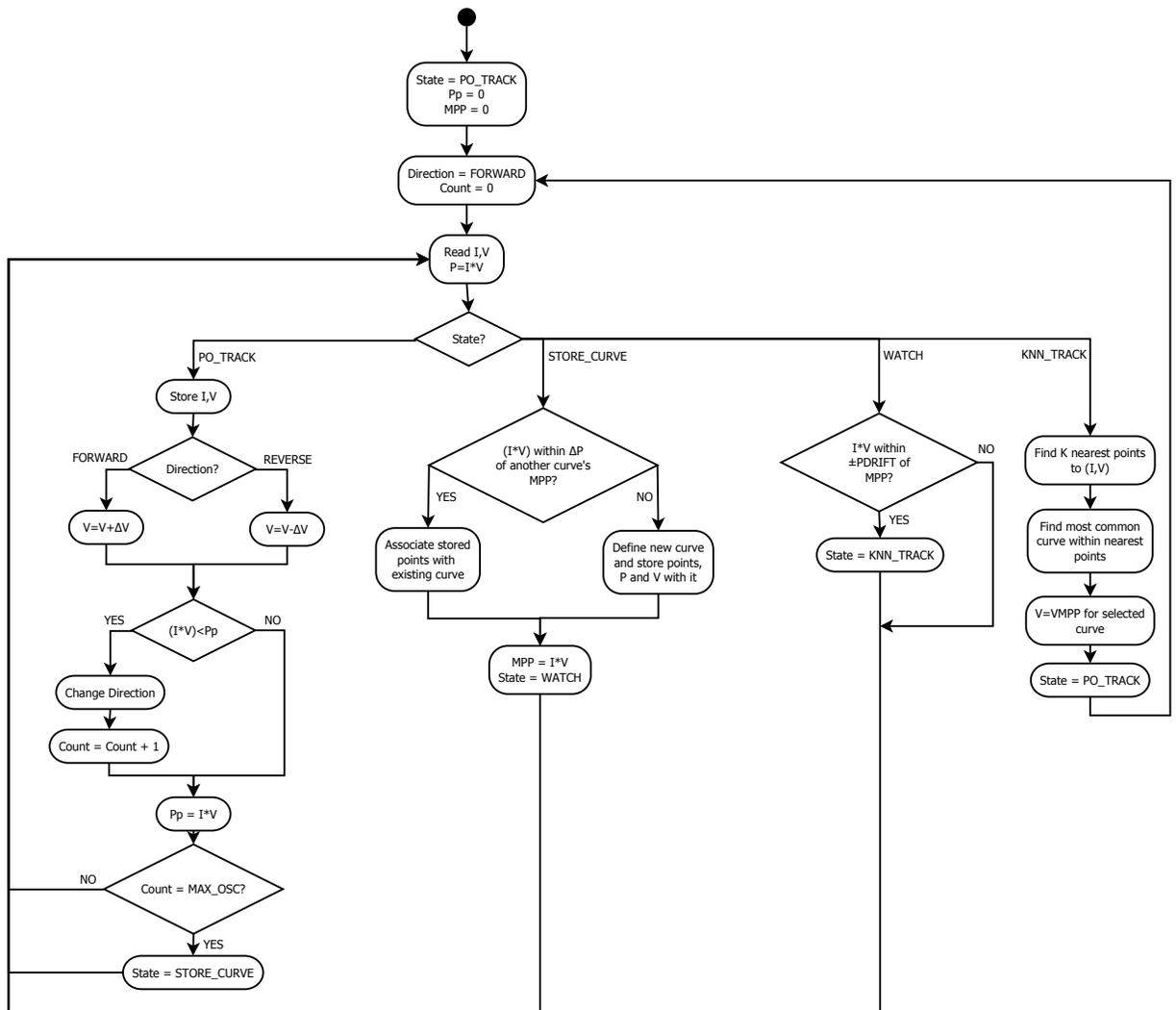


Figure 6-4: Activity diagram of the newly-developed learning maximum power point tracking algorithm.

6.2.2.2. Algorithm Comparison

In order to assess the effectiveness of the new algorithm, both simulation and experimental comparisons were carried out between the basic perturbation and observation algorithm, the incremental conductance algorithm and the new learning algorithm. Full details of the methods used to perform the comparisons are provided in Appendix E. The results of these comparisons are presented in Table 6-1.

The simulation results indicate that the new algorithm provides a higher average power output than the existing algorithms under both slowly and rapidly changing atmospheric conditions. However, the experimental comparison indicates that the Incremental Conductance algorithm provides slightly better performance under slowly changing conditions. The new algorithm is shown to provide an improvement on average power output of 7% under rapidly changing conditions. The poorer performance of the new algorithm under experimental conditions was attributed to the presence of noise in the measurements which was not modelled in the simulation. This noise causes the algorithm to oscillate at certain points during the tracking process and is wrongly interpreted as indicating the presence of the maximum power point.

Table 6-1: Results of simulation and experimental comparisons of maximum power point tracking algorithms under slowly and rapidly changing atmospheric conditions.

Algorithm	Average Power Output (W) ¹⁰			
	Simulation Comparison		Experimental Comparison	
	Slowly Changing Conditions	Rapidly Changing Conditions	Slowly Changing Conditions	Rapidly Changing Conditions
Perturbation and Observation	50.0878	53.3551	1.8206	2.4781
Incremental Conductance	50.0890	53.3736	1.8214	2.5246
Learning Algorithm	40.4403	53.9427	1.7875	2.7231

¹⁰ The large difference in simulated and experimental power outputs is attributed to the fact that in the simulation, the solar panel was subjected to levels of solar irradiation that could not be repeated in an experimental setup.

The best illustration of the improved tracking performance offered by the new algorithm is its ability to rapidly track to a previously discovered maximum power point. Figure 6-5 and Figure 6-6 illustrate the performance of the Incremental Conductance and the Learning algorithms when compared under experimental conditions using the same solar irradiation pattern. It can be observed from the second graph that the new algorithm behaves similarly to the Incremental Conductance method on the first attempt at obtaining the maximum power point. This is due to no prior knowledge of the system. Upon subsequent occurrences of the same maximum power point, the new algorithm reaches the maximum power point almost instantly. In comparison, the incremental conductance method takes the same amount of time as the initial tracking on each occasion.

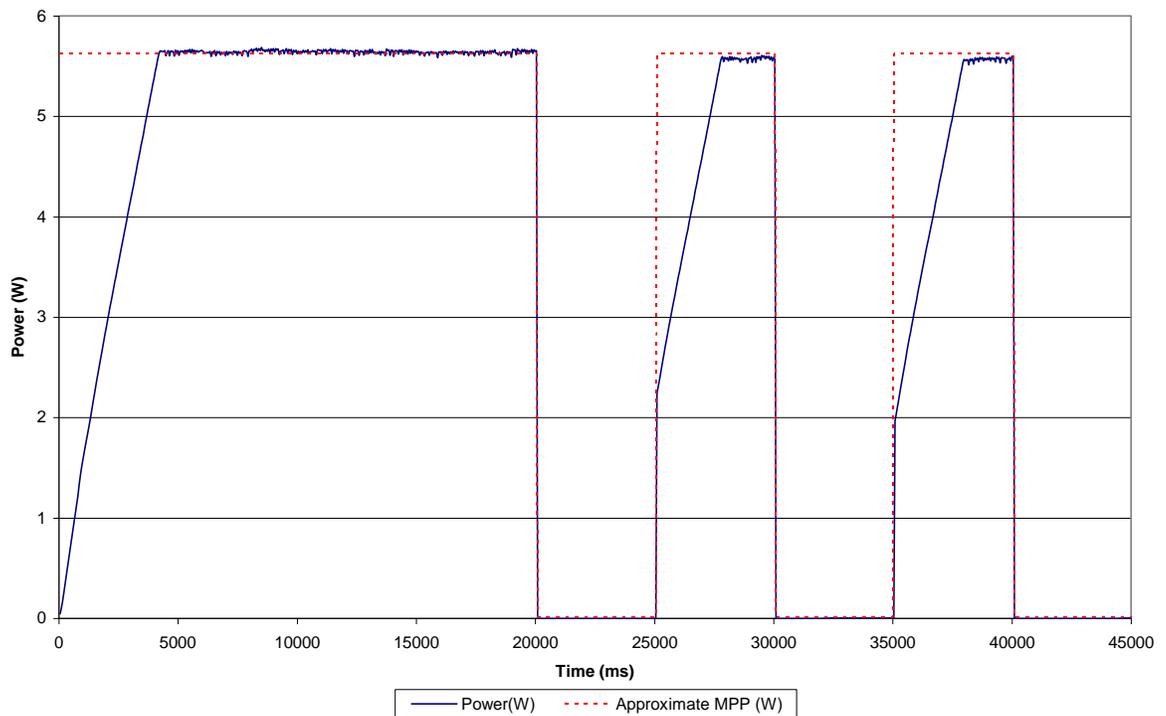


Figure 6-5: Power output of a solar panel under experimental conditions when using the Incremental Conductance maximum power point tracking algorithm. The dotted line indicates the approximate maximum power point of the panel under fully-illuminated experimental conditions.

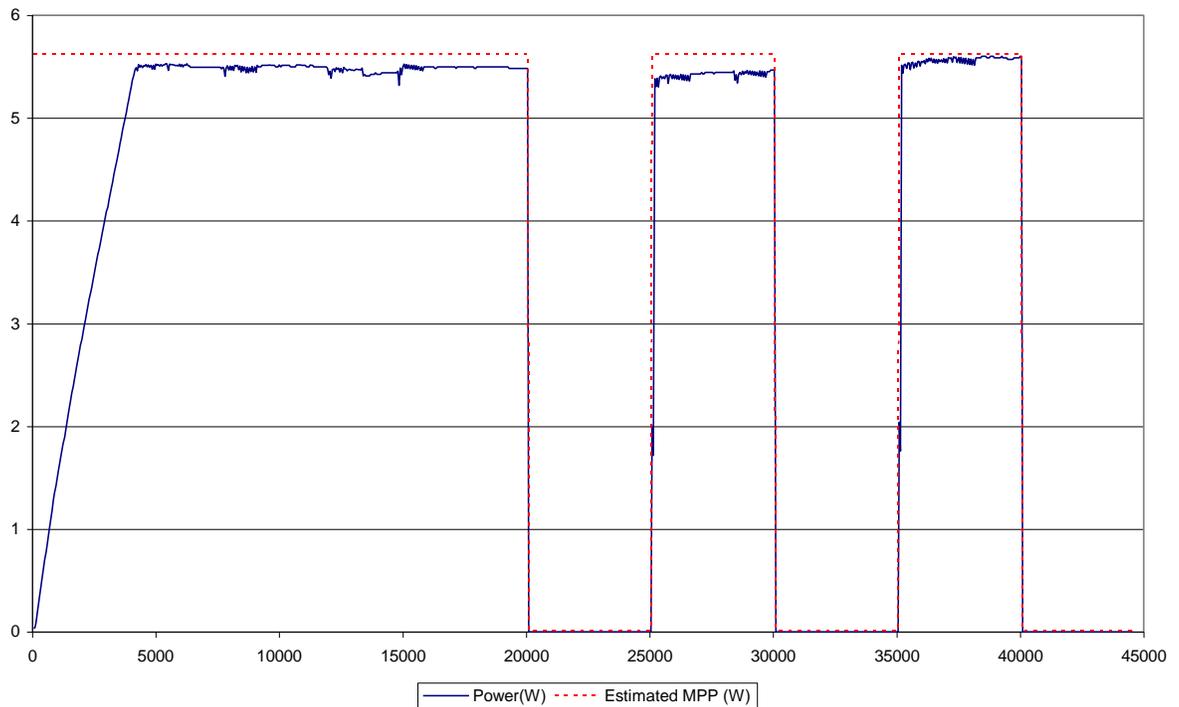


Figure 6-6: Power output of a solar panel under experimental conditions when using the Learning based maximum power point tracking algorithm. The dotted line indicates the approximate maximum power point of the panel under fully-illuminated experimental conditions.

6.2.3. MPPT Controller

The “MPPT Controller” component that is shown in Figure 6-1 (page 132) is responsible for implementing the learning-based maximum power point tracking algorithm described in section 6.2.2 to extract the maximum available energy from the roof-mounted solar panel.

The logical block within the system model that implements this controller runs the new maximum power point tracking algorithm to constantly extract power from the solar panel. The controller modifies the voltage on its output pin so that the power flowing to the home electricity network is equal to the power extracted from the solar panel multiplied by an efficiency percentage to model the losses in the power electronics of such a controller. Minimum and maximum voltage constraints are provided within the controller. The maximum voltage limit is applied when the photovoltaic panel is generating more power than can be consumed by the home energy network, storage systems or through grid export.

In this case, excess power generated by the photovoltaic panel will be unused within the system. The minimum voltage constraint relates to a scenario when no grid connection or battery storage is available and the controller is solely responsible for supplying electricity to the home. If the energy consumption in the home is sufficient to reduce the controller's output to the minimum voltage level then the controller's output will simulate the tripping of an overload protection device and disconnect the energy supply.

In this study, an 80W solar panel model was used, in conjunction with a typical Scottish solar irradiation pattern for the month of March [131].

6.2.4. Battery Controller

The "Battery Controller" component shown in Figure 6-1 is responsible for managing charging and discharging of the battery storage system. The controller accepts set-points for operation between -100% and 100% where a negative set-point indicates charging and a positive set-point indicates discharging. The controller will use these set-points to charge or discharge the battery at a percentage of its maximum charge or discharge rate. When charging the battery, the controller will model a resistive load connected to the home energy system and set the voltage at the battery terminal such that the power flow is equal to the requested charging rate. Conversely, when discharging the battery the controller will model a resistive load connected to the battery to discharge the battery at the requested rate and will use the same method as the MPPT controller to inject power into the home energy network. Output voltage constraints are used on this component in a similar manner to the MPPT controller. However, under limiting conditions, the charge or discharge rate of the battery reflects the limited power input or output at the connection to the home energy system. To allow the controller to detect the battery's state of charge, the minimum and maximum open-circuit voltage for the battery are defined in configurable parameters. When these values are reached, the controller will enter an idle state and cease to import from or export to the home's electricity network.

In this study a 1kWh battery system model is used with a maximum charge rate of 1kW and a maximum discharge rate of 3kW.

6.3. Water Heater

The electric water heater component described in section 4.1.5.4 was used within this model as the controllable thermal storage element. An additional capability was added to the heater for use in the smart energy system. Instead of heating water based on a thermostat setting, water is heated based on a percentage of input power setting. This means that when sent a percentage set-point, the heating elements will operate at the given percentage of their maximum power. To ensure that hot water comfort requirements are met, the power setting will be overridden to operate at maximum power when the water temperature is below a configured minimum value. Similarly, the heating elements will be switched off when the water temperature reaches a configured maximum value. This mode of operation places the focus on maximising the tank's thermal energy storage capability. The heater was configured as a 6kW immersion heater in a 120 litre storage tank with a maximum tank temperature of 90°C, minimum tank temperature of 55°C and a loss of 0.9kWh/24h.

The results of an Energy Saving Trust study into domestic hot water usage [134] were used as a basis for creating hot water usage patterns. Figure 6-7 illustrates the total hourly consumption of hot water as recorded for a typical home in their survey. To provide a simple representative consumption of hot water for the simulated home model, the total hourly consumption in litres for each hour was divided by 60 to provide a constant consumption rate in litres per minute for that hour which matched the total water consumption for the hour.

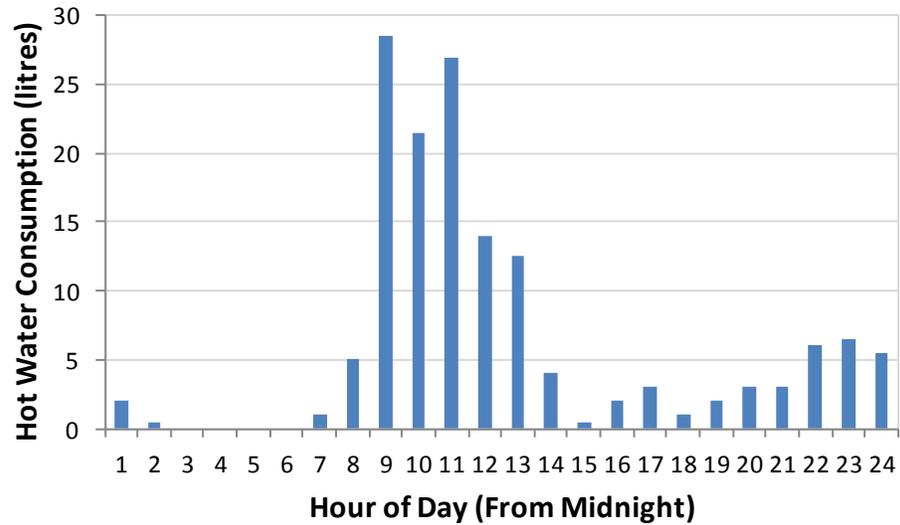


Figure 6-7: Consumption of hot water per hour as recorded in a typical home within the Energy Saving Trust hot water survey.

6.4. Domestic Load Model

In order to simplify the model for this case study, the domestic electricity load not under the control of the energy management system was modelled as a single time-varying lumped load, rather than as a set of individually scheduled appliances. To ensure that the simulated load was representative of the actual load in a typical UK dwelling, the model proposed by Richardson *et al* [135] was used to generate a 24-hour load profile with a one-minute resolution. The probability-based model used to generate this profile uses three steps: simulation of building occupancy based on maximum occupancy and time of day; assignment of appliances to the building; and modelling of appliance energy use dependent on building occupancy and probability of appliance use. Lighting load within the building is calculated based on a seasonal lighting model developed by the same authors [136].

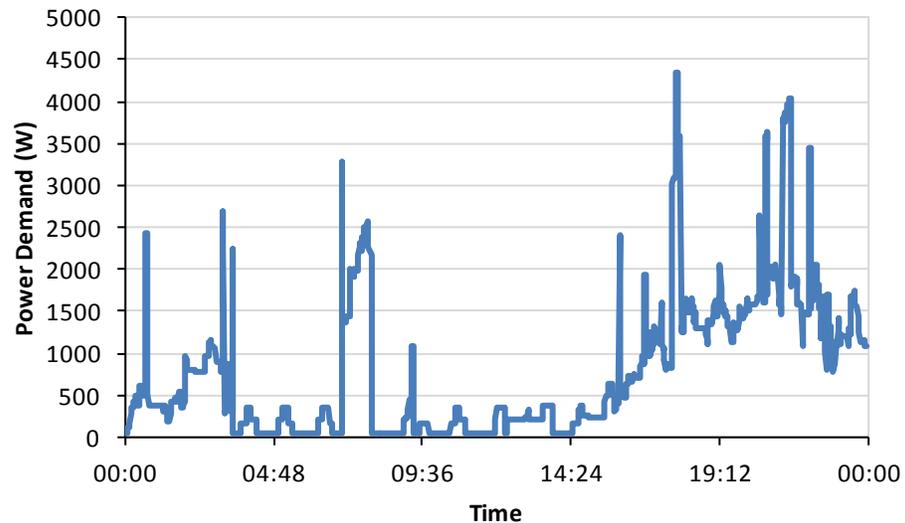


Figure 6-8: Simulated electrical power demand for the home.

The inputs to the model were set up to model a home with four occupants, with the seasonal profile initialised for the month of March. During the appliance assignment phase of the profile generation, the initial assignment made by the algorithm was taken as a starting point, but was modified to ensure that there were no electric space heating or water heating appliances assigned to the home before generating the profile. The resulting profile that was generated by the model is shown in Figure 6-8.

6.5. Metering System and Control Algorithm

The smart metering component described in section 4.1.4.5 was used to measure the energy import or export between the home and utility grid network. This meter supports the ability for the control system within the home to request the real-time power flow to or from the home using the meter's communications link.

The main in-home energy control system is capable of working in conjunction with the smart meter in a load-leveiling mode. When operating in this mode, the system uses a single output to control the behaviour of appliances within the home. This control parameter, known as the energy controller set-point (SP_{EC}) has a range of -100% to 100%. A set-point of 0% to any device indicates that the device should operate normally. A negative set-point indicates that the device should export energy, with -100% indicating the maximum energy export that is

possible from the device. A positive set-point indicates that the device should consume and store energy where possible, with 100% indicating that the device should consume energy at its maximum rate.

The battery controller supports receiving set-points directly in the range -100% to 100%, however, the water heater is only capable of responding to a set-point in the range 0% to 100% as it has no electricity export capability. An Intermediate communications component has therefore been included to convert set-points in the range -100% to 100% to a set-point in the correct range. The logic within the intermediate component converts any set-points between -100% and 0% to 0%.

A simple control algorithm is used to determine the set-point that should be issued to each controllable appliance in order to achieve the required control action. The controller uses a one-minute time-slot based average demand profile for the home for that particular day of the week. The average demand over the whole day is then calculated for use as a target constant load profile for the home. If the home energy use is less than the target average demand level, SP_{EC} is increased and if greater, SP_{EC} is decreased.

6.6. Simulation Results

6.6.1. Baseline Case

To provide an initial baseline for comparison of the different control inputs for the domestic energy controller, a 24-hour simulation of the home's electricity use was carried out with the controller set to perform no control operations. In this case, the water heater operates by following a fixed thermostat setting. The photovoltaic generation system is allowed to run normally but the battery storage system is not used. After reviewing a selection of tariffs offered by UK energy providers, the metering rate for the electricity was set to 22p/kWh and the PV feed-in rate to 39p/kWh. Figure 6-9 illustrates the load profile for the home in terms of power imported from the utility supplier. The net energy imported from the distribution network was 26.82kWh at a net cost of £5.90.

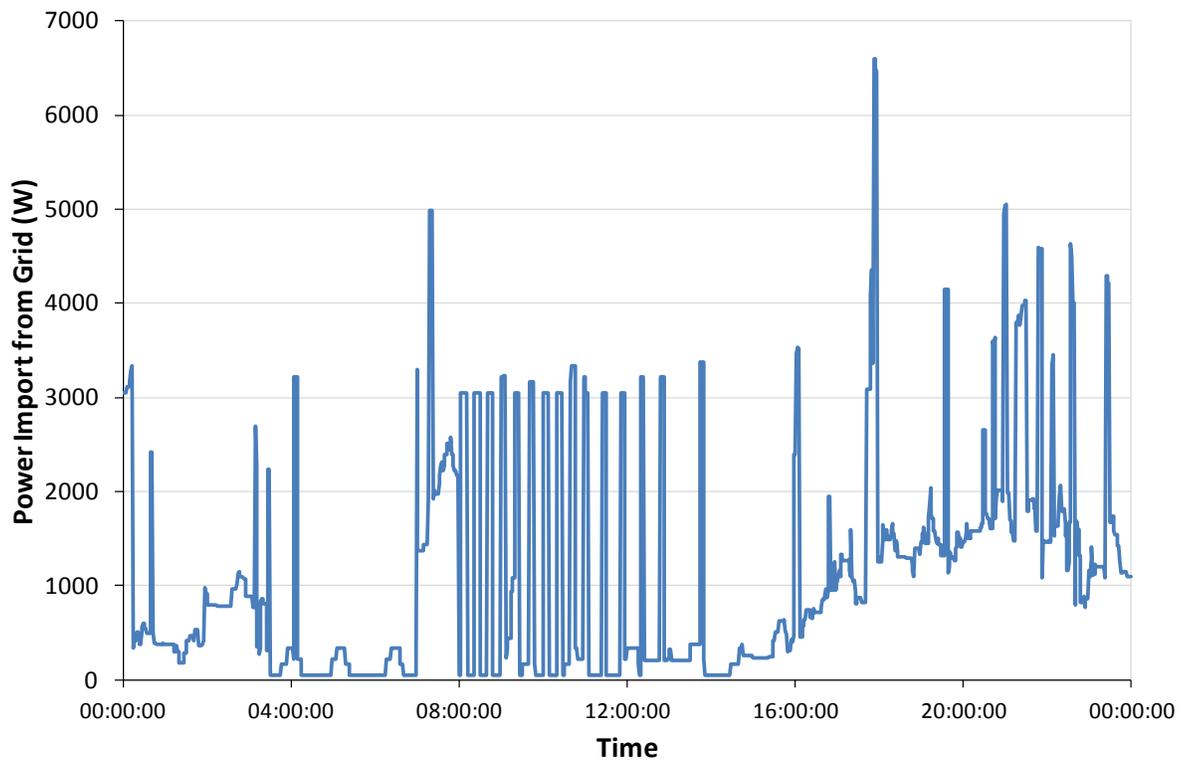


Figure 6-9: Load profile of the home over 24 hours with no thermal or electrical storage.

6.6.2. Load Balance

The next stage of this comparison involves adding some intelligent control to the home energy system by enabling the smart energy controller. The aim of this control strategy is to attempt to smooth the domestic load profile over the course of a day using the local storage capability. The results from the baseline case were used as the reference daily profile for the home from which to calculate average daily load for the home. Again, a fixed electricity tariff was used with the same energy prices as in the previous simulation.

Figure 6-10 illustrates the smoothed load profile for the home in terms of power imported from the supplier using only the hot water storage system. The net energy imported from the distribution network was 28.66kWh at a net cost of £6.30. The difference in the heating pattern used by the controller when the control system is enabled, compared to water heating without the control system is illustrated in Figure 6-11 and Figure 6-12.

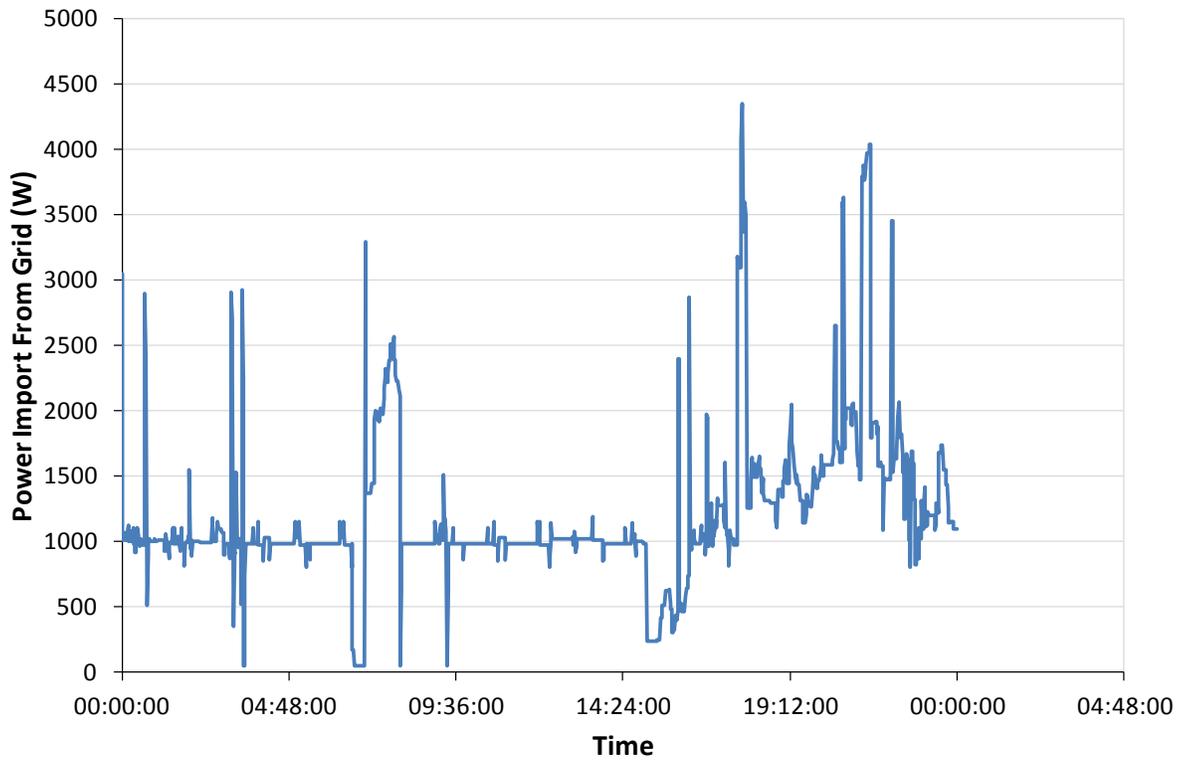


Figure 6-10: Load profile of the home over 24 hours with only the thermal storage system enabled.

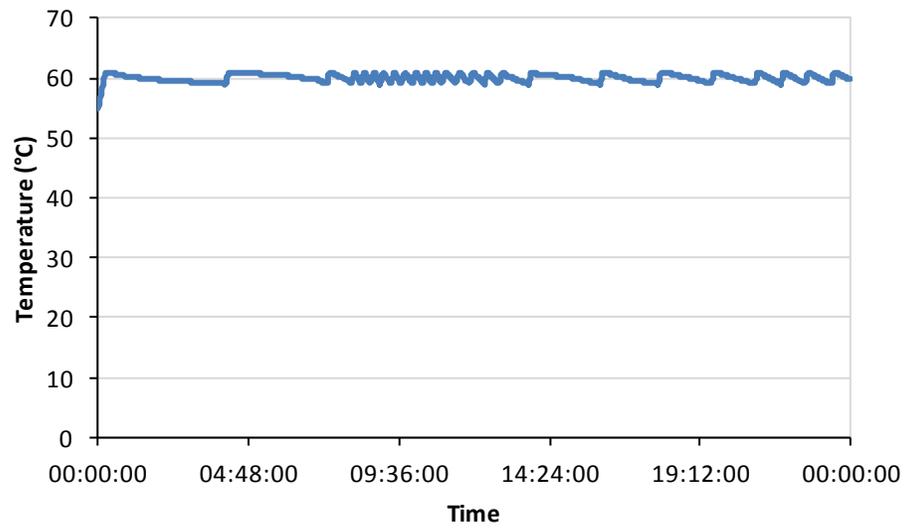


Figure 6-11: Water heater temperature over 24 hours without control system enabled.

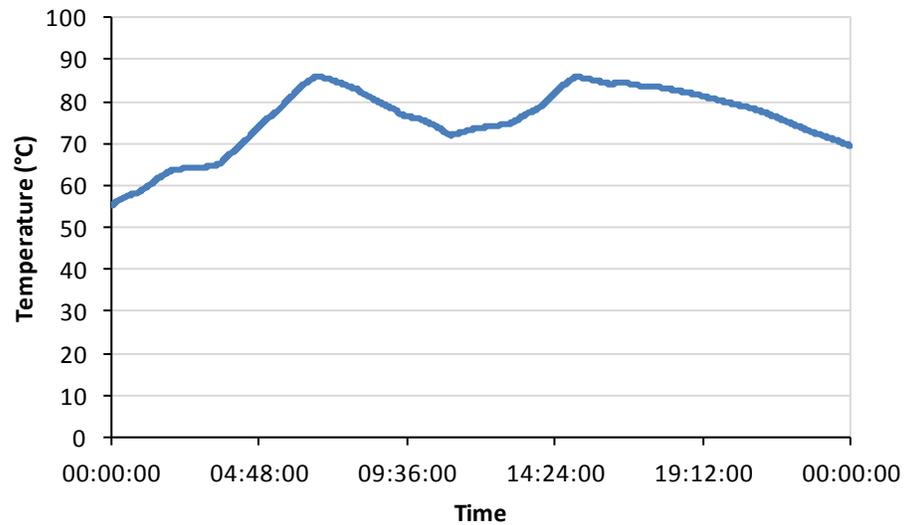


Figure 6-12:Water heater temperature over 24 hours with control system enabled.

Figure 6-13 illustrates the smoothed load profile for the home in terms of power imported from the supplier using only the battery storage system. The net energy imported from the distribution network was 22.51kWh at a net cost of £4.95.

Figure 6-14 illustrates the smoothed load profile for the home in terms of power imported from the supplier using both the hot water and battery storage systems. The net energy imported from the distribution network was 26.84kWh at a net cost of £5.90. Table 6-2 summarises the simulation results for the four control strategies.

Table 6-2: Summary of the results obtained by different domestic energy controller strategies.

Case Study	Net Energy Import from Distribution Network (kWh)	Net Energy Cost (£)
Baseline (No Control)	26.82	5.90
Thermal Storage Only	28.66	6.30
Battery Storage Only	22.51	4.95
Thermal and Battery Storage	26.84	5.90

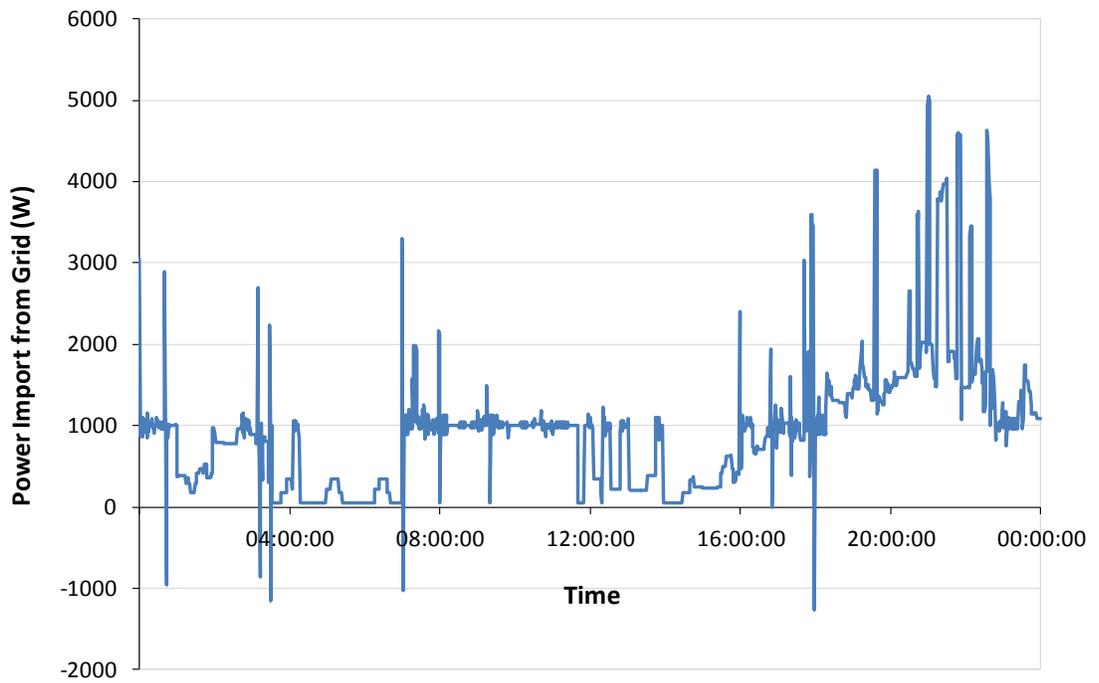


Figure 6-13: Graph illustrating power imported from the grid when only the battery storage system is enabled.

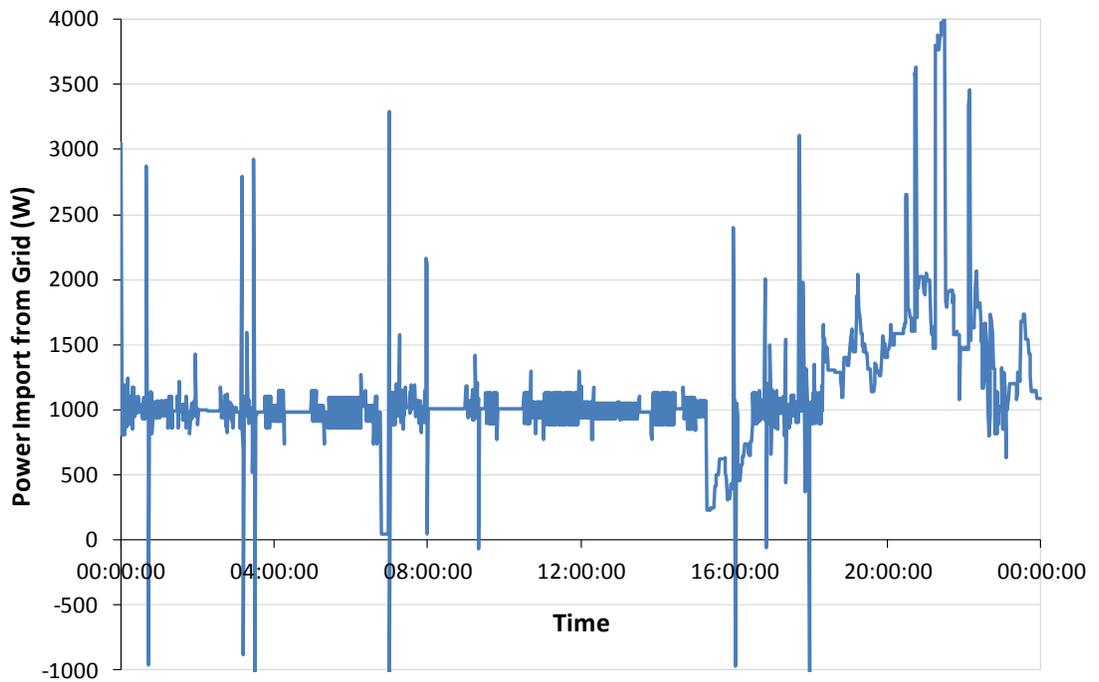


Figure 6-14: Graph illustrating power imported from the grid when both the battery and hot water storage systems are used.

6.7. Summary of Results

The baseline case illustrates a constant variation in power output throughout the day, with regular 3kW peaks due to the charging of the water cylinder. The baseline load from the home shows obvious peaks around 8AM and in the evening period from around 4PM.

Adding the control algorithm to the water heater only reduces the average load in the earlier part of the day by using a constant trickle charge rather than a large instantaneous charge in the water heater. The exceptions to this are periods around 7AM and 3PM when the water in the tank has been heated to its maximum temperature, as illustrated in Figure 6-12. A disadvantage of the control strategy used in this case is that the total daily power consumption has actually increased by around 2kWh because more energy has been stored in the water tank. This is because the control algorithm has only taken into account instantaneous power consumption and not the total energy requirement over the whole day. If the total hot water energy requirement was taken into account, this would prevent over-charging of the water tank.

As shown in Figure 6-13, using only the battery storage system with the control algorithm is less effective than the water cylinder for smoothing the power consumption in the earlier part of the day. This is due in part to the lower capacity of the battery system in comparison to the water storage system and also due to the battery being discharged to balance out the 3kW peaks introduced by running the water cylinder in its normal thermostat-based operation mode. However, the cost of electricity when using this system is lower, both due to the battery being 50% charged at the start of the simulation and also due to the higher feed-in tariff being used for electricity exported back to the grid for short periods.

The final scenario considered which uses both storage systems provides the best results for smoothing the load profile over the earlier part of the day. This is due to the battery charging sharing the power consumption with the water heater and therefore reducing the effect of the water heater being fully charged at 7AM and 3PM that was experienced previously. However, there is little effect on the evening peak load. The lack of any reduction in the peak

load is due to the battery quickly discharging within 20 minutes due to its low 1kWh total capacity.

The results obtained in this case study illustrate that smaller generation and storage systems have the ability to significantly alter the load shape of a home across a 24 hour period, which makes demand side management a useful tool for matching electricity load to grid demand. However, it has also been shown that if the control algorithm does not take into account the total energy requirements of the home then it is possible to increase the daily energy consumption using such systems, resulting in a negative financial impact on the home. It was also shown that a daily averaging profile like that used here has done little to reduce the evening peak consumption and therefore it may have been more beneficial to schedule charging of the storage to coincide with the time prior to the peak period.

Chapter 7

Conclusions

7.1. Chapter Summary

In **Chapter 2**, the background to this research project was described. In particular, the process that is currently being carried out in the UK to transition from a fossil-fuel based energy system to a system that is more dependent on renewable and sustainable energy sources was described. With these changes come new challenges based around the fact that renewable sources of energy generation may not always be available at the same time as the demand for energy. While there are larger-scale methods of addressing some of these problems, the focus of the work in this thesis was to look at ways in which the energy systems in individual homes could contribute to the storage or demand management requirements of the grid as a whole. Smart metering, microgeneration and local electrical and thermal storage were identified as possible technologies which could assist in offsetting the effect of increased penetration of renewable generation which does not coincide with demand. The second half of Chapter 2 describes different building energy simulation packages that could be used to model “smart micro-grids” within homes, taking into account the modelling of the electrical, thermal, communication and software control aspects of these packages. This summary provided the foundation for the development of a new package which better suited the requirements for the modelling of domestic smart grid systems.

Chapter 3 describes the design of the new domestic smart grid modelling package that was developed as the main part of the work for this thesis. The final design that was selected for the package models systems using block-based components representing different elements of a building’s energy system that are connected to other components through one or more pins on each component. Central to the design of the package is the support for multiple

physical modelling domains within each component. The domains currently supported are AC and DC electrical connections, one-dimensional heat transfer connections and byte-orientated communication connections. An asynchronous messaging system that does not rely on explicit connections between components is also provided to model external events such as user interaction with appliances in the home. The package itself simply provides the framework for defining components, creating models using these components and running simulations on the developed models. As well as allowing interactions between multiple physical domains to be modelled in a single component, one of the major benefits of the package is its ability to describe complex component behaviour or software control strategies through the use of full programming language support. Simple components can be implemented in a mathematical scripting language, but support for full C# and Visual Basic language use is also provided to implement complex components. Behaviour within each component can also be controlled by providing configurable parameters which can be changed at model design time, eliminating the need to modify components' source code for the design of each different system model.

In **Chapter 4**, the development of a number of component models for use in the package was described. Ideal voltage sources as well as a detailed solar panel model were developed to allow for the simulation of grid-based generation and local microgeneration from solar panels. A set of wire and switch components was developed to automate electrical connections and to model the losses in the connections. A smart meter component was also developed to represent the two-way communication interface with the utility supplier which could be expected in the future. A number of different household electrical loads were modelled with the majority of the models based on the characterisation of real appliances.

The library of building elements developed for the package includes a room component and a surface component that can be used to model walls, floors, ceilings and roofs. Separate components were developed to model the heat conduction through doors and windows. Components were developed to model an electric water heater, electric space heater and water-filled space heater. Another component simulates the heat loss from a room due to natural ventilation.

Chapter 5 described the testing and validation processes that were carried out both during and after the development of the package. Two forms of testing were carried out during the development of the package and the component libraries. Automated unit testing using the Microsoft Test Framework was used to test the package itself. This proved to be a very useful form of testing as every time that a change was made to the software, the existing set of tests could be re-run to ensure that the changes did not break any existing functionality. For the testing of component libraries, as well as performing manual testing of the functionality of components during development, an automated test program was used to run simulations of pre-defined system models and verify that the results produced by the components were as expected.

To validate the simulations carried out in the package against experimental data, three case studies were carried out. The first involved assessing the ability of the package to model the thermal properties of a room within a building with no forced heating or cooling systems in operation. The results of this study showed that the package was able to accurately model the thermal properties of a room, excluding the situation where direct sunlight caused a fluctuation in the measured room temperature which was not accurately captured in the simulations. This identified the need for improvement of the solar gain modelling capabilities of the package.

The second validation study built upon the first study, adding a space heating system into the simulation and validating against a room with two radiators operating from a gas-fired central heating boiler. This study identified a problem with the implementation of the radiator component within the package, which requires further work in order to more accurately model heating systems. This problem was partially due to the model of the radiator itself and also partially due to the way in which radiator temperature data was collected in the experimental phase of the experiment and input into the simulation. A basic linear model of a radiator was used, the inputs to which were the instantaneous temperature of the water in the radiator and the nominal heat output of the radiator for a given temperature difference between the water and the room. The temperature data collected for the validation study was from the water inlet and outlet pipes only. Due to the

significantly smaller volume of water in these pipes when water is not flowing to the radiator, the pipes cool significantly more quickly than the radiator itself. This resulted in erroneous input data being collected for the validation study. A resolution to this problem would be to improve the radiator component to more fully model the physical properties of a radiator, taking into account the flow rate of water through the radiator, and the rate of cooling of the radiator once water has stopped flowing. The experimental measurements could be improved to support a new model of this type by collecting flow rate data along with the water temperature data.

The third validation study was based upon the simulation of electrical appliances within a home. This study illustrated good correlation between the simulated and measured data but identified a discrepancy in the results when the electrical power consumption of the appliances varied dynamically. This discrepancy was due to the fact that all electrical appliances used in the simulation were modelled as non-varying static loads when in fact the appliances do have some variation in their power consumption when operating. The errors in this validation experiment could be reduced by a more detailed characterisation of the power consumption of each appliance. More detailed models would also require a more in-depth model of the interactions between occupants of the home and their appliances.

Chapter 6 describes an example domestic smart energy system that was modelled using the package. The purpose of this example system is to illustrate each of the different features of the package being used to solve a real-life domestic smart grid problem. In the example presented, a domestic energy system consisting of a photovoltaic microgeneration system, a battery storage system and thermal storage in the form of a hot water tank was considered. The home additionally had a load pattern representing the remainder of the home's uncontrollable electrical load. The aim of the study was to illustrate how the controllable generation and storage devices could be used to control net domestic energy consumption from the domestic supply.

The results from the study in Chapter 6 illustrate that while local microgeneration can slightly reduce the instantaneous power requirements of a home when energy is available, thermal and electrical energy storage systems are extremely effective in changing the shape of the

home's daily load profile. Thermal energy storage systems can level out troughs in the load profile while electrical energy storage systems have the additional capability of shaving peaks in the load profile. However, the study also highlighted the potential negative effects of automated domestic energy controllers where algorithms attempting to solve grid-level constraints can adversely affect end-users. In this case, one of the control scenarios was shown to increase users' energy consumption through wasted thermal storage and hence increase the cost of energy.

7.2. Novel Contributions of the Research

The smart domestic energy system simulation package, described in Chapter 3, provides the unique features of: full object-orientated programming language support within each component in a modelled system; modelling of data communication between components within a system; and modelling of interaction between occupants of a home and their appliances. These features are in addition to the modelling of electrical power flow and one-dimensional heat transfer – features already commonly available in building energy simulation packages. The object orientated programming language support with the platform provides the flexibility for it to be easily extended to accommodate new physical domain models, or enhancements to the existing models provided. This support also provides the potential for interaction with other simulation tools.

The development of a new learning-based maximum power point tracking algorithm, described in section 6.2.2 and a related conference paper, was another novel contribution of the work presented in this thesis. The algorithm provides the ability to learn the characteristics of a solar panel over time, allowing it to improve the tracking speed when tracking in a previously-encountered operating region. The new algorithm, which was shown to increase operating efficiency of a solar panel over a standard perturbation and observation approach, has the advantage over model-based maximum power point tracking approaches of not being prescribed to a particular type of solar panel. In the process of developing this algorithm, a related algorithm was developed for obtaining the current-voltage operating point of a solar panel when only the load and atmospheric conditions are

known. This algorithm, described in section 4.1.2 and a related journal paper, was used to create a solar panel model within the new domestic energy simulation package in order to perform studies on the new algorithm.

7.3. Future Work and Improvements

After some time spent reviewing the work from **Chapter 2**, it was concluded that the main improvement that could be made to this chapter would be to expand the scope of the literature search that was carried out. This would include other fields of research such as building automation which are related to the work presented in this thesis.

The experience of using the package to carry out the work in the latter chapters of this thesis identified a number of potential improvements to the design that is described in **Chapter 3**. The decision to use a component-based modelling paradigm was taken to simplify the overall implementation of the package. In hindsight, for the simulation of building fabric, a method of modelling based on architectural design similar to the Google Sketch-up input used by the EnergyPlus package would be preferable. When buildings are modelled using individual components for each room, surface, door and window the component-based models soon become very large. A future improvement to the design of the package would be to find some way to allow for an architectural input of building characteristics while maintaining the ability to have direct interaction between the electrical, thermal and communication domains within a single component.

The decision to use an inferred ground path for electrical connections was taken so that each electrical connection could be represented in the model as a single connection line, rather than a pair of connections. Combined with the Thévenin equivalent models used for each electrical pin, this allowed all electrical connections to be solved on a point to point basis rather than a system-wide solution. In hindsight, a preferable design would be to move to a system-wide solution of electrical systems as the point to point solution resulted in instability in some models which prevented timely convergence to a solution. This resulted in the necessity for a number of the components to have configurable convergence tolerances within their source code. This type of configurable parameter should actually be

implemented within the simulation engine itself, rather than being the responsibility of individual components.

The data communications simulation method used in the package is a primitive byte-orientated data communications mechanism which does not model the lower physical layer or the higher protocol layers of a full data communications system. A future improvement to the package would be to include these layers, allowing the communication system to be fully specified (for example, IP over 802.11g wireless). This would allow problems like interference in the physical layer to be fully simulated.

Two key features that were initially not included in the design of the package were identified later on in the project during the development of models within the package. The ability to embed smaller system models within a larger model would be useful to allow a number of individual rooms within a home to be modelled separately and then incorporated into a large home model, with each room represented by a single component in the larger model. This feature would allow the package to become more scalable by allowing for the re-use of subsystems within larger models. Another feature that would be a useful addition to the package is the ability to model the flow of both hot and cold water within a building. This would allow for more detailed simulation of water heating systems and water use within a building.

A number of potential improvements were identified in both the electrical and thermal component libraries that were described in **Chapter 4**. The electrical component library developed for the package could be improved by adding a greater range of generation devices including micro-hydro, micro-wind and a combined heat and power system. Additionally the modelling of some of the appliances could be improved to more accurately model the dynamic variation in the power consumed by each appliance.

To improve the library of thermal modelling components, a more accurate radiator model could be developed for the heating appliances, as the validation experiments carried out in Chapter 5 determined that there was a relatively large error in the results produced by the simulation of space heating. This would be carried out through both better characterisation

of real heating appliances and also through a more in-depth literature search to find existing methods of modelling water-based space heaters. The modelling of the thermal properties of a room within the package was found to be accurate when forced heating was not considered, with the main area for improvement being the addition of solar gain through windows doors and walls.

In order to improve the accuracy of the package, it is important that further validation studies are carried out in addition to those described in **Chapter 5**. Two areas for significant improvement in the validation process were identified. The first is that the electrical validation experiments should use a larger electrical network within a home with a greater range of appliances since the range of appliances tested in the one study carried out was small. The second area for improvement was in the thermal validation studies. Both studies that were carried out modelled a single room, using the measured temperatures from the surrounding rooms as inputs to the simulation model. To more fully exercise the package, a much larger study which involves modelling a whole building should ideally be carried out, using only the outside temperature as an input to the model. This will more accurately assess the package's ability to model the heat flow between rooms in a building. Before such a study could be carried out, more work would be required to accurately model heating systems within the buildings and to add the ability to the package to model solar gain.

The case study that was carried out in **Chapter 6** used a very simple domestic control system which relied on a perturbation and observation based control algorithm. The weaknesses in the algorithm were illustrated by oscillations in the load profile when a fast response to changing load was required. Further work to enhance the controller could introduce a control strategy with error feedback such as a P-I (proportional-integral) controller.

Although this study illustrated smoothing of the domestic load over the course of a whole day, the control technique could be enhanced through the addition of communication from the utility supplier. This would allow the home energy controller to manage demand in response to pricing events, for example through the use of the time-of-use (TOU) or critical peak pricing (CPP) tariffs described in section 2.1.5. Integration with the energy supplier in

this way would allow storage to be charged prior to high-priced periods and then stored energy could subsequently be fully utilised during these periods.

In addition to the control strategy itself, a number of potential improvements have been identified for the model of the home used in the study to make it more representative of a real home. The water consumption model currently used simulates water use at a constant rate over a whole hour to make up the total hourly consumption for that hour. In reality, water consumption will take place at a higher flow rate for shorter periods throughout the hour and this may have a different effect on the thermal storage tank than that simulated. To improve the water model, a stochastic model of water consumption based on known total hourly consumption could be developed. Solar irradiation is also varied based on known hourly values and a similar model could be developed to simulate faster changes in the irradiation, for example due to cloud cover. Such a model would more fully exercise the maximum power point tracking algorithm.

Finally, the solar panel model used in conjunction with the maximum power point tracking algorithm was the same 80W solar panel model used throughout the project. An enhancement to this study would be to characterise a larger solar panel (for example, 1-4kW, typical of current domestic installations) for use with the maximum power point tracker.

Appendix A

Component Scripting API

This appendix illustrates the classes which make up the Scripting API provided within the application that users can access from within the C# or VB.NET code that implements the behaviour of components. As an entry point to the API, two properties, one of type “Component” and one of type “Simulation” are accessible from inside the scope of a component’s code. These objects provide access to all other aspects of the API.



Scheduler
Class

Methods

- GetFirstValue(): IParameter (+ 1 overload)
- GetValues(): IParameter[] (+ 1 overload)

Component
Class

Fields

- events : Dictionary<string, ulong>

Properties

- Name : string
- Parameters : ParameterCollection
- Pins : ComponentPinCollection
- Temperature : double

Methods

- CreateScheduler(): Scheduler (+ 1 overload)
- IsTimerElapsed(): bool
- NextEvaluationIn(): void
- NoMoreEvaluation(): void
- SendMetadata(): void
- StartTimer(): void

Nested Types

Complex
Class

Properties

- A : double
- B : double
- R : double
- Theta : double
- Zero : Complex

Methods

- Conjugate(): Complex
- Equals(): bool
- FromCartesian(): Complex
- FromDecimal(): Complex
- FromPolar(): Complex
- GetHashCode(): int
- ToString(): string

Simulation
Class

Properties

- AmbientTemperature : double
- Components : ComponentCollection
- DebugMode : bool
- EnvironmentName : string
- GlobalParameters : ParameterCollection
- StepSizeNanos : ulong
- StopTimeNanos : ulong
- Time : TimeParameter
- TimeNanos : ulong

Methods

- Break(): void
- Pause(): void
- RaiseError(): void
- ShowDebug(): void
- ShowInformation(): void
- ShowWarning(): void

Nested Types

ParameterType
Enum

- Boolean
- Decimal
- Integer
- List
- String
- Table
- Time
- File
- Configuration

PinType
Enum

- Electrical
- Communication
- ThermalCapacitance
- Metadata
- HeatTransfer

Weekday
Enum

- Monday
- Tuesday
- Wednesday
- Thursday
- Friday
- Saturday
- Sunday

Appendix B

Window Model Class Diagram

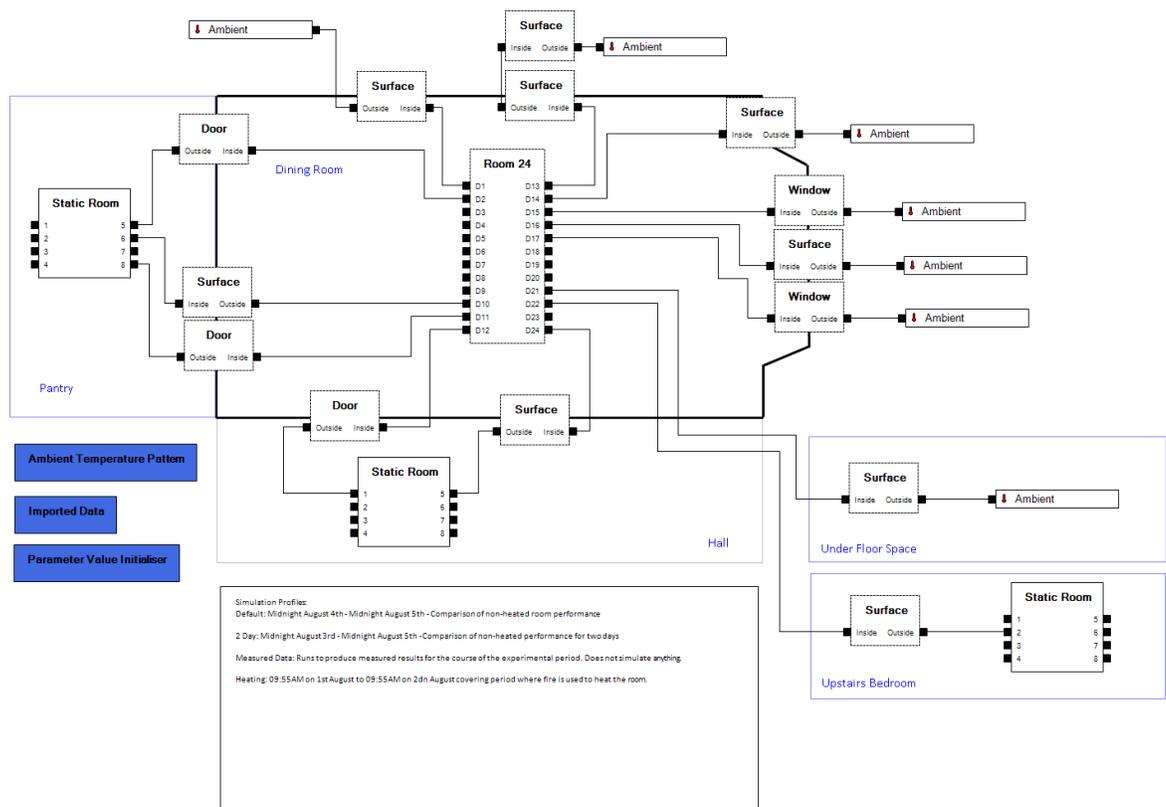


Appendix C

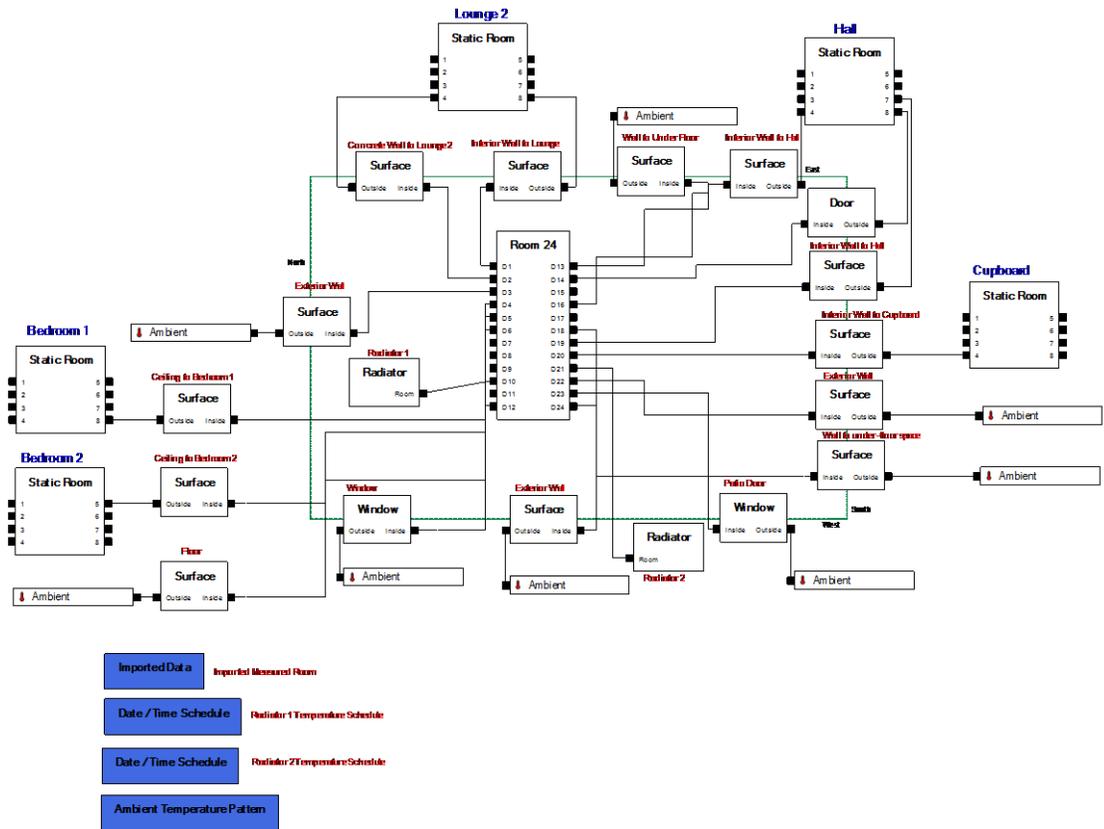
Case Study Simulation Models

C.1. Case Study 1 – Home 1, Dining Room

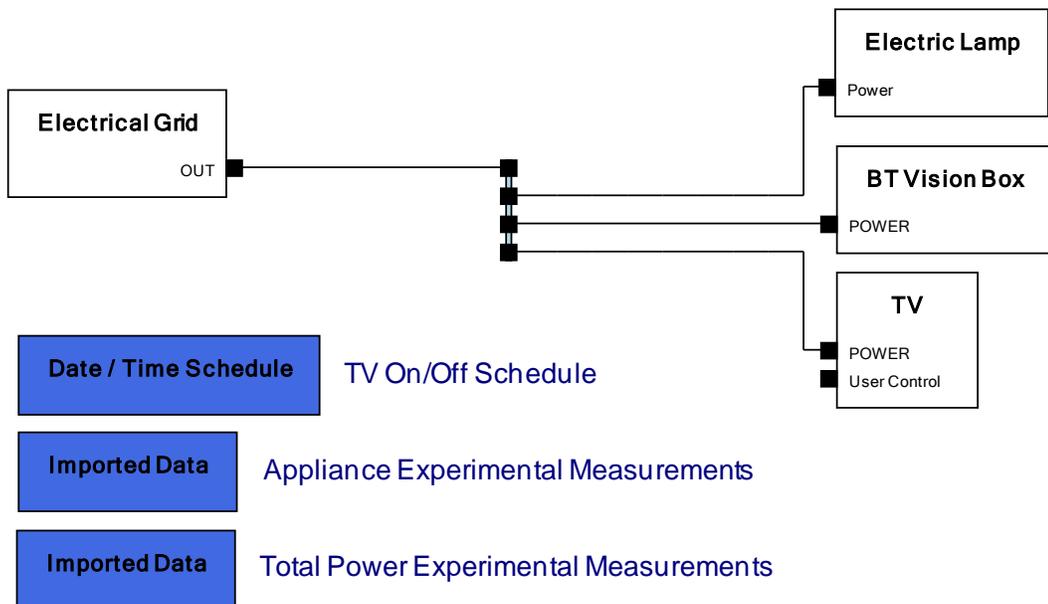
Dining Room Simulation Model



C.2. Case Study 2 – Home 2, Lounge



C.3. Case Study 3 – Electrical Validation Model



Appendix D

Power Analyser Unit

This appendix describes a purpose-built power analyser unit that was built for use in the electrical validation studies carried out in section **Error! Reference source not found.**. The reason for designing a purpose-built analyser was that the validation experiments had a requirement to measure the power consumption and power factor of domestic appliances without any modification of the appliances or connections to the appliances. The measurement system therefore had to be inserted in series with the connection to the mains supply and hence a power analyser with standard 13A sockets was required.



Figure D-1: Purpose-built power analyser unit with three standard 13A sockets and a 10A phase-controlled output.

D.1. System Overview

The power measurement system (Figure D-1) that was created is a self-contained unit mains-powered unit that provides four independent outlets from which measurements can be taken. One outlet is connected through a phase controller to permit the analysis of phase

controlled devices such as heating and lighting. Measurements can be taken from the system in two ways: 1. BNC connectors on the unit provide analogue measurement of scaled representations of the voltage and the current through each outlet. 2. A USB port enables connection to a PC which, when used with the appropriate software, can provide frequency, RMS voltage, RMS current, power and power factor readings for all outlets. Readings for all channels can be provided once per second. Alternatively, a single channel can be sampled at high speed. For a 50Hz mains waveform the maximum sampling rate is 15KHz with a resolution of 10 bits. The former is useful for steady-state analysis of power consumption while the latter is more useful for analysing transient behaviour of appliances. Figure D-2 illustrates the main components of the system.

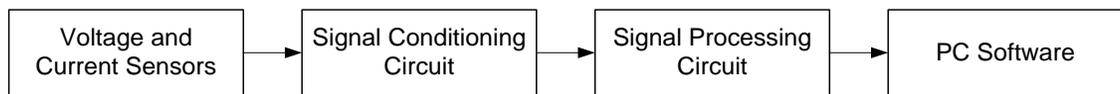


Figure D-2: Block diagram illustrating the main components of the power analyser system.

D.2. Voltage and Current Sensors

The voltage measurement in the unit is obtained using a 1.6VA 230V:9V toroidal transformer connected directly to the mains input to the system. The output of this transformer is connected through a potential divider to the signal conditioning circuit. This reduces the peak supply voltage from $\pm 370V$ to approximately $\pm 10V$ and enables the system to measure voltages up to 260V RMS. The transformer has a relatively poor regulation of 29%. However, the load on the transformer is essentially constant over the range of input voltages the unit will be used with and hence the poor regulation does not affect measurement accuracy. Laboratory tests were performed to verify linearity over the range 200-260V RMS.

The use of a current transformer around the live connection to each outlet on the unit and an additional current transformer on the live supply connection to the unit provides readings of individual and total current consumed by the devices connected to the unit. AC104 current transformers with a ratio of 1000:1 were used in conjunction with 240Ω burden resistors. A primary RMS current of 13A, the maximum supported by the apparatus, provides a voltage

across the burden resistor of 9.1V peak. The potentials across the burden resistors are fed into the signal conditioning circuit.

D.3. Signal Conditioning Circuit

A signal conditioning stage was implemented using six AD628 programmable gain differential amplifiers – one for the voltage signal and for each of the five current signals. These amplifiers were configured with a gain of 0.25 to reduce signals in the range $\pm 10\text{V}$ to the $\pm 2.5\text{V}$ range. An offset voltage of 2.5V was added so that the resulting output signal was in the range 0-5V, centred around 2.5V, to correspond with the acceptable range of input voltages to the analogue to digital converter used.

The signal conditioning circuit contains a frequency detection subsystem consisting of LM339 comparators which convert the processed sine waves into square waves, the rising edges of which can be used to detect the frequency and phase of each waveform. Both the processed voltage signals and frequency detection signals are fed from the signal conditioning circuit into the processing circuit.

D.4. Signal Processing Circuit

The final stage of processing within the unit is to digitise the voltage and frequency signals from the signal conditioning circuit and calculate the RMS voltage, the frequency, the RMS current and phase angle for each outlet. The circuit, which is powered by a freescale S12C128 microcontroller, is also responsible for sending these values to the computer when requested to by the software. As well as sending RMS data for all channels once per second, the circuit is capable of streaming the raw voltage and current readings from a single channel to the PC.

D.5. PC Software

The accompanying PC software application for the power measurement unit, shown in Figure D-3, displays the live readings of voltage, current, power, phase angle and power factor for each channel on the unit. Additionally, the software also has the facility to request a finite number of high-resolution samples for a particular channel and save these in spreadsheet format.

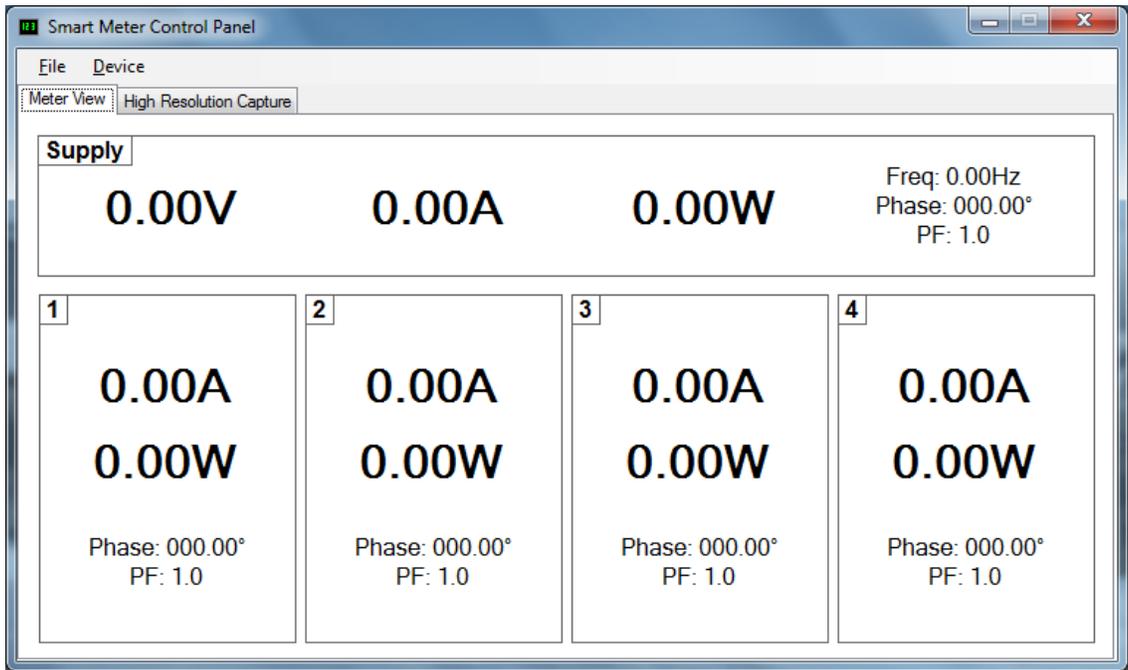
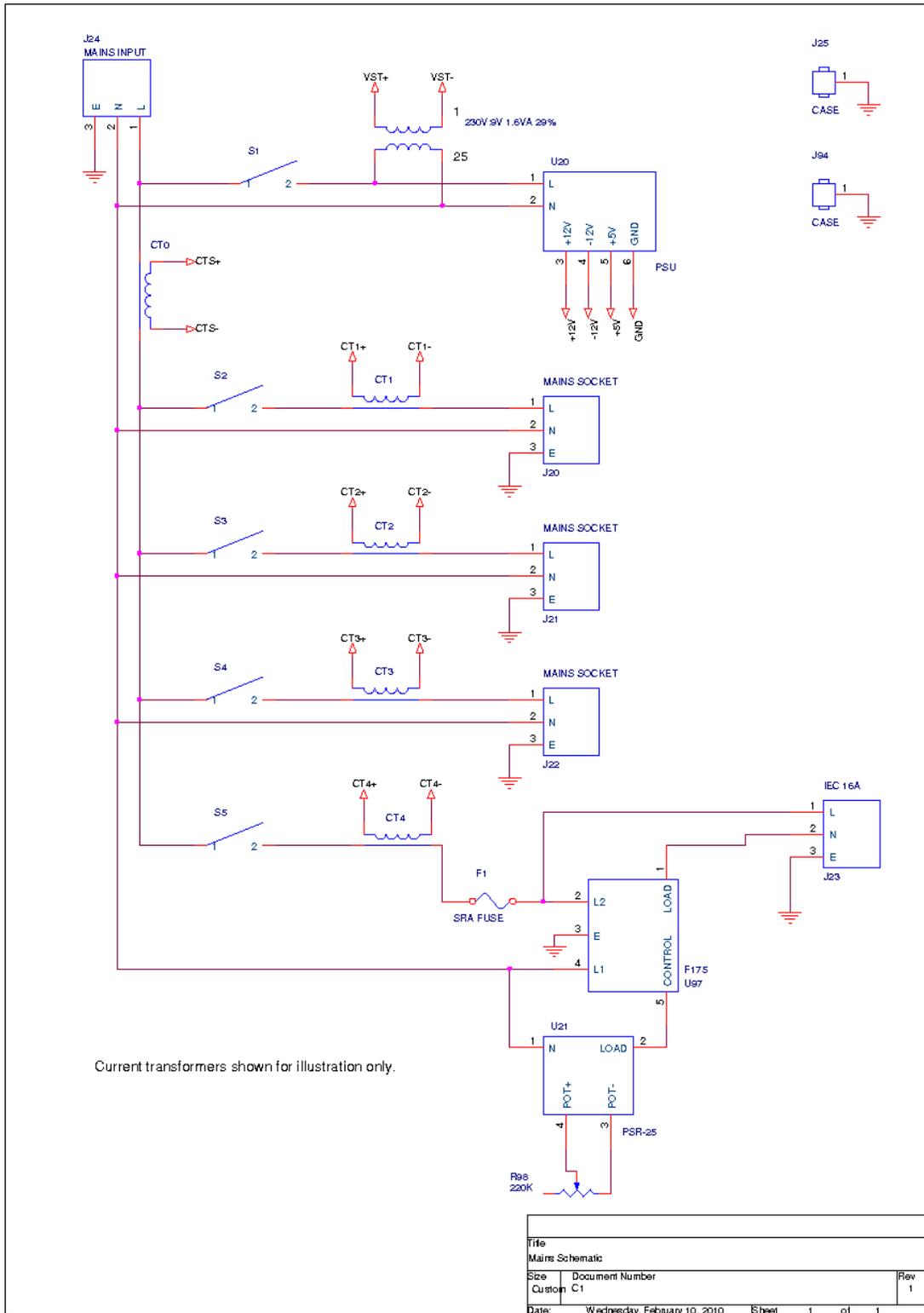


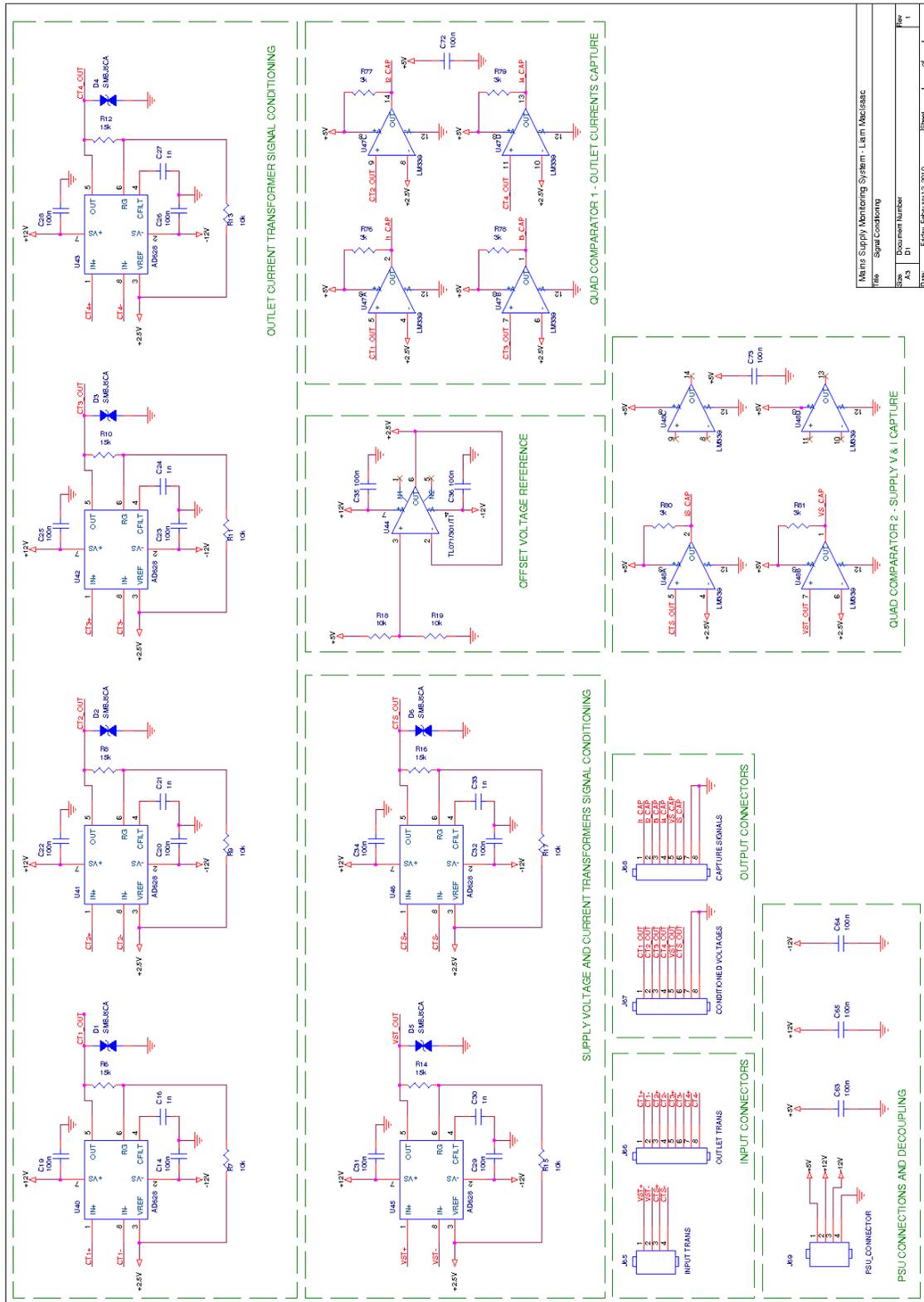
Figure D-3: Illustration of the PC software developed to interface with the power analyser unit.

D.6. Power Analyser Circuit Diagrams

D.6.1. Mains Wiring

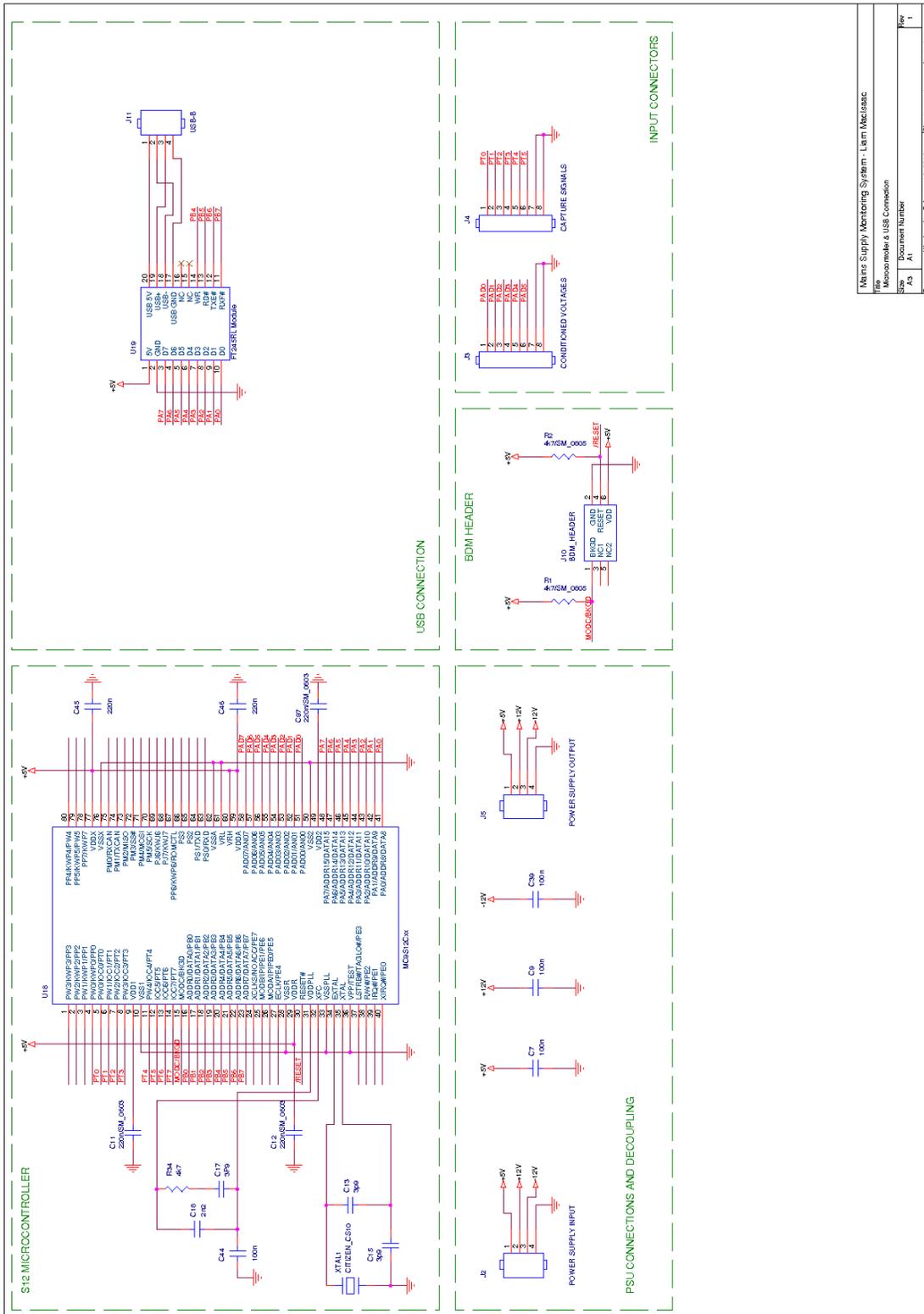


D.6.2. Signal Conditioning Circuit



METS Supply Metering System - Liam McLisac	
File	Signal Conditioning
Size	Document Number
Ab	DI
Rev	1

D.6.3. Signal Processing Circuit



Maine Supply Monitoring System - Liam McLeisc	
Title	
Microcontroller & USB Connection	
Size	Document Number
A3	A1
Date	Wednesday, February 15, 2010
Sheet	1 of 1
Rev	1

Appendix E

MPPT Algorithm Comparison Method

This appendix describes a system of hardware and software that was developed to allow for both simulation and experimental comparison of different photovoltaic maximum power point tracking algorithms. The purpose of this test system is to allow a standard test – for example, a pattern of changing solar irradiation on a solar panels – to be repeated using different maximum power point tracking algorithms. The system was designed to provide support for both simulated solar panel models and physical connections to a solar panel.

E.1. Control Software

The central component of the MPPT algorithm comparison system is a control software application, shown in Figure E-1. The application consists of three main components – the simulated or connected solar panel, the control system and the maximum power point tracking algorithm. The interface between the control system and solar panel is defined so that the control system can set the voltage across the solar panel and the solar panel can respond with the output current from the panel. The interface between the control system and the maximum power point tracking algorithms is defined so that the control system can report current, voltage and power readings for the solar panel to the MPPT algorithm and the MPPT algorithm can provide voltage set-points to the control system to be used with the solar panel. The interface definition also specifies that maximum power point tracking algorithms must have defined iterations so that the control system can periodically perform iterations of the algorithm to obtain new voltage set-points. An additional interface is provided to support simulated solar panel models which allows the solar irradiation and cell temperature values to be set in the solar panel model by the control system. The block diagram in Figure E-2 illustrates the architecture of the control software.

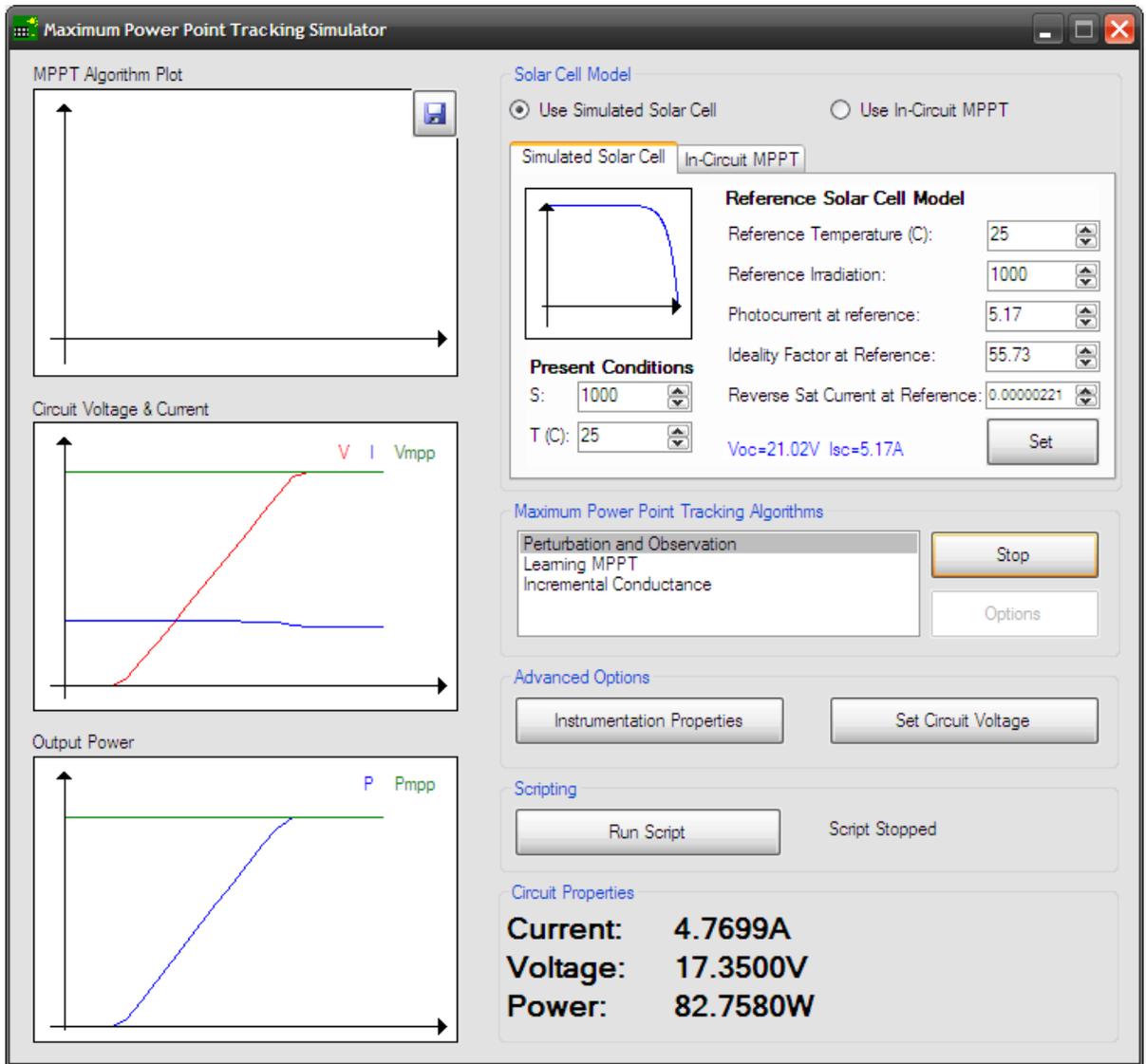


Figure E-1: Maximum power point tracking algorithm comparison system – control software user interface.

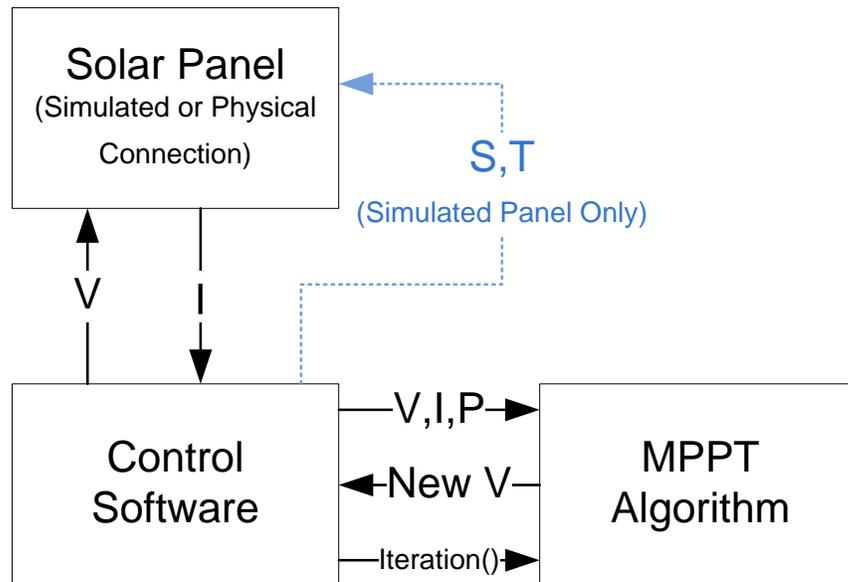


Figure E-2: Block diagram illustrating the architecture of the maximum power point tracking comparison software.

E.2. Simulated Solar Panel

The simulated solar panel model uses the single diode solar cell model which is described in detail in section 4.1.2. The model uses the voltage across the panel V (V) as well as the solar irradiation S (W/m^2) and cell temperature T (K) to evaluate the output current I (A) from the cell. The input parameters are provided by the control software, either by being directly entered by a user or through a pre-defined script. The properties of the modelled solar panel are defined through the user interface.

E.3. In-Circuit Solar Panel

To allow the use of an in-circuit MPPT comparison, the Agilent N6700 electronic load was used to provide a connection to a solar panel, along with a method to be able to control the load on the panel in order to set the panel voltage. This was achieved through the use of the load's constant voltage mode, which utilises the voltage setting provided to the panel from the control software. The controllable load has a built-in current measurement which can be passed back to the control software. Using this hardware set-up allows both simulated and in-circuit solar panels to be presented to the software through an identical interface.

E.4. Result Recording

The comparison software constantly displays voltage, current and power readings from either the simulated or in-circuit solar panel. Additionally, when using a simulated solar panel the maximum power point and corresponding MPP voltage can be displayed on traces along with these readings.

A logging facility is also built into the control software to log periodic readings of voltage, current and power to a spreadsheet file.

E.5. Maximum Power Point Tracking Algorithms

The software supports any iteration-based maximum power point tracking algorithm. New algorithms are added by creating a DLL file that implements the required interfaces to report the algorithm name, perform an iteration of the algorithm and to display a dialog to define the algorithm settings. The control software searches for these DLL files in its program folder upon start-up and loads any algorithms that are found into the list of available algorithms.

E.6. Control of Comparison Scenarios

Manual comparison of algorithms can be performed by selecting an algorithm in the user interface and running it. Solar panel parameters can then be modified to observe the algorithm's response to changes in atmospheric conditions. However, to perform equal comparisons of different algorithms in response to the same conditions, the software has a basic scripting support built in. A text file of the format shown in Figure E-3 can be used to run a MPPT algorithm, script changes in atmospheric conditions, perform with time delays and record results when using a simulated solar panel.

```
#Run the MPPT algorithm – replace name for a different algorithm  
RUN Perturbation and Observation  
  
#Vary irradiation, adding 20s delay in between each change  
SETIRRADIATION 985  
DELAY 20  
SETIRRADIATION 970  
DELAY 20  
  
#Stop Algorithm  
RESET
```

Figure E-3: Script used for the comparison of maximum power point tracking algorithms using a simulated solar panel.

When using an in-circuit solar panel, scripts cannot be used to set the atmospheric conditions, however a script like the one shown in Figure E-4 can be used to perform the same comparison of algorithms as the script in Figure E-3 by prompting the experiment's operator to vary the atmospheric conditions on the panel.

```
#Run the MPPT algorithm – replace name for a different algorithm  
RUN Perturbation and Observation  
  
#Vary irradiation, adding 20s delay in between each change  
WAIT Modify illumination on panel for 985W/m2 illumination.  
DELAY 20  
WAIT Modify illumination on panel for 500W/m2 illumination.  
DELAY 20  
  
#Stop Algorithm  
RESET
```

Figure E-4: Script used for the comparison of maximum power point tracking algorithms using an in-circuit solar panel. This script differs from the script for a simulated solar panel because the adjustment of atmospheric conditions is carried out through manual intervention by the user.

Appendix F

Software Implementation

Chapter 3 described the design of the new software package which was used as a basis for its implementation. In this appendix, more detail will be provided on the exact methods used to implement the certain parts of the package. Due to the size of the package it is not possible to document in detail the implementation of every element of the software. The sections within this appendix have therefore been carefully selected to describe parts of the implementation which were unique to this package or which were technically challenging.

F.1. Source Code Compilation

In section 3.4.3, the design of the mechanism for implementing component behaviour within the package was described. Two methods are provided – the use of full programming languages in the form of C# or VB.NET to implement complex behaviour, or the use of a mathematical scripting system to implement basic components. This section describes the technology used to implement each of these programming methods within the package.

F.1.1. C# and VB.NET

The general principle of using the fully-featured programming languages C# and VB.NET to implement component behaviour was described in section 3.4.3. Component models are based on a parent class which provides an empty implementation of the five methods which can be used to control component behaviour. The implementation of each component can override the default implementation of any number of these methods as required for that particular component. The remainder of this section describes how a set of user-defined methods and other source code defining a component's behaviour are processed and compiled into source code that can eventually be run by the simulation engine. The steps that are used to transform user-defined source code into an executable binary for use in the simulation engine are as follows: pre-processing of the code into a C# or VB.NET class;

compiling the class into an executable code component reporting any compilation errors to the user; and loading the executable code in the simulation engine. The remainder of this section describes this process for the C# language as the process is similar for both C# and VB.NET.

F.1.1.1. Pre-Processing Code

The full source code that is required to implement a typical C# class for a model component is shown in Figure F-1. The base class, *ComponentBase*, that defines the default behaviour for the component also contains the references to the properties *Component* and *Simulation* that provide the API for implementing the component behaviour.

In order to provide the simplest possible code implementation for components, users are only required to enter the source code illustrated as section 3 in Figure F-1. This is the section of code containing the component's behavioural functions, along with any extra functions or sub-classes which are necessary to implement the component's behaviour. This keeps the focus of a component's source code on the component behaviour, rather than the semantics of creating the code in the correct format for this particular application. The method follows the approach documented in the "Script Happens .NET" [107] and other similar articles on using .NET languages for scripting.

The initial implementation of the pre-processing stage of code compilation took the user defined code (indicated as "3" in Figure F-1) and inserted it within the full .NET class source code shown in Figure F-1. During testing of the completed compilation process a number of issues were found with this procedure.

Any errors reported by the compiler in the user-defined code referred to a line number which was different to the line numbers on the user interface due to the user only viewing the code shown in section 3 of Figure F-1. This problem was easily overcome in the C# implementation since C# has a pre-processor statement "#line N" which instructs the compiler to treat the line that immediately follows as line N of the code for the purposes of error reporting. Therefore, "#line 1" was inserted immediately before the user-defined code section before compiling. The problem was not as easily overcome in VB.NET as the language

did not provide the same pre-processor statement. The problem therefore had to be manually resolved by inserting a comment of the form “#line1” before the user-defined code and subtracting the position of that comment line from the line numbers generated in error messages.

Another issue that was identified was that because the list of imported namespaces shown in section 1 of the sample source code is hard-coded, component implementations would be required to fully qualify any .NET libraries that they needed to use. For example “System.Collections.Generic.List” would have to be used throughout the code rather than defining “using System.Collections.Generic” at the top of the source code and then referring to the class as “List” within the component’s implementation. This is because “using” (or “import” in VB.NET) statements can only be defined outside a class. To resolve this issue, a custom pre-processor statement was added to the C# and VB.NET scripting capabilities where a user could define “#namespace X” at any point in the source code as an alternative of the “using” or “import” statement. When the source code was processed to convert it into the class format shown in Figure F-1, each “#namespace” statement was replaced with a blank line and the required namespace was declared with the appropriate “using” or “import” statement at the top of the generated class.

A related issue was discovered at the compilation stage where if a particular component relied on a 3rd party library then the particular library file required would have to be linked into the component’s code at compile time. To provide the option to include additional external libraries in the code, a second custom pre-processor statement “#dll X” was added to both languages. When processing the code into the full class format, these “dll” statements were replaced with blank lines and a list of required external DLL files was generated for use by the compiler.

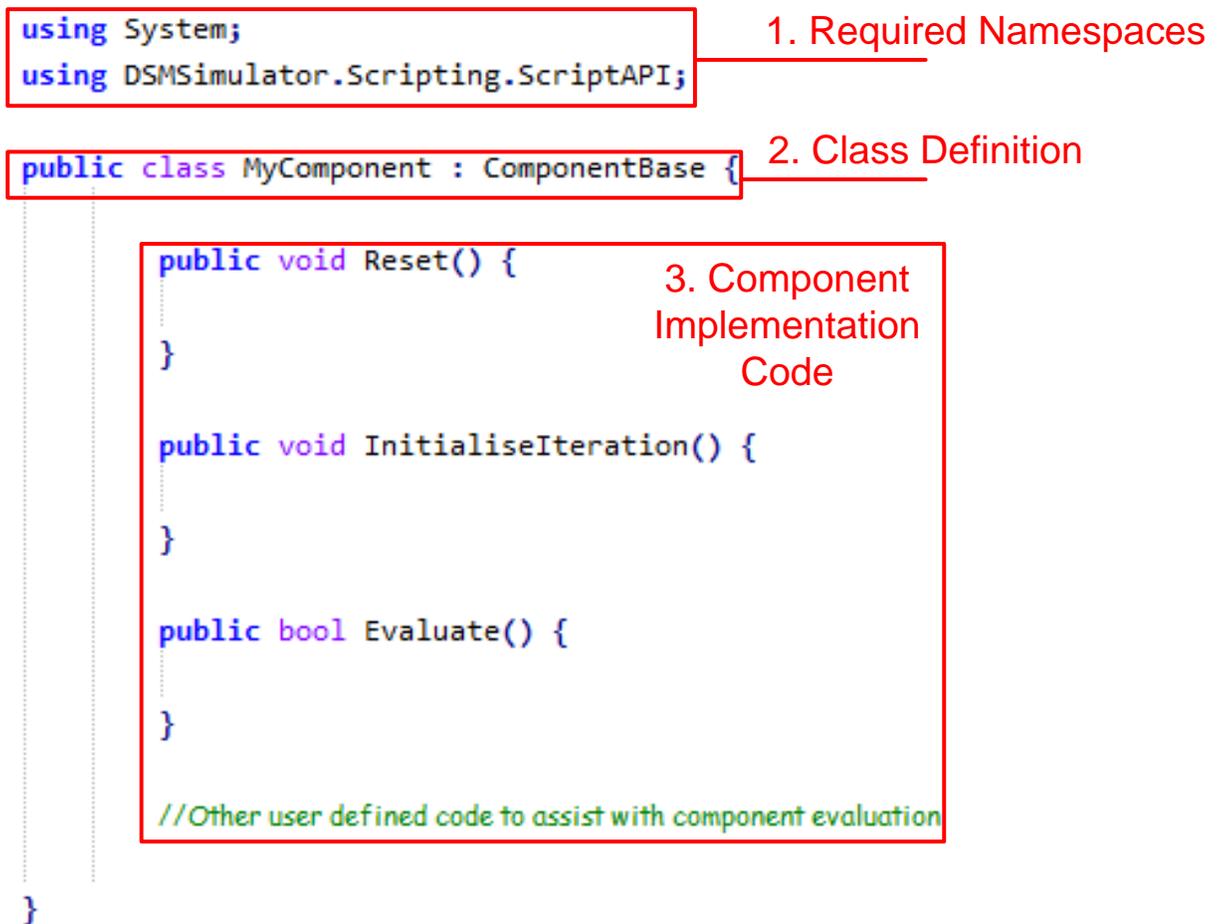


Figure F-1: Illustration of the C# code required to implement component behaviour.

F.1.1.2. Code Compilation

The .NET framework provides a set of libraries in the form of the *System.CodeDom.Compiler* namespace which provide runtime code compilation facilities for applications. These libraries provide a generic method for compiling code written in any language into a .NET executable file. Separate libraries must be used with the compilation classes to provide bindings for particular languages. The *Microsoft.CSharp* and *Microsoft.VisualBasic.CompilerServices* language binding namespaces are provided in the end-user distribution of the .NET framework and therefore these languages were chosen to be supported in the simulation package. The compilation classes were used to compile the pre-processed component source code into a DLL file for use by the simulation engine. Any errors during the compilation

process are reported in the form of a line number and an error message by the compilation classes. This can be used in the relevant part of the application's user interface.

F.1.1.3. Loading Compiled Code

The generated DLL files were loaded for use in the simulation engine using the .NET *Assembly* class which is an element of the .NET code reflection features. The class within the DLL containing the actual component source code was then identified by searching for classes in the DLL file which implement the *ComponentBase* class. A new instance of the class was then created and the *Simulation* and *Component* properties of the class set to represent the model being simulated. One problem with this method of dynamically loading code is that once an assembly (DLL or EXE file) has been loaded into memory, it cannot be unloaded until the application exits. To minimise the effect of the memory leak caused by this constraint, the executable code for each component within a model was only compiled once per application session and re-used wherever possible. An exception to this rule is when the component's source code is changed using the component editor. The code must then be re-compiled to reflect the new behaviour, leaving the old code in memory until the application exits.

A potential solution for this memory leak was found. However, it requires a significant architectural change to the software and therefore its implementation was not possible during the time available. Briefly, the solution is to run the simulation engine in a separate .NET application domain to the user interface of the software. This is essentially equivalent to running in a separate process but with the benefit of being able to communicate between the processes using a method known as remote method calls. Creating a new application domain for each simulation run and then destroying it at the end of the simulation would eliminate the slight memory leak caused by loading compiled component code. Using a separate application domain would also allow the component code to be run in an environment with greater security than the application's main code, restricting access to system resources for security.

F.1.2. Mathematical Mark-up

In section 3.4.3, a mathematical mark-up language was defined for the implementation of simpler component models which did not require the use of a full programming language. A mathematical solver engine was created to solve single-line mathematical expressions. The ability to handle simple numeric variables and functions was included in the mathematical solver to allow it to be used with the main simulation engine.

In contrast to the fully-featured programming languages the mathematical scripts were not compiled into executable code but instead interpreted at run-time. The “compilation” stage of a mathematical script merely involves parsing the script, checking it for syntax errors and creating a class which can be used to execute the mathematical code. The class, *MathMarkupEvaluator*, which is an implementation of *ComponentBase* pre-processes the mathematical script to identify pre-processor statements, comments and script lines. Pre-processor statements are handled directly by the *MathMarkupEvaluator* class to identify the mappings of component properties onto mathematical variables and the type of optimisation to use for running the mathematical script.

A separate mathematical statement evaluator class was developed for performing the actual evaluation of mathematical statements. This class, illustrated in Figure F-2, has two methods for setting and retrieving the values of variables within the mathematical solver and a method for evaluating a single line of mathematical script.

MathSolver
+SetVariable(in Name : string, in Value : double) : void
+GetVariable(in Name : string) : double
+Evaluate(in Markup : string) : double

Figure F-2: Class for evaluating mathematical statements within the simulator.

The *MathMarkupEvaluator* class uses the pre-processor statements detected in a mathematical script to set variables containing the component state before evaluating the

mathematical script. The script is then passed line-by-line into the *MathSolver* class before retrieving the updated variable values from it to feed into the simulation model.

A number of third-party mathematical solution libraries were considered for the implementation of the *MathSolver* class, however the majority of these were either not directly compatible with the .NET framework or offered significantly more functionality and therefore memory overhead than was required for this purpose. Despite there being no off-the-shelf solution that was fit for purpose, there are established methods of solving mathematical expressions within computer programs. Two such methods were combined to create the *MathSolver* class for this application. These are the Shunting Yard Algorithm and the Reverse Polish Notation solution method. Combining these methods to solve a mathematical expression requires three stages: Tokenising; Shunting Yard Conversion; Reverse Polish Solution.

F.1.2.1. Tokenising

Tokenising is the process of converting a string representation of an expression into a format that can be understood by a computer. For the purposes of this solver, six token types were defined. These were numeric literals, variables, functions, operator symbols and opening and closing brackets. Figure F-3 illustrates how an example mathematical expression is converted into tokens.

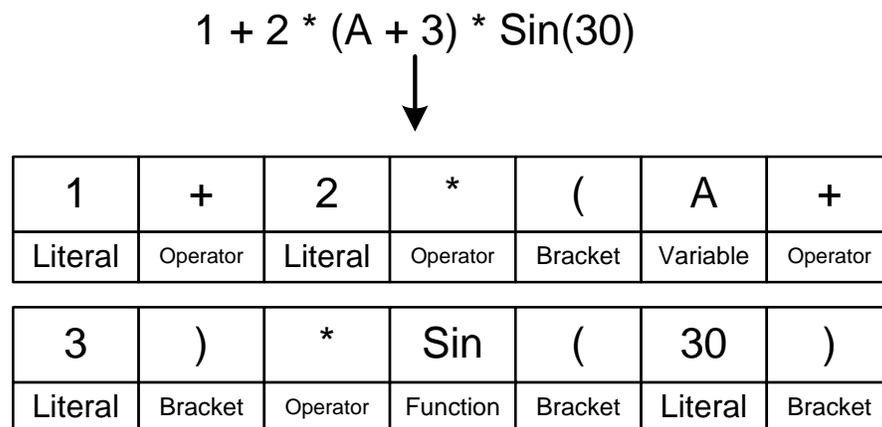


Figure F-3: Tokenising a mathematical expression.

F.1.2.2. Shunting Yard Conversion

The purpose of the shunting yard algorithm is to convert a mathematical expression in the standard notation shown in Figure F-3 (known as infix notation) into a notation that can be solved easily within a computer program. In this particular case, Reverse Polish Notation (RPN) was chosen as the output format because the solution of an expression in this notation is trivial within a computer program. The shunting yard algorithm takes into account the operator and bracket precedence rules within the expression to produce a RPN output expression which can be solved directly by the program. A number of open-source implementations of the algorithm are available. The implementation used in this case was based on [137]. Figure F-4 shows an example of the equation in Figure F-3 being converted into reverse polish notation.

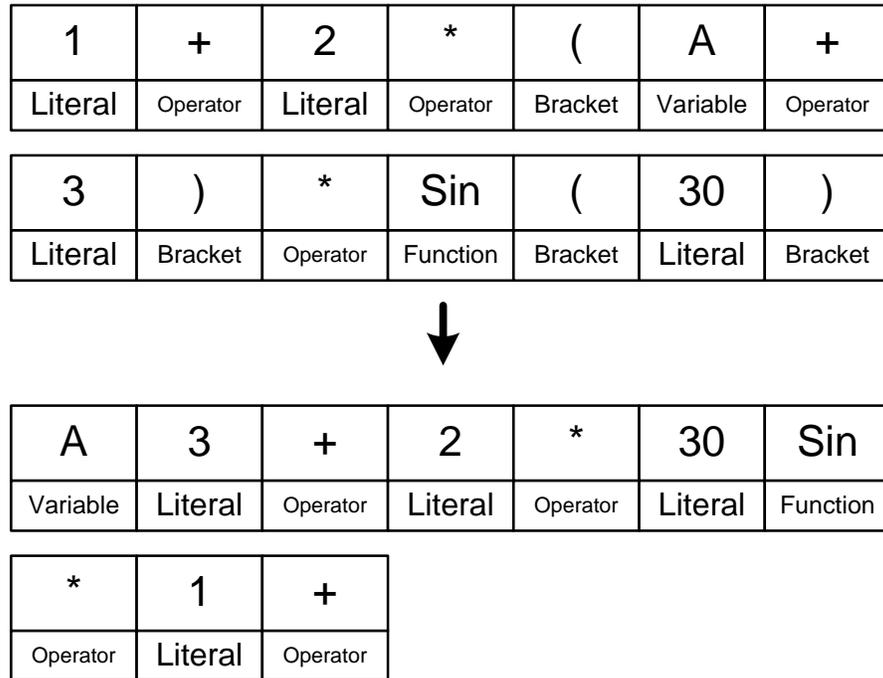


Figure F-4: Illustration of tokenised mathematical expression being converted into reverse polish notation.

F.1.2.3. Reverse Polish Notation Solution

Mathematical statements expressed in reverse polish notation can be solved using a stack-based solution algorithm. The basic premise of the algorithm is to iterate through tokens

within the expression, pushing each numeric token onto a last-in-first-out (LIFO) stack until an operator or function token is found. When an operator or function token is found, the number of parameters required by the function is removed from the stack and the operator or function is evaluated. The result of the function is then pushed onto the stack. This process continues until every token within the expression has been processed. The result of the expression is the numeric value that remains on the stack.

In the version of the algorithm implemented for use in the simulation package, variables can be used as well as numeric literal values. These variables are treated in the same manner as numeric literals when processing the tokens in an expression and are only treated differently during the final function or operator evaluation step. In the case that a function or operator is a standard mathematical function, the mathematical solver will attempt to resolve the variable to its numeric value. If the variable has not been set, an error will be generated at this stage. The assignment operator “=” has a special behaviour when dealing with variables in that if the first argument for the operator is a variable, its value will be set to the value specified in the second argument. Table F-1 below illustrates the process involved in solving the reverse polish notation expression shown in Figure F-4.

Table F-1: Illustration of the Reverse Polish Notation solution of the expression in Figure F-4. It is assumed that before solving this expression the variable “A” is set to 3. Trigonometric functions are solved in degrees.

Expression Tokens Remaining	Token Stack Before Solution	Operation Carried Out	Token Stack After Solution
A 3 + 2 * 30 Sin * 1 +			
2 * 30 Sin * 1 +	A 3 +	A+3 = 6	6
30 Sin * 1 +	6 2 *	6*2 = 12	12
* 1 +	12 30 Sin	Sin(30) = 0.5	12 0.5
1 +	12 0.5 *	12*0.5 = 6	6
	6 1 +	6+1=7	7 (Final Solution)

The implementation of this mathematical solver is an example of the type of functionality provided in the application that was well-suited to a test-driven development approach. While developing the mathematical solver, a large number of test case calculations were implemented in the Microsoft Test Framework to ensure that expressions were solved as

desired. These tests established a good level of confidence that the solution method was operating as intended.

F.2. Simulation Engine

The part of the package referred to as the “Simulation Engine” is the set of classes that are responsible for running simulations and collecting results. The method that is used for simulating models was described in detail in the design in section 3.4.2. This method involves executing user-defined source code iteratively to evaluate the behaviour of individual components. When the solution of each component is deemed to have reached a stable state, a set of result generators are used to take measurements at the desired points within the model and store these measurements for analysis at the end of the simulation.

While the algorithm proposed in section 3.4.2 is conceptually simple to implement, one practical difficulty that was encountered during the initial implementation of the algorithm concerns when control of the execution of the code is handed over to user-defined code, the simulation engine no longer has control over the application’s execution. Should an erroneous section of component logic enter a section of code that is either slow to converge or loops infinitely then the main application has no way of stopping the execution of the code using the originally proposed logic.

In order to give the user interface complete control over the execution of a simulation, the simulation engine logic was executed in a separate thread. This thread has the logic shown in Figure F-5 which allows for simulations to be paused or stopped by the user interface. When pausing a simulation, the simulation engine thread continues to execute but waits in a loop until allowed to continue. In this case, if component logic is executing when the pause request is made, it is allowed to complete execution before the thread transitions into the pause state. If the user interface requests simulation execution to be terminated, the first step that is taken is to make a cancel request to the simulation thread. This request is comprised of two actions: a “user cancel request” flag is set and a 10 second timer is started. As illustrated in Figure F-5, the cancel request flag is handled during the normal thread

processing to end the simulation once component logic for a particular time-step has been complete.

If the normal stop request process is not sufficient to cause a simulation to end gracefully, the 10-second timer that was started when the stop request was made is used to perform a more severe form of thread exit. When the timer elapses, the logic that is executed checks whether or not the simulation has stopped through the normal method. If the cancellation request has not caused the simulation to stop, the timer logic calls the .NET *Thread.Abort* method which throws an asynchronous exception within the simulation engine thread. This results in the thread being forcibly stopped. Due to the asynchronous nature of this method, there is no way of reliably determining which execution phase the simulation engine was in when it was stopped and therefore all results generated during the simulation run are cleared. If the simulation is successfully stopped using the request method then it is guaranteed that a time-step has completed and therefore the results generated up to that time-step can be used for analysis.

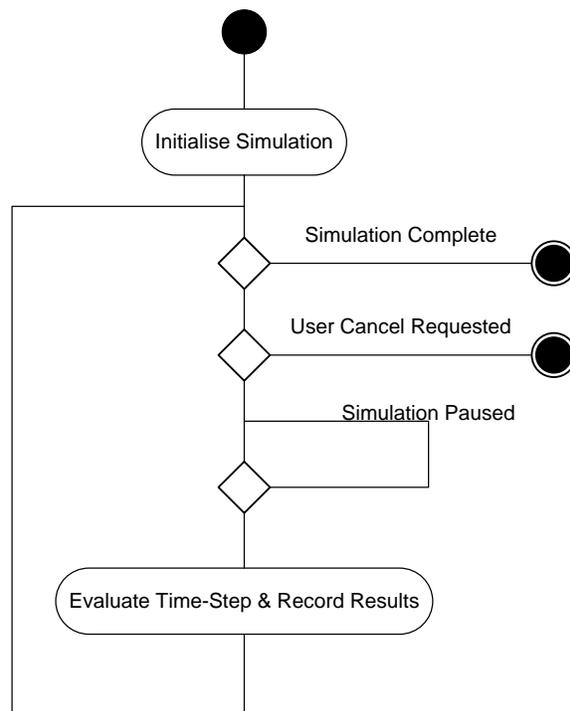
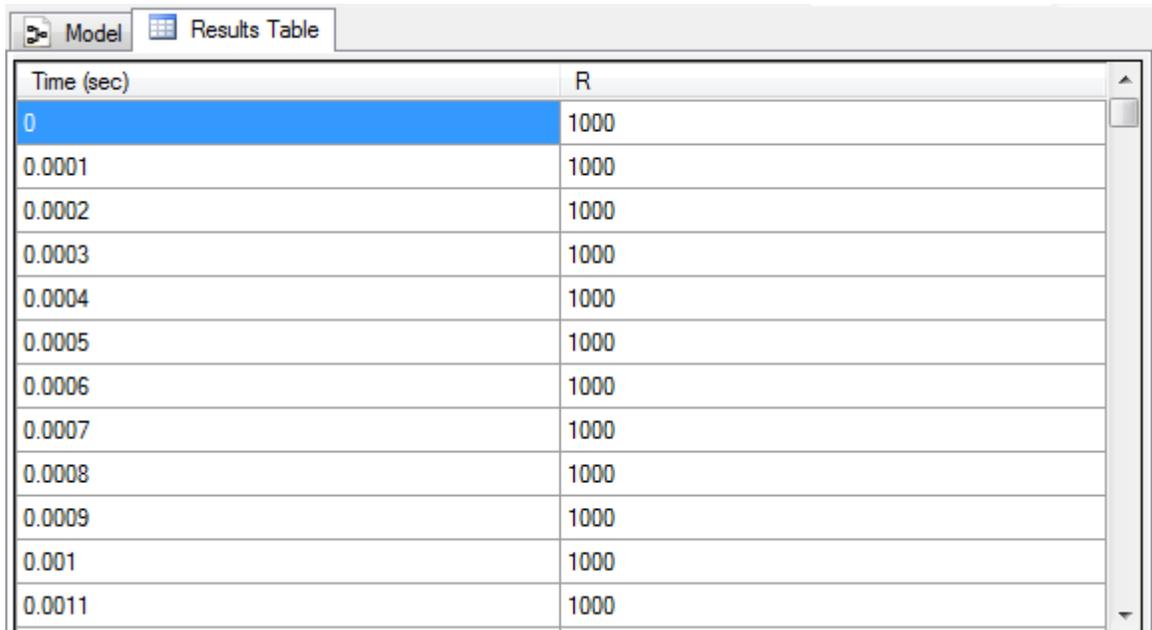


Figure F-5: Activity diagram illustrating the process used by the simulation engine thread to execute simulations while allowing the user interface to pause or stop execution.

F.3. Result Viewers

F.3.1. Table

The table result viewer, shown in Figure F-6, was implemented entirely using off-the-shelf software components that are included as part of the .NET framework. The *DataGridView* control was used as the viewing mechanism for the table data. This provides the built-in features such as cell resizing, copying of table content and sorting by column that would be expected in any standard spreadsheet package. The control also has the benefit of operating in a mode called “Virtual Mode” where the names of the columns in the table are specified along with the number of rows required but no data is added to the table contents. The control then raises an event each time that it requires data to display in a particular cell. These events can then be used to pass data from the simulation results to the control as required. The benefit of this approach is that the control does not need to store a separate copy of the entire set of simulation results – it only needs to know the data for the section of the results currently being viewed. This improves the responsiveness of the control and reduces the memory requirements of the application.



Time (sec)	R
0	1000
0.0001	1000
0.0002	1000
0.0003	1000
0.0004	1000
0.0005	1000
0.0006	1000
0.0007	1000
0.0008	1000
0.0009	1000
0.001	1000
0.0011	1000

Figure F-6: Table result viewer control.

F.3.2. Line Graph

During the initial stages of development of the application, a purpose-built .NET line graph control was used. This control served the purpose that it was required for, i.e. displaying readable graphs that could be used in reports and presentations. However, the control had minor layout bugs and lacked the formatting features of a full graphing package.

The reason behind creating a custom graphing control was that there were no suitable free-to-use controls that could be easily integrated into the environment at the time. However, upon the release of version 4.0 of the .NET framework, a chart control was added to the .NET core libraries which could be used to display numerous types of graph. Due to the loosely coupled architecture used around the result viewer system within the package, it was relatively easy to swap in the new .NET chart control in place of the custom line graph control. The new control, shown in Figure F-7, provides a vast array of formatting options for graphs and has built in methods to export the displayed graph to a file. Using this control also allows the application to be expanded in the future to include other types of graph.

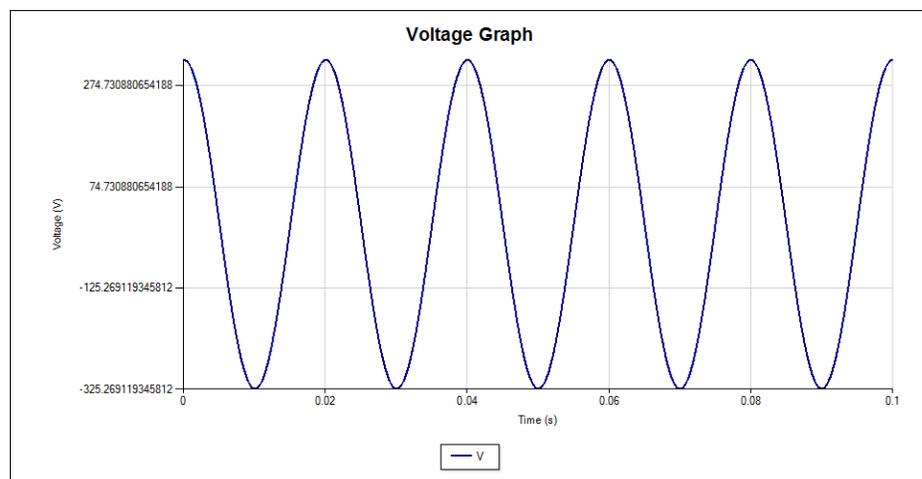


Figure F-7: Illustration of a line graph produced by the .NET 4.0 Chart control.

F.3.3. Pie Chart

As with the line graph result viewer, pie chart rendering was initially carried out using a custom-made control. Manually drawing a pie chart to screen was significantly less complex than a line graph because the Windows drawing libraries contain built-in function calls to

draw or fill a pie chart segment on screen. Despite the good quality of chart image obtained, it was decided to swap the pie chart control for the built-in .NET version when it became available. This removed the need for any further development work to be carried out in future on the drawing of charts.

The pie chart result viewer displays a snapshot in time of the state of a system. The result viewer control therefore requires a method of allowing the user to select which instant in time that the chart control should provide an illustration of. As shown in Figure F-8, this was implemented by providing a scrollbar to select a particular time to view.

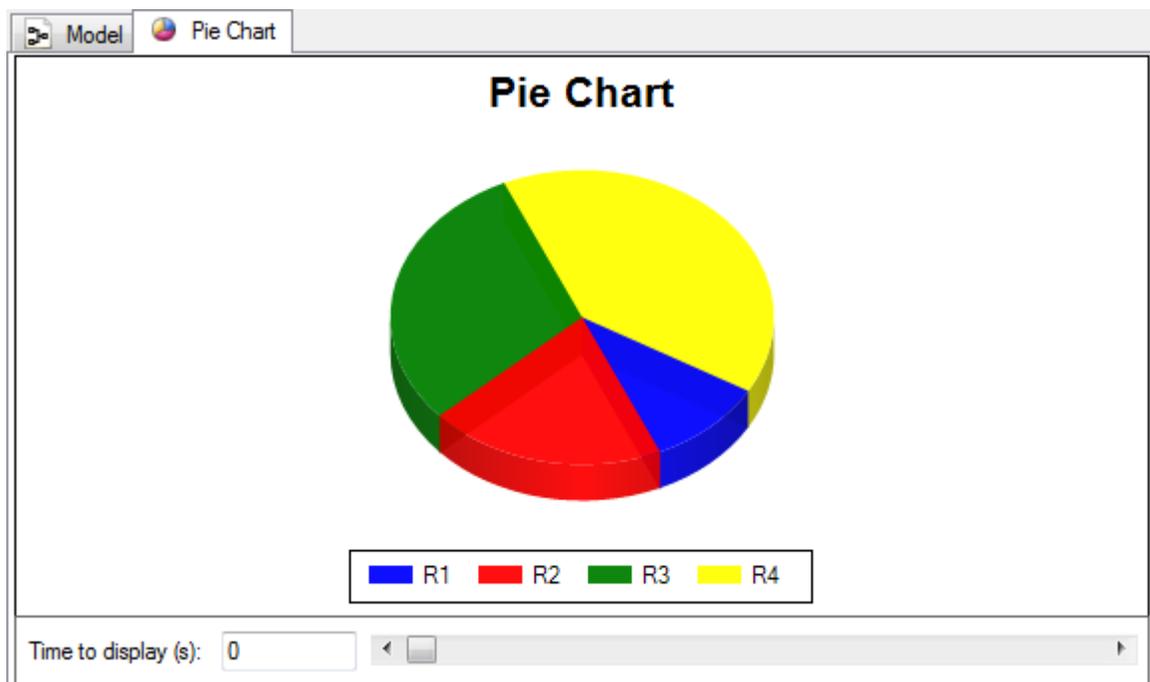


Figure F-8: Illustration of the pie chart result viewer included in the package.

F.4. Parameter Value Editors

In section 3.4.1 of the application design, a number of different configurable parameter types were proposed for components. Within the graphical user interface, a method was required to display all of the configurable parameters of a component, both during the initial creation of components and for editing purposes when a component was used within a simulation.

A customised implementation of the .NET *ListBox* component was used for the purpose of displaying lists of configurable parameters on screen. The list uses a custom drawing method to display the list of parameters by name, type and value as shown in Figure F-9.

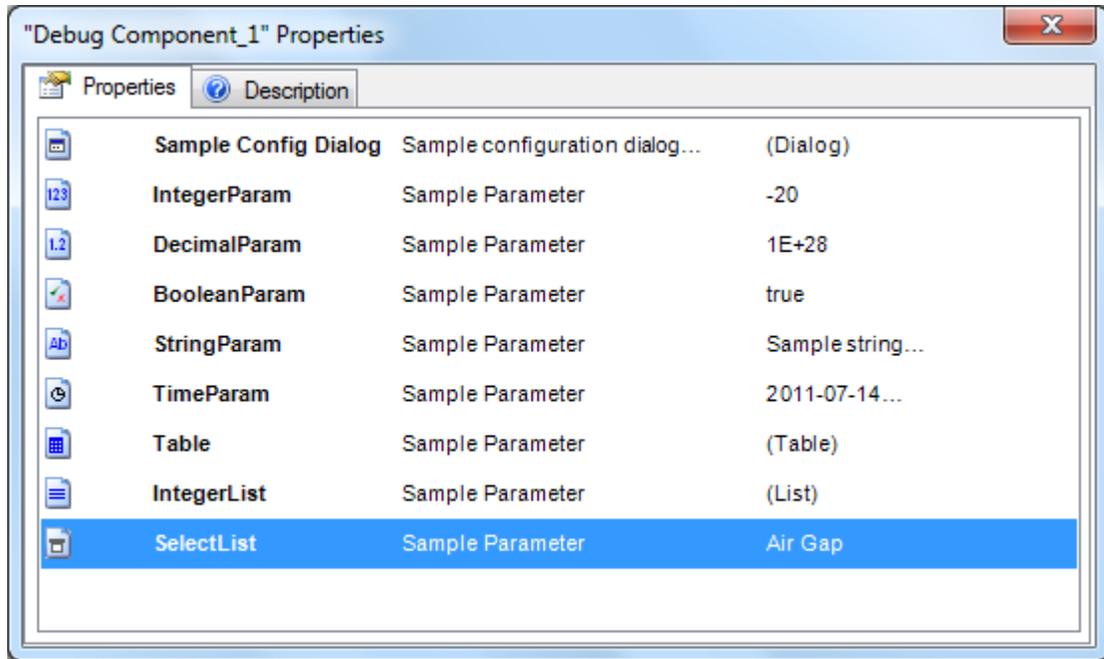


Figure F-9: Configurable parameter list control shown within the component properties viewer.

A number of user interfaces were developed to allow editing of configurable parameter values. Some parameters – the Integer, Decimal, Boolean, String, Time and List types – use the same editor dialog in all parts of the application. The Table and Selection List parameter types require extra user interface options at component design over the options that are provided during normal model editing and therefore these parameter types have two version of the editing dialog. The custom configuration parameter type loads a dialog from an external DLL file for its configuration and therefore no edit-time dialog is required for this parameter type. A dialog is provided at component design time to allow the class name of the custom configuration dialog to be specified.

The Integer, Boolean and String parameter types are edited using the dialogs shown in Figure F-10, Figure F-11 and Figure F-12 respectively.

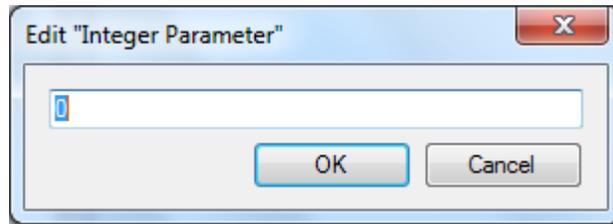


Figure F-10: Integer parameter editing dialog.

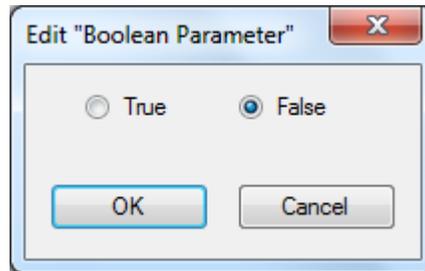


Figure F-11: Boolean parameter editing dialog.

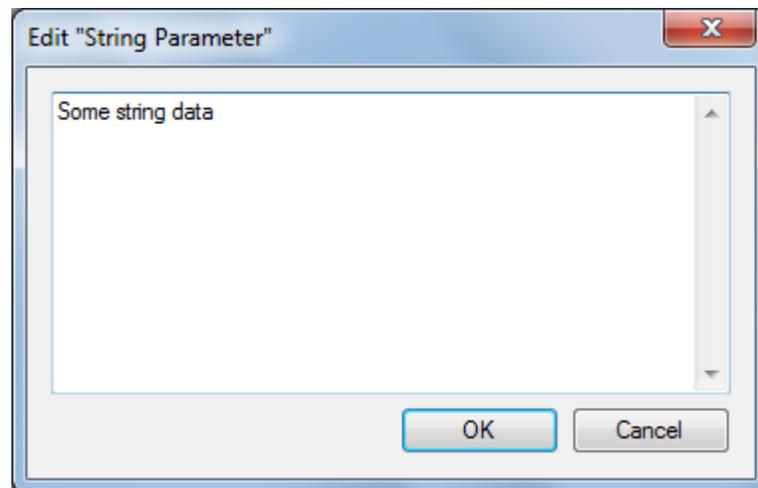


Figure F-12: String parameter editing dialog.

The Decimal parameter editing dialog, shown in Figure F-13 provides extra editing options so that the value may be specified in Standard Notation, Scientific Notation or set to Infinity. It should be noted that “Infinity” is supported as a native feature of the .NET double-precision floating point type and is treated as a special numeric value of this type.

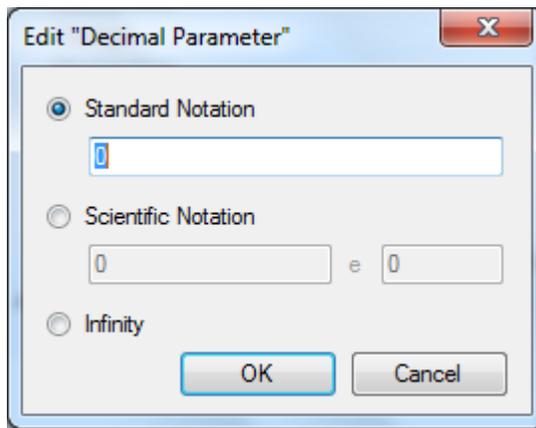


Figure F-13: Decimal parameter editing dialog.

The time parameter editing dialog, shown in Figure F-14, contains three fields which allow for the entry of different parts of the time parameter. Standard .NET date and time controls allow for the entry of the date and time components and a separate Numeric field is provided for entry of the nanoseconds component of the time. The nanoseconds field contains validation logic to ensure that the value is within the range 0 → 999,999,999.

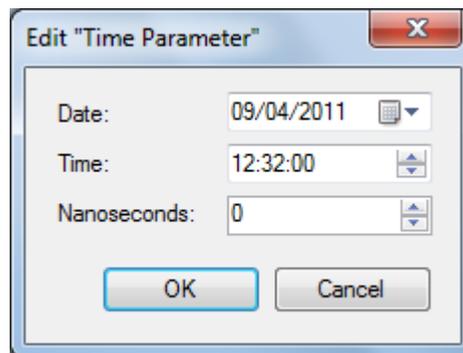


Figure F-14: Time parameter editing dialog.

The list parameter editing dialog, shown in Figure F-15, uses a combination of the .NET *DataGridView* control which displays data in tabular format and some of the custom parameter editors described above in order to edit tabular data. The dialog also provides buttons which allow list items to be added, removed and re-ordered. When editing lists of String, Integer or Decimal values, the list editor operates in a spreadsheet-style mode, where items are edited directly by the user. Validation is performed on values that are entered to

ensure that they are compatible with the underlying data type. Should the value not be compatible, no change is made to the edited list item and a warning is displayed to the user.

When editing lists of Boolean parameters, the grid cells become drop-down menus containing the entries “true” and “false” from which the value can be selected. When editing Time parameters, the cells display the time value currently set in each list item. To edit the value, the user must click on a button within the cell which displays the full time value editor shown in Figure F-14.

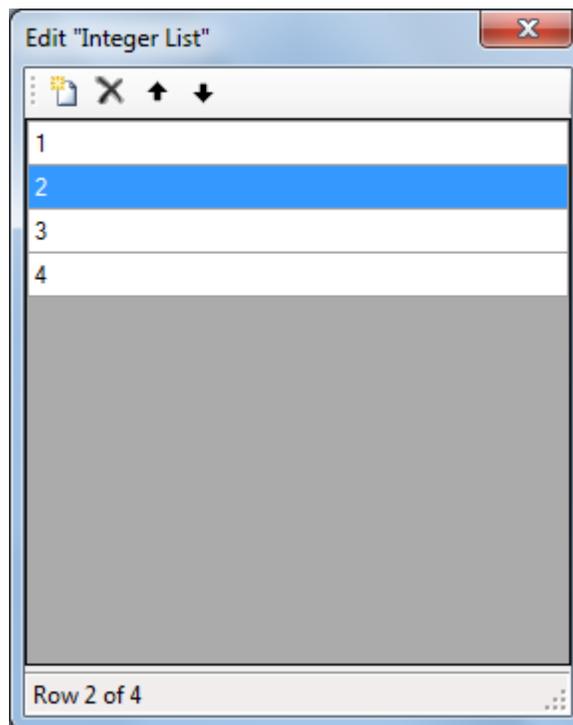


Figure F-15: List parameter editing dialog.

The table parameter editor, shown in Figure F-16 uses the same principle for editing data values as the list editor control but provides an additional option to import tabular data from a spreadsheet file. The import option validates data to ensure that each column is in the correct format for that column’s data type before performing the import.

An enhanced version of the table editing dialog is provided for use in component design mode to allow the structure of the table as well as the data that it contains to be edited. The

additional options provided in this enhanced version of the dialog, shown in Figure F-17, are the capabilities to add, re-order and delete columns.

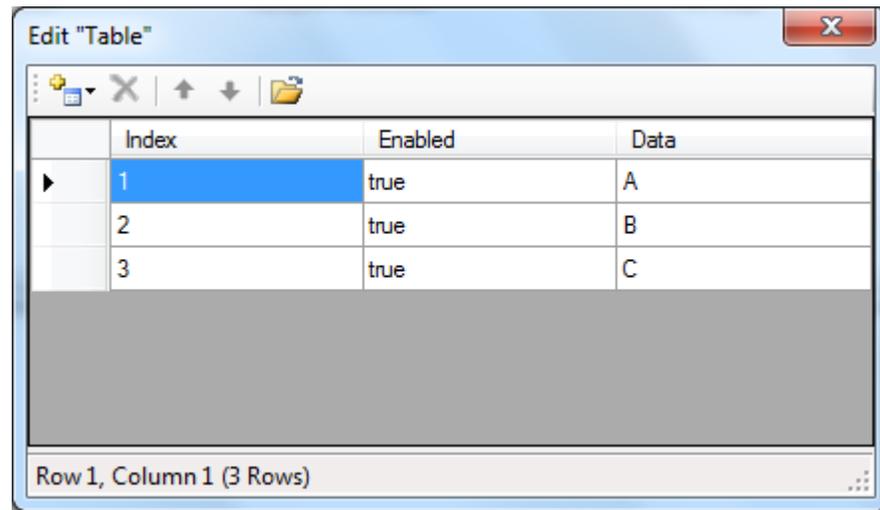


Figure F-16: Table parameter editing dialog in normal operating mode.

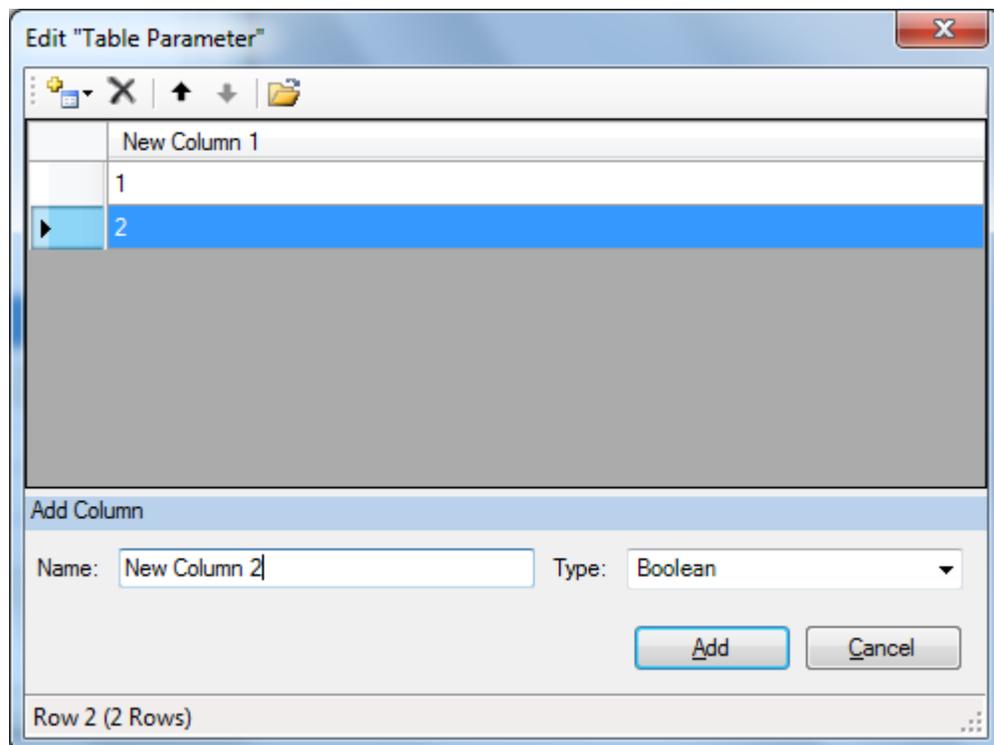


Figure F-17: Enhanced version of the table editing dialog for use at component design time. Options to edit the table structure are also provided.

The selection list parameter editor, shown in Figure F-18 provides users with a single drop-down list from which to select a value for the parameter. The enhanced version of the editor, shown in Figure F-19, which is used at component design time provides additional options to add or remove entries from the list. Additionally, an option is provided to import a list of items from a text file. The file can either be a plain-text file containing one list entry per line, or a CSV spreadsheet. If a spreadsheet file is used, list entries are imported from the first column of the spreadsheet only – the remainder of the columns are ignored.

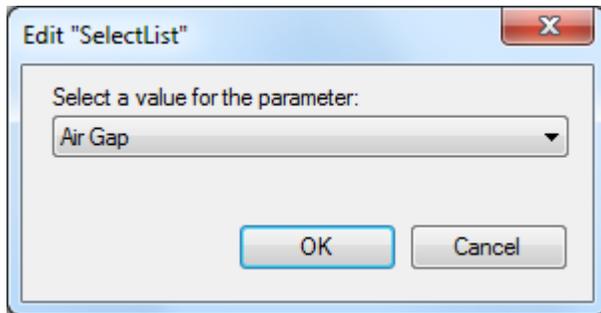


Figure F-18: Selection list editing dialog.

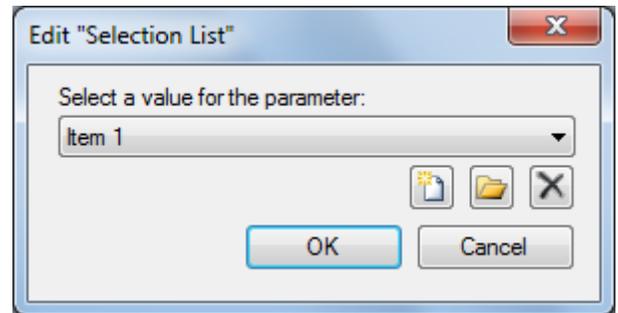


Figure F-19: Enhanced version of the selection list editing dialog used at component design time.

While the custom configuration dialog parameter type uses a custom dialog to allow the user to edit the parameter value during the editing of a model, a configuration dialog of the format shown in Figure F-20 is used at component design time to specify the name of the .NET class containing the dialog that should be displayed to configure the parameter. The dialog also provides an option to enter a URL where the DLL file for the custom configuration dialog can be downloaded from when running the application if it is not installed. Buttons are provided to allow the loading of the dialog and the download URL to be tested.

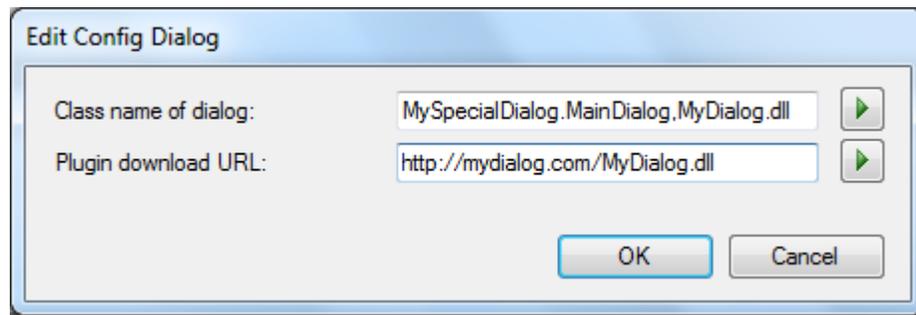


Figure F-20: Custom configuration parameter editing dialog.

F.5. Graphical Model Editor

Alongside the central simulation engine, the graphical model editor is one of the key features of the modelling package. This editor provides a visual method for the creation and editing of system models within the package. Due to the bespoke nature of the graphical editor, there were no off-the-shelf software components that were suitable for use in its implementation. The editor was therefore purpose built for this application.

F.5.1. Overview

The graphical model editor that was provided within the package originally had the requirement of being able to place components on a canvas and create connections between them in order to create a system model. As the design of the package progressed, the additional requirement of being able to add simple shapes and text for the purpose of annotating diagrams was added.

The discussion in the remainder of this section focuses on the implementation of the editing canvas itself. However, to provide context for the description of the implementation of the canvas, a short overview of the functionality of the editor will be provided.

The annotated screenshot in Figure F-21 shows the main editing interface of the application. The central feature within this interface is the editing canvas itself **(1)**. The tool selection palette **(2)** is used to select which editing tool is used by the canvas at any given time. The hint status bar **(3)** is used to provide hints to the user while performing editing actions within the canvas. For example, in this case after selecting a pin on a component, the hint is

provided to select a second pin to complete the connection. The component library browser (4) provides a list of all of the components currently loaded from library files which can be used in the model being edited.

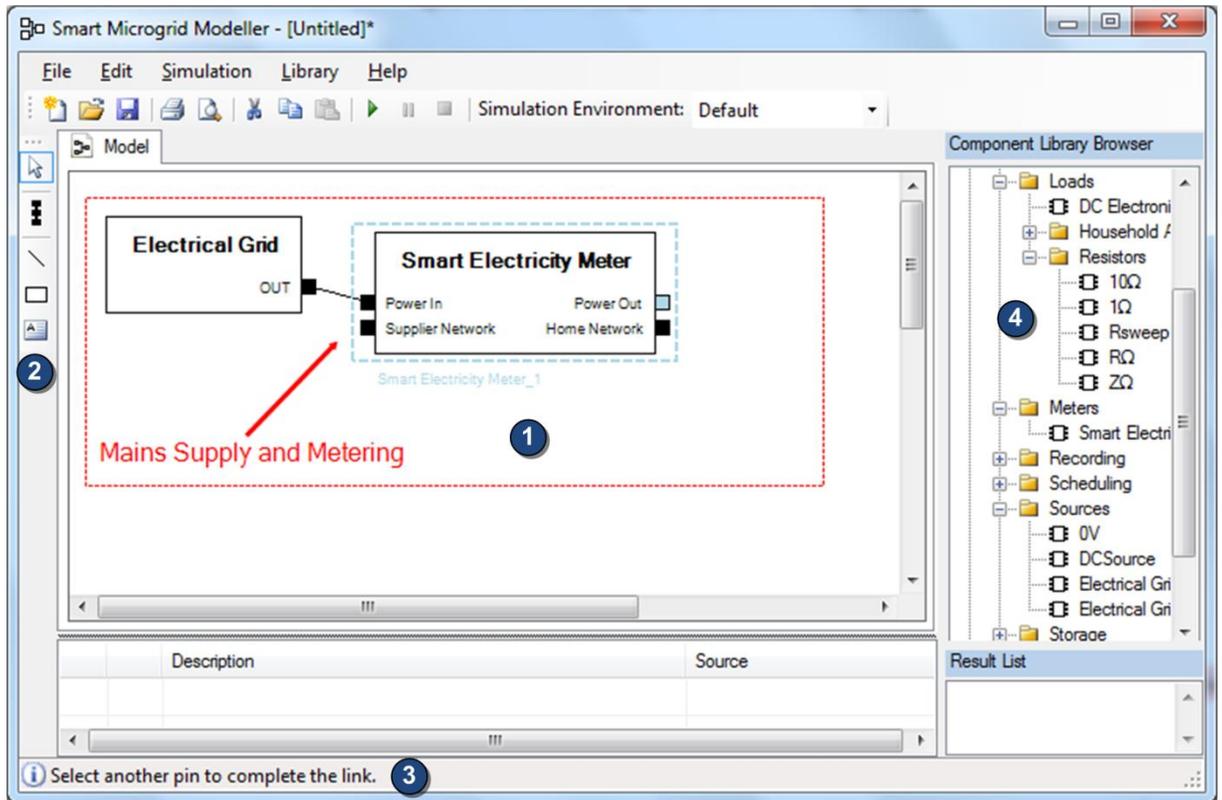


Figure F-21: Annotated diagram illustrating the main features of the graphical model editor.

The editing canvas supports seven basic shape types. These are: nodes, component instances, connections between components, illustrative rectangles, illustrative lines and illustrative text. Components are added to a model by dragging the relevant component from the library browser onto the editing canvas. Connections between components are created by using the arrow tool to select the component pins that form the start and end of the connection. Intermediate routing points can be added by right-clicking on a line. Additionally, an auto-route option is included.

Nodes – special electrical or thermal components – are created by selecting the node tool. Selecting this tool will display a pop-up menu to allow the user to choose which type of node

to create. Clicking on the diagram with the node tool active will create the selected node type on the diagram. Shapes – rectangles and lines – are created by selecting the relevant tool and clicking and dragging at the appropriate location on the diagram. Context menus provide options for formatting lines and rectangles. Selecting the text tool and clicking on the diagram will add text at the selected location. A context menu option is provided to edit and format text.

F.5.2. General Design Considerations for Editor

During the process of designing the editor, a number of prototype editors were developed in order to determine the best method to use to implement the editor. Two broad categories of implementation emerged from this process: 1) an editor created using a Windows Forms Panel Control with child components to represent the shapes; 2) an editor created using a Windows Forms Panel Control using manually programmed user interaction and drawing behaviour.

The former approach relies on using individual windows forms components to represent each shape on the diagram which offers the benefit of a reduction in implementation time. This is due to the fact that the majority of the drawing and user input processing functionality is already provided by the Windows Forms Framework. However, the critical drawback to this approach that resulted in the latter implementation technique being selected was that the Windows Forms library does not provide any built in controls which can easily represent a line. Therefore, the drawing of connections between components and illustrative lines had to be carried out manually by painting directly to the panel control containing the diagram. Using this drawing method resulted in a large amount of flicker when moving components and a significant lag in scrolling around the diagram.

The latter approach was chosen and uses a Windows Forms Panel (an empty, scrollable area within a form) as the parent control for the editor but uses completely custom logic to draw the visible portion of the diagram on the surface of the panel and to handle user input to the panel. This method was chosen because the initial prototype developed provided a much better editing experience than the Windows Component based method. Despite the

disadvantage of a slightly longer implementation time, it was felt that this editor architecture had significant benefits both in terms of usability and in scope for future enhancements.

F.5.3. Editor Design

Figure F-22 illustrates the design of the model editing canvas and its associated user interface elements. The main editor class, *AdvancedDiagramEditor*, implements two interfaces. The *IEditPane* interface is used by the main application to pass commands to the editor, to access the model being edited and to obtain status information from the editor such as whether or not a copy or paste operation can be carried out on the editor. The *IShapeHost* interface is used to provide a set of call-back functions that shapes hosted within the editor can use to control the editor and provide feedback about operations that have been carried out as a result of user interaction with shapes. The *EnhancedDiagramEditor* class holds a reference to the *IDiagramEditorToolbar* interface to provide access to the user-selected editing tool. This interface is implemented by the *EnhancedDiagramEditorToolbar* class which implements the graphical palette **(2)** on Figure F-21. The editor also holds a reference to the *IHintStatusBar* interface which provides a generic interface for a status bar providing editing hints. This interface is implemented directly by the *ApplicationInterface* class which is the main Windows Form shown in Figure F-21. Calling any of the methods in the *IHintStatusBar* class changes the text shown at the bottom of the form **(3)**.

Shapes used within model diagrams are implemented by the *ComponentShape*, *LinkShape*, *RectangleShape*, *LineShape* and *TextShape* classes, each of which implements the *IDiagramShape* interface. This interface defines a set of methods which allow the diagram editor to control the drawing of shapes and the communication of user input actions to each shape. The *EnhancedDiagramEditor* class stores a collection of *IDiagramShape* objects which make up the model currently being edited.

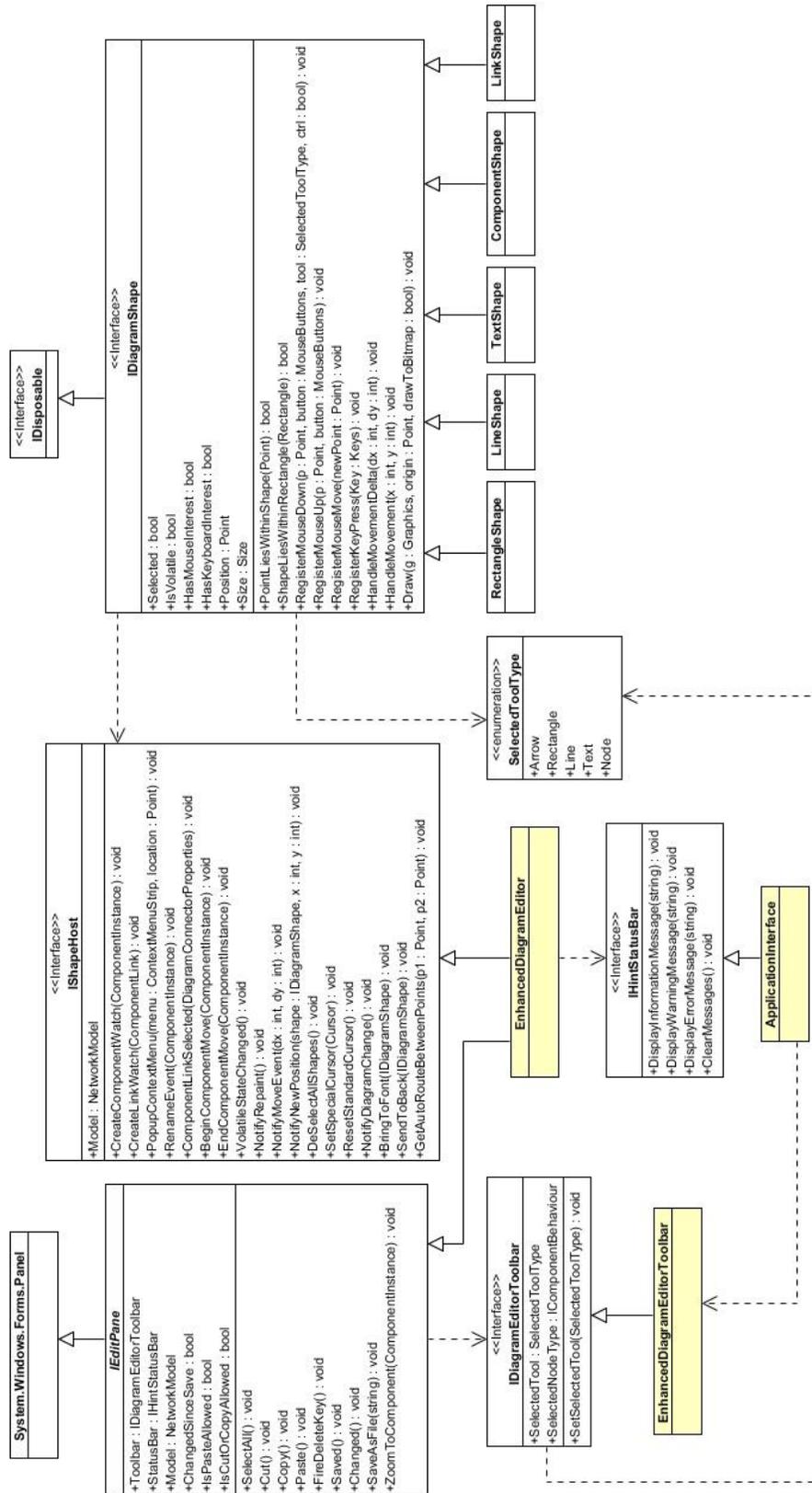


Figure F-22: Model editor class diagram.

F.5.4. User Input Handling

User input actions can be broadly described in three categories: direct interaction with shapes including clicking on shapes, dragging shapes and pressing keys while a shape is selected; group selection of shapes by dragging a selection box on the diagram around a group of shapes; and indirect movement of shapes which involves dragging one shape within a selected group to move the entire group.

Direct interaction with shapes is handled using a multi-stage process. When a mouse button is pressed within the editor panel, the editor queries each shape in the diagram to determine if the point clicked lies within the shape's bounds. If this is the case then the editor issues a "Mouse Down" event to the shape with the selected co-ordinates. Logic within each shape then determines whether or not any action should be taken according to the position of the mouse click. An example of a shape-specific action that could be taken would be if one of the pins on a component shape was clicked, the pin would be selected and marked as the beginning of a new connection between components. Examples of non-shape-specific actions to take would be to mark the shape as selected if the mouse is clicked within its bounds.

In order to reduce the number of computational operations performed to process user interaction, only "Mouse Down" events are universally reported to all shapes. Two flags *HasMouseInterest* and *HasKeyboardInterest* are made available in the *IDiagramShape* interface which informs the editor whether or not a particular shape would like to receive "Mouse Move", "Mouse Up" and "Key Press" events. For all of the shapes currently implemented, shapes set their "Mouse Interest" flag upon a "Mouse Down" event and clear the flag upon a "Mouse Up" event. The "Keyboard Interest" flag is set when shapes are selected and cleared when they are no longer selected.

Upon processing a "Mouse Down" event, if the editor detects that the mouse has been pressed on an area of the canvas that does not lie within the bounds of any shape, then the group selection process begins. The first step of this process is to de-select all shapes within the diagram by setting the *Selected* property of each *IDiagramShape* object to false. As the mouse is moved, a box is drawn which can be used to select a group of shapes. When the

mouse is released, the editor queries the *ShapeLiesWithinRectangle* method of each shape within the diagram to determine which shapes lie within the user-drawn box. Any shapes which lie within the box have their *Selected* property set to true. As well as providing a group selection technique, this process also provides the benefit of de-selecting all selected shapes if the user clicks the mouse on a blank area of the panel.

Group movement of objects requires interactions between the logic built into shapes and the logic within the editor, unlike shape-specific operations such as re-sizing which can be handled directly by internal shape logic. In order to support movement of both single and multiple objects, the editor's design enforces a particular pattern for handling the movement of shapes. If a shape detects that user input is causing it to move (for example, in the case of a component shape, the mouse is pressed within the body of the shape and is moving) then rather than deal with the movement operation directly, the shape is required to invoke the call-back method *NotifyMoveEvent* within the *IShapeHost* interface, reporting the amount by which the shape has been moved. When this method is invoked, the editor will iterate through all selected shapes on the diagram and call the *HandleMovementDelta* for each shape with the difference in X and Y co-ordinates by which to move the shape. This logic within this method is then responsible for moving each shape and re-drawing the shape at the new location.

F.5.5. Drawing

The shape-drawing logic within the editing component uses a double-buffering system to reduce flicker during the drawing process. Further efficiency is added to the drawing process by only drawing shapes which need to be re-drawn each time the editor is drawn. The process of drawing the contents of the editor can be initiated by a number of events: the operating system requesting a re-draw of the component; the component scrollbar positions being changed by the user; and a component's code requesting a re-draw (for example, when a component becomes "Selected" it may change its colour or paint a selection box around the component or when a shape is moved it will be required to be re-drawn at its new location).

Each *IDiagramShape* instance has a Boolean property, *IsVolatile*, that is used by shape to indicate when it is in a state where it is likely to need re-drawn frequently. Examples of such states are when the shape is being moved or when the shape is selected. Each time that the *IsVolatile* property of a shape is changed, the shape invokes a call-back method on the *IShapeHost* interface to inform the editor of the change. Upon such a change of volatile state, the editor will create a bitmap buffer on which all non-volatile shapes are drawn by invoking the *Draw* method of each shape. Each time a re-draw of the diagram is requested, the editor will first re-draw the bitmap buffer of non-volatile shapes to the editing surface, followed by each non-volatile shape. This process improves the performance of the editor's drawing process while moving shapes, almost entirely eliminating flicker while moving shapes.

F.6. Graphical Component Editor

As well as providing an editor for creating system models, an editor was also required within the application to create and edit individual components for use within the models. This editor is responsible for four main aspects of component configuration: naming, description and categorisation of components; defining the number, name and type of pins on components; defining and initialising configurable parameters within components; and defining component logic through C#, VB.NET or Mathematical scripting code.

A single user interface was created to handle each of these four configuration tasks. The interface is split into four tabs, each handling an aspect of component configuration. Before displaying the interface, the user is prompted to either select an existing library of components or to create a new empty file. The contents of the selected file are then loaded into a collection of "Component Behaviour" classes¹¹ for use within the editor. Any changes made to components within the editor are made in-memory to these classes which are then saved back to the file when the user selects the save option.

The first stage of component configuration is shown in the annotated screen-shot in Figure F-23. The configuration interface displays a list at the top of the screen **(1)** containing all of the

¹¹ See section 3.2.2 for a full description of the "Component Behaviour" class.

components that are currently available in the loaded library file. When a component is selected within this list, its configuration is loaded into the main editor tabs. A set of toolbar buttons **(2)** allows users to add and remove components as well as save any change to the library file. The text boxes **(3)** allow the name, category, version number and description of the component to be edited. Checkboxes allow for configuration of advanced options within the component. A preview of the component appearance is provided **(4)** by using a modified version of the model editor described in section F.5. This preview control has been modified to make it read-only and is populated with only one component.

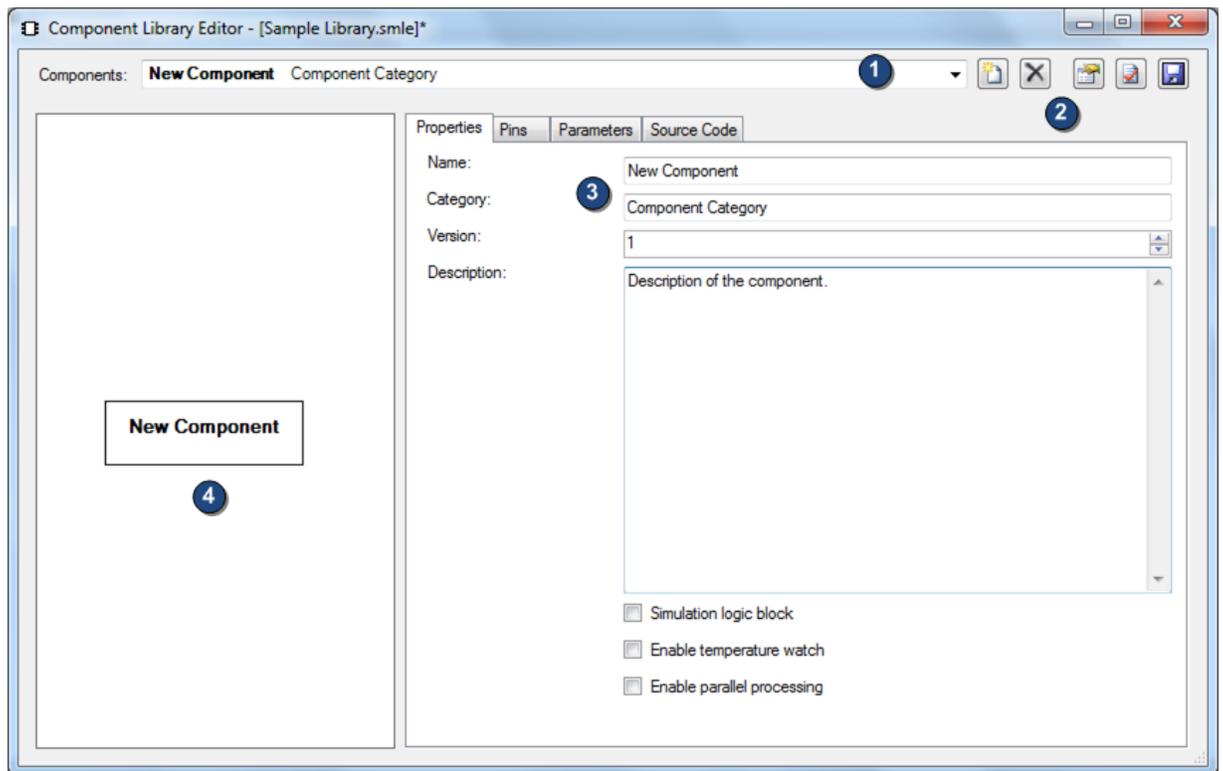


Figure F-23: Properties page from the component library editor.

The second stage of component configuration is shown in the annotated screen-shot in Figure F-24. This tab allows for the configuration of the pins on the component. A list **(1)** of the pins currently on the component is shown, along with the properties of each pin. Buttons **(2)** are provided to add or remove pins. When a pin within the list is selected, its properties are shown at the bottom of the screen **(3)** for editing. Options are provided to edit the pin

Name, Type and Placement on the component. Any changes made to pins are reflected instantly on the component preview.

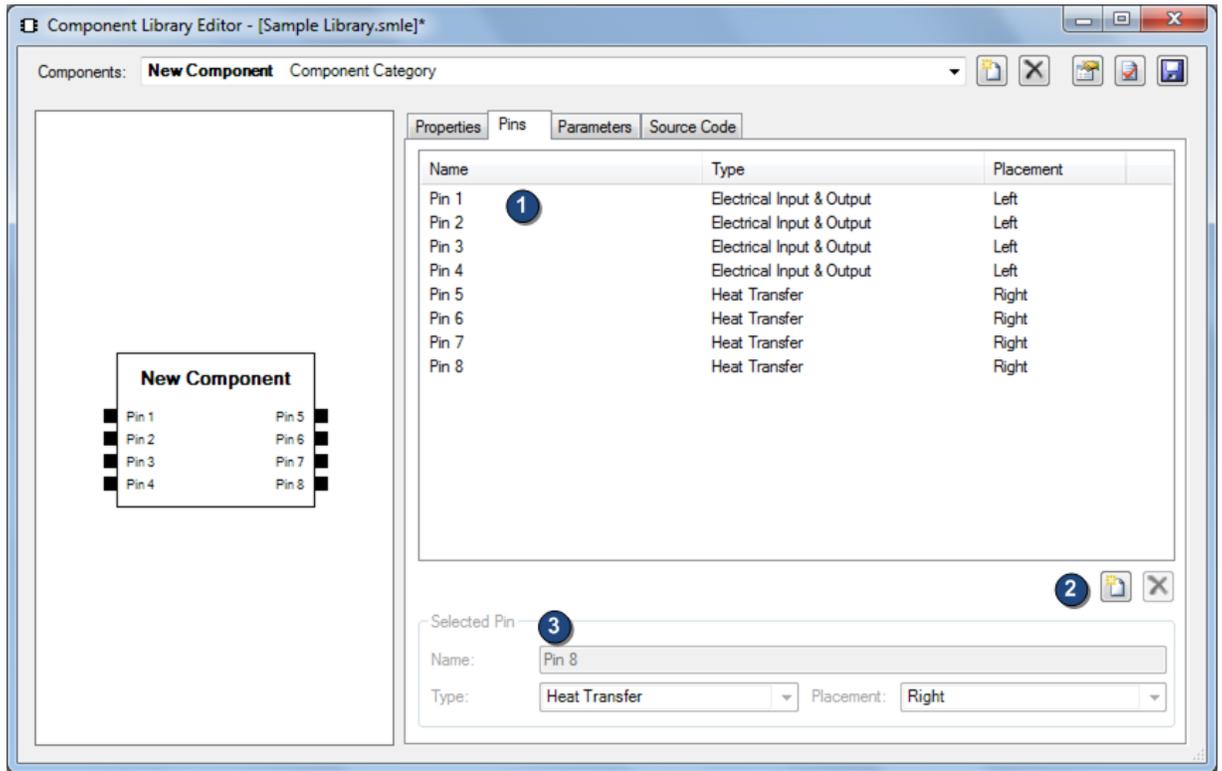


Figure F-24: Component library editor pin configuration page.

The third page of the library editor, shown in Figure F-25, configures the parameters that are made available within the component which allow for user configuration of each component's behaviour. Similarly to the pin editor, this editor displays a list of the current parameters available within the component (1) and provides buttons to create or delete parameters (2). Selecting the "New Parameter" buttons displays a menu of available parameter types to create – the type cannot be changed after creation. Buttons are also provided to duplicate existing parameters and change the order of parameters within the list. The properties of the selected parameter can be edited using the controls at the bottom of the screen (3). The default value of a parameter is modified by selecting the button beside

the value **(4)**. Selecting this button displays the relevant editing dialog¹² for the selected parameter type.

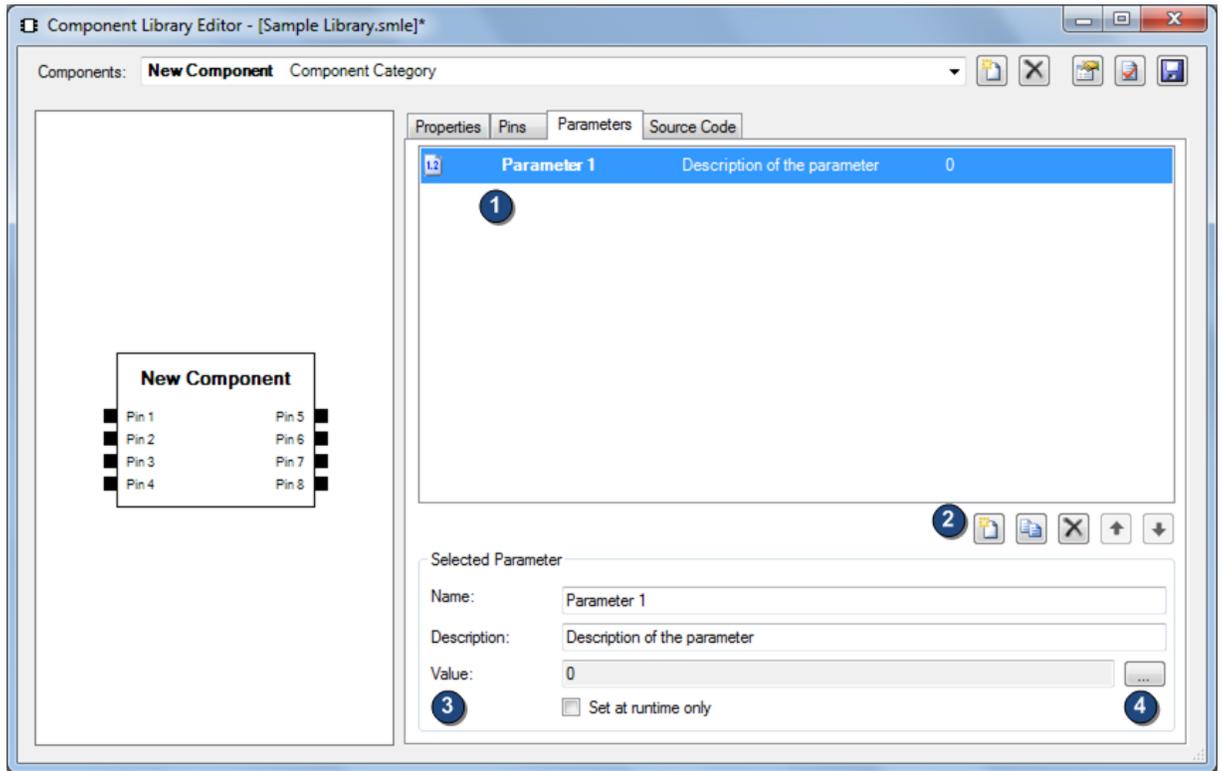


Figure F-25: Parameter editing page of the library editor.

The final configuration page within the graphical component editor, shown in Figure F-26, provides a source code editor **(1)** for defining the component logic. The source code editor that is used in this interface is a third-party component taken from the open-source SharpDevelop IDE [138]. The editor provides useful built-in features for code editing such as undo/redo, syntax highlighting, find and replace and code region folding. Minor changes were made to the built-in functionality to add syntax highlighting for the scripting API elements, the custom pre-processor statements that were introduced and the Mathematical Mark-up language.

A menu of available programming languages above the source code editor **(2)** controls the syntax highlighting and code folding strategies that are in use within the editor. The selected language is also saved within each component's behaviour so that the simulator knows which

¹² Parameter editing dialogs are described in section F.4.

compiler to use to compile the code when running simulations. A syntax checker **(3)** is included within the editor. This syntax checker uses the compiler for the selected programming language and reports any compilation errors along with their line numbers in the information box **(4)**. Additionally, the editor supports the highlighting of lines which generate compiler errors or warnings. The toolbar **(5)** provides common text editing operations. Additionally the toolbar provides access to API documentation and a menu of commonly used code-snippets. Code snippets are defined in all programming languages and code for the current language is inserted when a snippet is chosen.

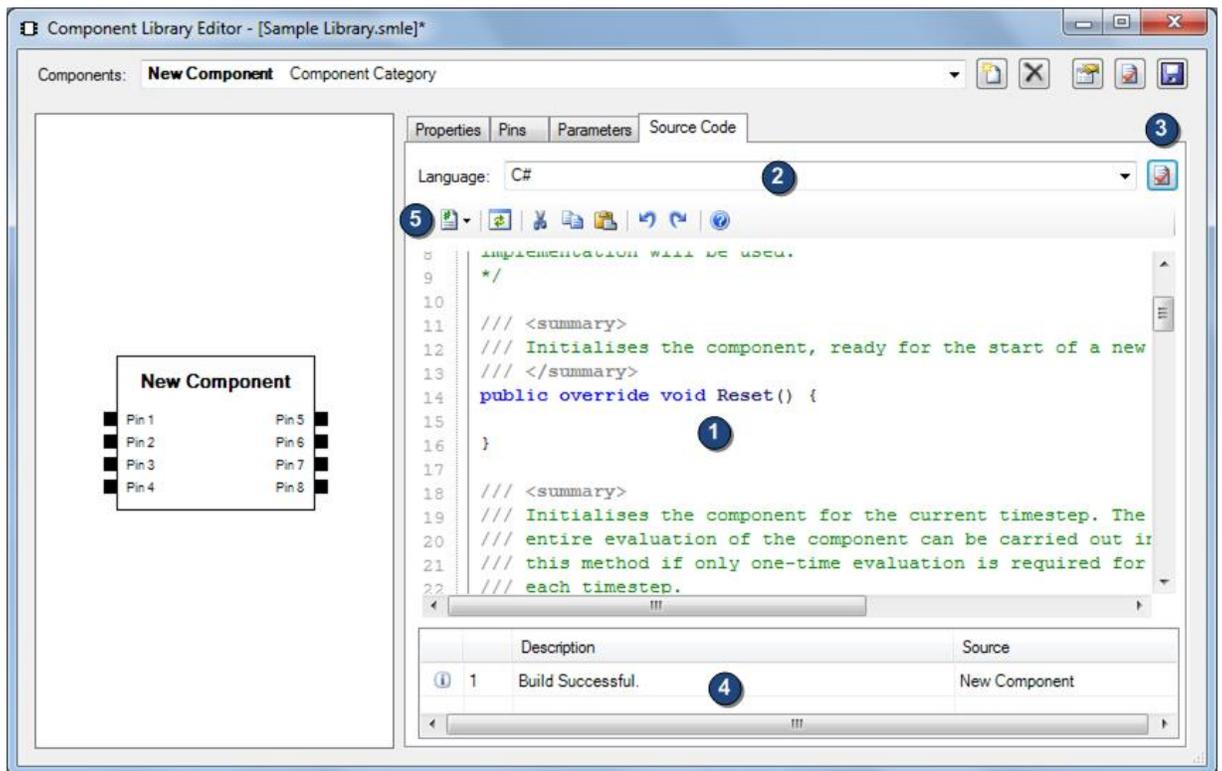


Figure F-26: Source code editing page of the component library editor.

The main model editor stores all component libraries in memory and therefore any changes to a library file using the component library editor require component libraries to be re-loaded in the model editor. To simplify this process, a “re-load” option is provided in the model editor to refresh all libraries. This option also invokes an upgrade process on the currently open model if any of its components have been edited.

F.7. Implementation Summary

The design and implementation described up to this point is the culmination of multiple iterations of re-design and re-implementation that were carried out. First-hand experience of using the package provided a helpful insight into defects in the software and useful enhancements. Continuous testing of the software throughout the implementation process was also extremely useful in quickly resolving defects and immediately identifying an errors that were introduced by the implementation of new features.

The implementation of the package that has been described in this appendix was concluded in a state where the package was suitable for use for this project. While there are endless future enhancements that could be carried out to the program to improve its functionality, usability and performance, the current implementation was felt to offer sufficient features to continue with the implementation of a library of components within the package to enable the modelling of domestic smart grid systems.

References

- [1] L. Maclsaac and A. Knox, "Domestic end-use simulation of smart grid technologies," in *Innovative Smart Grid Technologies (ISGT Europe), 2011 2nd IEEE PES International Conference and Exhibition on*, Manchester, 2011, pp. 1-6.
- [2] L. Maclsaac and A. Knox, "Improved Maximum Power Point Tracking Algorithm for Photovoltaic Systems," in *International Conference on Renewable Energies and Power Quality*, Granada, Spain, 2010.
- [3] L. Maclsaac and A. Knox, "A New Algorithm for Obtaining the Operating Point of Photovoltaic Systems," *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, vol. 226, no. 6, 2012.
- [4] HM Government, "Postnote: Renewable Heating," Parliamentary Office of Science and Technology, 2010
- [5] HM Government, "Digest of UK Energy Statistics (DUKES)," Department of Energy and Climate Change, 2010
- [6] European Commission, "UK Energy Mix Fact Sheet," 2007
- [7] HM Government, "The UK Low Carbon Transition Plan: National strategy for climate and energy," DECC, 2009
- [8] Scottish Government, "Draft Electricity Generation Policy Statement 2010: Scotland – A Low Carbon Society," 2010
- [9] Scottish Government. "National Indicators: Increase Renewable Energy Production." Internet:
<http://www.scotland.gov.uk/About/Performance/scotPerforms/indicator/renewable>, 22nd December 2011 [Accessed: 1st May 2012,]
- [10] BBC News. "Scots windfarms paid cash to stop producing energy." Internet:
<http://www.bbc.co.uk/news/uk-scotland-13253876>, 1st May 2011 [Accessed: 21st September 2011]
- [11] A. Al-Kandari, M. Gilany, and A. Shaltout, "A PLC Controller Algorithm for Optimum Operation of Photovoltaic-Battery System," in *Power Engineering, 2006 Large Engineering Systems Conference on*, 2006, pp. 112-118.
- [12] S. S. Choi, K. J. Tseng, D. M. Vilathgamuwa, and T. D. Nguyen, "Energy storage systems in distributed generation schemes," in *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, 2008, pp. 1-8.
- [13] J. Mendez, A. Falcon, and D. Hernandez, "Simulation of Storage Systems for increasing the Power Quality of Renewable Energy Sources," in *International Conference of Renewable Energies and Power Quality*, Granada, Spain, 2010.
- [14] J. A. Carr, J. C. Balda, and H. A. Mantooth, "A Survey of Systems to Integrate Distributed Energy Resources and Energy Storage on the Utility Grid," in *Energy 2030 Conference, 2008. ENERGY 2008. IEEE*, 2008, pp. 1-7.
- [15] E. G. Cazalet, "Role of storage technologies for increased deployment of distributed and renewable resources," in *Innovative Smart Grid Technologies (ISGT)*, 2010, pp. 1-1.

- [16] G. Celli, S. Mocci, F. Pilo, and M. Loddo, "Optimal integration of energy storage in distribution networks," in *PowerTech, 2009 IEEE Bucharest*, 2009, pp. 1-7.
- [17] R. Datta and V. T. Ranganathan, "A method of tracking the peak power points for a variable speed wind energy conversion system," *Energy Conversion, IEEE Transactions on*, vol. 18, no. 1, pp. 163-168, 2003.
- [18] S. Armstrong and W. G. Hurley, "Self-regulating maximum power point tracking for solar energy systems," *UPEC 2004: 39th International Universities Power Engineering Conference, Vols 1-3, Conference Proceedings*, pp. 604-609, 2005.
- [19] A. M. A. Mahmoud, H. M. Mashaly, S. A. Kandil, H. El Khashab, and M. N. F. Nashed, "Fuzzy logic implementation for photovoltaic maximum power tracking," in *Industrial Electronics Society, 2000. IECON 2000. 26th Annual Conference of the IEEE*, 2000, pp. 735-740 vol.1.
- [20] M. A. Vitorino, L. V. Hartmann, A. M. N. Lima, and M. B. R. Correa, "Using the model of the solar cell for determining the maximum power point of photovoltaic systems," in *Power Electronics and Applications, 2007 European Conference on*, 2007, pp. 1-10.
- [21] I. H. Altas and A. M. Sharaf, "A novel on-line MPP search algorithm for PV arrays," *Energy Conversion, IEEE Transactions on*, vol. 11, no. 4, pp. 748-754, 1996.
- [22] K. Yeong-Chau, L. Tsorng-Juu, and C. Jiann-Fuh, "Novel maximum-power-point-tracking controller for photovoltaic energy conversion system," *Industrial Electronics, IEEE Transactions on*, vol. 48, no. 3, pp. 594-601, 2001.
- [23] K. H. Hussein, I. Muta, T. Hoshino, and M. Osakada, "Maximum photovoltaic power tracking: an algorithm for rapidly changing atmospheric conditions," *Generation, Transmission and Distribution, IEE Proceedings-*, vol. 142, no. 1, pp. 59-64, 1995.
- [24] J. Youngseok, S. Junghun, Y. Gwonjong, and C. Jaeho, "Improved perturbation and observation method (IP&O) of MPPT control for photovoltaic power systems," in *Photovoltaic Specialists Conference, 2005. Conference Record of the Thirty-first IEEE*, 2005, pp. 1788-1791.
- [25] X. Liu and L. A. C. Lopes, "An improved perturbation and observation maximum power point tracking algorithm for PV arrays," in *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*, 2004, pp. 2005-2010 Vol.3.
- [26] S. Parmigiani, D. Zani, C. Invernizzi, A. Mazzu, V. Villa, and A. Lezzi, "A biomass powered Ringbom-Stirling engine for developing countries: a low-budget solution for distributed electricity generation," in *International Conference of Renewable Energies and Power Quality*, Granada, Spain, 2010.
- [27] D. G. Thombarse, K. H. J. Buschow, W. C. Robert, C. F. Merton, I. Bernard, J. K. Edward, M. Subhash, and V. Patrick, "Stirling Engine: Micro-CHP System for Residential Application," in *Encyclopedia of Materials: Science and Technology Oxford: Elsevier*, 2008, pp. 1-8.
- [28] M. Chen, H. Lund, L. A. Rosendahl, and T. J. Condra, "Energy efficiency analysis and impact evaluation of the application of thermoelectric power cycle to today's CHP systems," *Applied Energy*, vol. 87, no. 4, pp. 1231-1238, 2009.
- [29] X. Gou, H. Xiao, and S. Yang, "Modeling, experimental study and optimization on low-temperature waste heat thermoelectric generator system," *Applied Energy*, vol. 87, no. 10, pp. 3131-3136, 2010.

- [30] EnviroEnergy. "Welcome to Nottingham Green Energy." Internet: <http://www.enviroenergy.co.uk/index2.htm>, 16th July 2004 [Accessed: 21st September 2011]
- [31] D. Wickersham. "Pimlico District Heating Undertaking Presentation." Internet: <http://www.lep.org.uk/uploads/David%20Wickersham%20CityWest%20Homes.pdf>, 6th July 2008 [Accessed: 21st September 2009]
- [32] HM Government, "Low Carbon Transport: A Greener Future," Department for Transport, 2009
- [33] Mayor of London. "An Electric Vehicle Delivery Plan for London." Internet: <http://www.london.gov.uk/sites/default/files/uploads/electric-vehicles-plan.pdf>, May 2009 [Accessed: 10th January 2012]
- [34] J. Gallardo-Lozano, M. I. Milanés-Montero, M. A. Guerrero-Martínez, and E. Romero-Cadaval, "Electric vehicle battery charger for smart grids," *Electric Power Systems Research*, vol. 90, no. 0, pp. 18-29.
- [35] M. Metz and C. Doetsch, "Electric vehicles as flexible loads - A simulation approach using empirical mobility data," *Energy*, 2012.
- [36] N. Hartmann and E. D. Özdemir, "Impact of different utilization scenarios of electric vehicles on the German grid in 2030," *Journal of Power Sources*, vol. 196, no. 4, pp. 2311-2318, 2011.
- [37] Energy Efficiency News. "EU takes first steps towards a smart grid." Internet: <http://www.energyefficiencynews.com/i/2376/>, 7th September 2009 [Accessed: 4th April 2011]
- [38] A. Faruqui, D. Harris, and R. Hledik, "Unlocking the €53 billion savings from smart meters in the EU: How increasing the adoption of dynamic tariffs could make or break the EU's smart grid investment," *Energy Policy*, vol. 38, no. 10, pp. 6222-6231, 2010.
- [39] Centrica. "British Gas plans two million smart meters in British homes by 2012." Internet: <http://www.centrica.co.uk/index.asp?pageid=39&newsid=1970>, [Accessed: 13th April]
- [40] ZigBee Alliance. "ZigBee Smart Energy Overview." Internet: <http://www.zigbee.org/Standards/ZigBeeSmartEnergy/Overview.aspx>, 2011 [Accessed: 12th April 2011]
- [41] Scottish Power. "ScottishPower Ramps Up Smart Meter Trials Following Government Announcement." Internet: http://www.scottishpower.com/PressReleases_1968.htm, 2nd December 2009 [Accessed: 19th March 2011]
- [42] Landis+Gyr. "Scottish and Southern." Internet: http://www.landisgyr.com/en/pub/products_and_services/case_studies/scottish_and_southern.cfm, 2009 [Accessed: 19th March 2011]
- [43] National Grid. "OnStream launches revolutionary new smart meters." Internet: <http://www.nationalgrid.com/uk/Media+Centre/PressReleases/2010/20.04.10+Onstream.htm>, 20th April 2010 [Accessed: 18th March 2011]
- [44] Energy Choices. "Economy 7 energy tariffs." Internet: <http://www.energychoices.co.uk/economy-7.html>, 26th May 2009 [Accessed: 12th April 2011]
- [45] A. Faruqui and S. George, "Quantifying Customer Response to Dynamic Pricing," *The Electricity Journal*, vol. 18, no. 4, pp. 53-63, 2005.

- [46] A. Faruqui, R. Hledik, and S. Sergici, "Piloting the Smart Grid," *The Electricity Journal*, vol. 22, no. 7, pp. 55-69, 2009/9// 2009.
- [47] ENSG. "A Smart Grid Routemap." Internet: http://www.ensg.gov.uk/assets/ensg_routemap_final.pdf, February 2010 [Accessed: 18th March 2011]
- [48] D. B. Crawley, J. W. Hand, M. Kummert, and B. T. Griffith, "Contrasting the capabilities of building energy performance simulation programs," *Building and Environment*, vol. 43, no. 4, pp. 661-673, 2008.
- [49] US Department of Energy. "Building Technologies Program: EnergyPlus." Internet: <http://apps1.eere.energy.gov/buildings/energyplus/>, [Accessed: January 30th]
- [50] D. B. Crawley, L. K. Lawrie, C. O. Pedersen, R. K. Strand, R. J. Liesen, F. C. Winkelmann, W. F. Buhl, Y. J. Huang, M. J. Witte, R. J. Henninger, J. Glazer, D. E. Fisher, and D. Shirey, "EnergyPlus: New, Capable and Linked," *Journal of Architectural and Planning Research*, vol. 21, no. 4, pp. 292-302, 2004.
- [51] F. Winkelmann, B. Birdsall, W. Buhl, K. Ellington, A. Erdem, J. Hirsch, and S. Gates, "DOE-2 supplement: version 2.1 E," Lawrence Berkeley Lab., CA (United States); Hirsch (James J.) and Associates, Camarillo, CA (United States)1993.
- [52] D. Jacobson, "Bibliography of BLAST Related Articles," BLAST Support Office, University of Illinois at Urbana-Champaign1986.
- [53] Google. "Google Sketchup." Internet: <http://sketchup.google.com/>, 2011 [Accessed: 1st June 2011]
- [54] US Department of Energy. "EnergyPlus Simulation Software: Learn more about OpenStudio." Internet: <http://apps1.eere.energy.gov/buildings/energyplus/openstudio.cfm>, 7th March 2011 [Accessed: 1st June 2011]
- [55] US Department of Energy. "EnergyPlus Energy Simulation Software: EnergyPlus Graphical User Interfaces." Internet: http://apps1.eere.energy.gov/buildings/energyplus/ep_interfaces.cfm, 7th March 2011 [Accessed: 1st June 2011]
- [56] F. C. Winkelmann, "Modeling windows in EnergyPlus," *Proc. IBPSA, Building Simulation*, 2001.
- [57] D. E. Fisher, R. D. Taylor, F. Buhl, R. J. Liesen, and R. K. Strand, "A modular, loop-based approach to HVAC energy simulation and its implementation in EnergyPlus," in *Proceedings of Building Simulation*, 1999, pp. 1245-1252.
- [58] US Department of Energy, "EnergyPlus Engineering Reference," 2010.
- [59] R. Judkoff and J. Neymark, "Model validation and testing: The methodological foundation of ASHRAE Standard 140," *Transactions - American Society of Heating Refrigerating and Air Conditioning Engineers*, vol. 112, no. 2, p. 367, 2006.
- [60] M. J. Witte, R. H. Henninger, J. Glazer, and D. B. Crawley, "Testing and validation of a new building energy simulation program," *Proceedings of Building Simulation 2001*, pp. 353-360, 2001.
- [61] R. H. Henninger and M. J. Witte, "EnergyPlus Testing with Building Thermal Envelope and Fabric Load Tests from ANSI/ASHRAE Standard 140-2007," 2006.
- [62] R. H. Henninger and M. J. Witte, "EnergyPlus Testing with HVAC Equipment Performance Tests CE100 to CE200 from ANSI/ASHRAE Standard 140-2007," 2009.

- [63] R. H. Henninger and M. J. Witte, "EnergyPlus Testing with HVAC Equipment Performance Tests CE300 to CE545 from ANSI/ASHRAE Standard 140-2007," 2010.
- [64] R. H. Henninger and M. J. Witte, "EnergyPlus Testing with Fuel-Fired Furnace Tests HE100 to HE230 from ANSI/ASHRAE Standard 140-2007 " 2010.
- [65] R. D. Taylor, C. O. Pedersen, and L. Lawrie, "Simultaneous simulation of buildings and mechanical systems in heat balance based energy analysis programs," in *Proceedings of the 3rd International Conference on System Simulation in Buildings*, 1990, pp. 3-5.
- [66] University of Strathclyde. "ESP-r." Internet: <http://www.esru.strath.ac.uk/Programs/ESP-r.htm>, 6th December 2010 [Accessed: 1st June 2011]
- [67] J. Clarke and J. Hensen, "Integrated simulation for building design: an example state-of-the-art system," in *Proceedings International Conference Construction Information Technology*, 2000, pp. 465-475.
- [68] J. Clarke, W. Dempster, and C. Negrao, "The implementation of a computational fluid dynamics algorithm within the ESP-r system," in *Proc. Building Simulation*. vol. 95: Citeseer, 1995, pp. 166-75.
- [69] J. Hensen and J. Clarke, "A simulation approach to the evaluation of coupled heat and mass transfer in buildings," in *Proc. 2nd IBPSA World Congress" Building Simulation*. vol. 91, 1991, pp. 219-226.
- [70] J. A. Clarke and N. J. Kelly, "Integrating power flow modelling with building simulation," *Energy and Buildings*, vol. 33, no. 4, pp. 333-340, 2001.
- [71] P. A. Strachan, G. Kokogiannakis, and I. A. Macdonald, "History and development of validation with the ESP-r simulation program," *Building and Environment*, vol. 43, no. 4, pp. 601-609, 2008.
- [72] Danish Building Research Institute. "BSim - Building Simulation — Danish Building Research Institute (SBI)." Internet: http://www.en.sbi.dk/publications/programs_models/bsim, 4th November 2009 [Accessed: 1st June 2011]
- [73] K. Grau and C. Rode, "Simulation of Whole-Building Hygrothermal Conditions," in *Proceedings of 8th International Conference on Air Distribution in Rooms* Copenhagen, Denmark, 2002.
- [74] C. Rode and K. Grau, "Integrated calculation of hygrothermal conditions of buildings," in *Proceedings of the 6th Symposium on Building Physics in the Nordic Countries*. vol. 1 Trondheim, Norway, 2002, pp. 23-30.
- [75] K. B. Wittchen, "Building Integrated Photovoltaics in a Thermal Building Simulation Tool," in *Eighth International IBPSA Conference* Eindhoven, Netherlands, 2003.
- [76] K. Grau, K. B. Wittchen, and C. G. Sørensen, "Visualisation of Building Models," in *Eighth International IBPSA Conference* Eindhoven, Netherlands, 2003.
- [77] K. Grau and K. B. Wittchen, "Building Design System and CAD Integration," in *Proceedings of IBPSA Conference: Building Simulation*. vol. 99 Kyoto, Japan, 1999.
- [78] Building Research Establishment, "iSBEM User Guide," Building Research Establishment 2010.
- [79] Building Research Establishment. "NCM: National Calculation Method." Internet: <http://www.ncm.bre.co.uk/>, 8th April 2011 [Accessed: 1st June 2011]

- [80] Building Research Establishment, "National Calculation Methodology (NCM) Modelling Guide for Non-Domestic Buildings in Scotland " Directorate for the Built Environment Building Standards Division, 2010
- [81] HM Government, "National Calculation Methodology (NCM) modelling guide (for buildings other than dwellings in England and Wales)," Department for Communities and Local Government, 2008
- [82] Sustainable Buildings Industry Council. "Energy 10 Software." Internet: <http://www.sbicouncil.org/energy10-soft>, 2011 [Accessed: 1st June 2011]
- [83] Solar Energy Laboratory, "TRNSYS 17: A Transient System Simulation Program," Univ. of Wisconsin-Madison, 2010.
- [84] University of Wisconsin-Madison. "TRNSYS - Official Website." Internet: <http://sel.me.wisc.edu/trnsys>, 11th January 2011 [Accessed: 1st June 2011]
- [85] Mathworks. "MATLAB - The language of technical computing." Internet: <http://www.mathworks.co.uk/products/matlab/>, 2012 [Accessed: 10th January 2012]
- [86] Mathworks. "Simulink - Simulation and Model-Based Design." Internet: <http://www.mathworks.co.uk/products/simulink/>, 2012 [Accessed: 10th January 2012]
- [87] O. Gumu. "C# versus C++ versus Java performance comparison." Internet: <http://reverseblade.blogspot.com/2009/02/c-versus-c-versus-java-performance.html>, 7th February 2009 [Accessed: 20th December 2011]
- [88] N. Kelly and J. Clarke, "The simulation of building electrical power flows," in *Proceedings of the 6th International IBPSA Conference, International Building Performance Simulation Association*, Kyoto, 1999, p. 15.
- [89] *Heating systems in buildings - Method for calculation of design heat load*, BS EN 12831, 2003.
- [90] J. Brogden, "Smart metering interoperability in the GB market," in *Smart Metering - Making It Happen, 2009 IET*, 2009, pp. 1-29.
- [91] J. Brogden, "ERA View on Smart Metering and Interoperability," in *Smart Metering - Gizmo or Revolutionary Technology, 2008 IET Seminar on*, 2008, pp. 1-19.
- [92] *IEEE Standard for Local and Metropolitan Area Networks - Part 15.4: Low-rate Wireless Personal Area Networks*, IEEE Standard 802.15.4, 2011.
- [93] *ZigBee Smart Energy Profile Specification*, 2011.
- [94] *ZigBee Home Automation Public Application Profile*, 2010.
- [95] Z-Wave Alliance. "Z-Wave." Internet: <http://www.z-wave.com>, 2011 [Accessed: 24th October 2011]
- [96] *IEEE Standard for Broadband over Power Line Networks: Medium Access Control and Physical Layer Specifications*, IEEE Standard 1901, 2010.
- [97] HomePlug Powerline Alliance. "HomePlug 1.0 Technology White Paper." Internet: https://www.homeplug.org/tech/whitepapers/HP_1.0_TechnicalWhitePaper_FINAL.pdf, 10th February 2005 [Accessed: 24th October 2011]
- [98] Sensus. "BT Chooses Sensus-Arquiva Partnership for Smart Grid Communication." Internet: <http://www.sensus.com/web/usca/news/display/bt-chooses-sensus-arquiva-partnership-for-smart-grid-communication-press-release-html-page>, 5th August 2010 [Accessed: 24th October 2011]

- [99] Arquiva. "Leading Meter Manufacturers to Integrate FlexNet, from Sensus." Internet: <http://www.arquiva.com/corporate/press/archive/2011/2011-8-12%20-%20Leading%20meter%20manufacturers%20to%20integrate%20FlexNet,%20from%20Sensus.pdf>, 12th August 2011 [Accessed: 24th October 2011]
- [100] Sensus. "FlexNet System Specifications." Internet: http://www.sensus.com/documents/10157/32460/amr_456.pdf, 22nd April 2010 [Accessed: 24th October 2011]
- [101] *IEEE Standard for Metropolitan Area Networks - Part 16: Air Interface for Broadband Wireless Access Systems*, IEEE Standard 802.16, 2009.
- [102] R. P. Lewis, P. Iqic, and Z. Zhongfu, "Assessment of communication methods for smart electricity metering in the U.K," in *Sustainable Alternative Energy (SAE), 2009 IEEE PES/IAS Conference on*, 2009, pp. 1-4.
- [103] S. Kim, E. Y. Kwon, M. Kim, J. H. Cheon, S. Ju, Y. Lim, and M. Choi, "A Secure Smart-Metering Protocol Over Power-Line Communication," *Power Delivery, IEEE Transactions on*, vol. 26, no. 4, pp. 2370-2379, 2011.
- [104] *IEEE Standard for Information Technology - Part 3: Carrier sense multiple access with Collision Detection Access Method and Physical Layer Specifications*, IEEE Standard 802.3, 2008.
- [105] *IEEE Standard for Information Technology Specific Requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; Amendent 5: Enhancements for Higher Throughput*, IEEE Standard 802.11n, 2009.
- [106] W. Stallings, "Data and computer communications," 8th ed. ed Upper Saddle River, N.J.: Pearson Prentice Hall ; London : Pearson Education Ltd., 2007, p. 43.
- [107] A. Clinick. "Script Happens.NET." Internet: <http://msdn.microsoft.com/en-us/library/ms974577.aspx>, 25th June 2001 [Accessed: 14th November 2011]
- [108] W3C. "Extensible Markup Language (XML) 1.0 (Fifth Edition)." Internet: <http://www.w3.org/TR/REC-xml/>, 26 November 2008 [Accessed: 23rd November 2011]
- [109] S. Chowdhury, G. A. Taylor, S. P. Chowdhury, A. K. Saha, and Y. H. Song, "Modelling, simulation and performance analysis of a PV array in an embedded environment," in *Universities Power Engineering Conference, 2007. UPEC 2007. 42nd International*, 2007, pp. 781-785.
- [110] J. A. Gow and C. D. Manning, "Development of a photovoltaic array model for use in power-electronics simulation studies," *Electric Power Applications, IEE Proceedings -*, vol. 146, no. 2, pp. 193-200, 1999.
- [111] K. Kurobe and H. Matsunami, "New two-diode model for detailed analysis of multicrystalline silicon solar cells," *Japanese journal of applied physics*, vol. 44, no. p. 8314, 2005.
- [112] K. Ishaque, Z. Salam, and Syafaruddin, "A comprehensive MATLAB Simulink PV system simulator with partial shading capability based on two-diode model," *Solar Energy*, vol. 85, no. 9, pp. 2217-2227, 2011.
- [113] M. AbdulHadi, A. M. Al-Ibrahim, and G. S. Virk, "Neuro-fuzzy-based solar cell model," *Energy Conversion, IEEE Transactions on*, vol. 19, no. 3, pp. 619-624, 2004.

- [114] L. Ho, L. Min-Jung, L. Se-Na, L. Hwa-Chun, N. Hae-Kon, and P. Sung-Jun, "Development of photovoltaic simulator based on DC-DC converter," in *Telecommunications Energy Conference, 2009. INTELEC 2009. 31st International*, 2009, pp. 1-5.
- [115] M. R. AlRashidi, M. F. AlHajri, K. M. El-Naggar, and A. K. Al-Othman, "A new estimation approach for determining the I/V characteristics of solar cells," *Solar Energy*, vol. 85, no. 7, pp. 1543-1550, 2011.
- [116] A. R. Hambley, *Electrical engineering : principles and applications*, 2nd ed. ed. Upper Saddle River, N.J. :: Prentice Hall, 2001.
- [117] N. B. Dale, C. Weems, and J. McCormick, "Programming and problem solving with Ada," Lexington, Mass: D. C. Heath, 1994, p. 722.
- [118] T. Croft, R. Davison, and M. Hargreaves, *Engineering mathematics : a foundation for electronic, electrical, communications, and systems engineers*, 3rd ed. ed. Harlow: Prentice Hall, 2001.
- [119] N. B. Dale, C. Weems, and J. McCormick, *Programming and problem solving with Ada*. Lexington, Mass: D. C. Heath, 1994.
- [120] R. Rynkiewicz, "Discharge and charge modeling of lead acid batteries," in *Applied Power Electronics Conference and Exposition, 1999. APEC '99. Fourteenth Annual*, 1999, pp. 707-710 vol.2.
- [121] M. Durr, A. Cruden, S. Gair, and J. R. McDonald, "Dynamic model of a lead acid battery for use in a domestic fuel cell system," *Journal of Power Sources*, vol. 161, no. 2, pp. 1400-1411, Oct 2006.
- [122] H. J. Schaetzle and D. P. Boden, "Lead-Acid Batteries for Remote Photovoltaic Applications," in *Telephone Energy Conference, 1978. INTELEC '78. International*, 1978, pp. 244-248.
- [123] O. Tremblay, L. A. Dessaint, and A. I. Dekkiche, "A Generic Battery Model for the Dynamic Simulation of Hybrid Electric Vehicles," in *Vehicle Power and Propulsion Conference, 2007. VPPC 2007. IEEE*, 2007, pp. 284-289.
- [124] Mathworks. "Simple Battery Model." Internet: <http://www.mathworks.co.uk/help/toolbox/phymod/elec/ref/genericbattery.html>, [Accessed: 29th September 2009]
- [125] *Electricity metering equipment (a.c.). General requirements, tests and test conditions. Metering equipment (class indexes A, B and C)*, BS EN ISO 50470-1, 2006.
- [126] *Building materials and products. Procedures for determining declared and design thermal values*, BS EN ISO 10456, 2007.
- [127] F. P. Incropera, *Fundamentals of heat and mass transfer*, 6th ed. ed. Hoboken, N.J.: Wiley, 2007.
- [128] *Thermal performance of windows, doors and shutters - Calculation of thermal transmittance - Part 1: General*, BS EN ISO 10077-1, 2006.
- [129] R. Perez, P. Ineichen, R. Seals, J. Michalsky, and R. Stewart, "Modeling daylight availability and irradiance components from direct and global irradiance," *Solar Energy*, vol. 44, no. 5, pp. 271-289, 1990.
- [130] Heatline Radiators. "Radiators Specification." Internet: http://www.heatlineradiators.co.uk/menu_radiators_spec.htm, [Accessed: 30th September 2010]

- [131] US Department of Energy. "EnergyPlus Weather Data." Internet: http://apps1.eere.energy.gov/buildings/energyplus/weatherdata_about.cfm, [Accessed: 22nd January 2012]
- [132] G. M. S. Azevedo, M. C. Cavalcanti, K. C. Oliveira, F. A. S. Neves, and Z. D. Lins, "Evaluation of maximum power point tracking methods for grid connected photovoltaic systems," in *Power Electronics Specialists Conference, 2008. PESC 2008. IEEE*, 2008, pp. 1456-1462.
- [133] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, MA: MIT Press, 2004.
- [134] Energy Saving Trust, "Measurement of Domestic Hot Water Consumption in Dwellings," 2008.
- [135] I. Richardson, M. Thomson, D. Infield, and C. Clifford, "Domestic electricity use: A high-resolution energy demand model," *Energy and Buildings*, vol. 42, no. 10, pp. 1878-1887, 2010.
- [136] I. Richardson, M. Thomson, D. Infield, and A. Delahunty, "Domestic lighting: A high-resolution energy demand model," *Energy and Buildings*, vol. 41, no. 7, pp. 781-789, 2009.
- [137] "The Shunting Yard Algorithm." Internet: http://en.literateprograms.org/Shunting_yard_algorithm_%28C%29, 1st February 2008 [Accessed: 10th January 2012]
- [138] ic#code. "SharpDevelop." Internet: <http://www.icsharpcode.net/OpenSource/SD/>, 2012 [Accessed: 10th January 2012]