THE UNIVERSITY OF QUEENSLAND

GridIQ - A Test Bed for Smart Grid Agents

By
Colin John Bankier

SUBMITTED FOR THE DEGREE OF
BACHELOR OF ENGINEERING (HONOURS)
IN THE DIVISION OF
SOFTWARE ENGINEERING

Department of Information Technology and Electrical Engineering,
The University of Queensland.

September 2010

1st September 2010

Head of School

School of Information Technology and Electrical Engineering

University of Queensland

St Lucia, Qld 4072

Dear Professor Paul Strooper,

In accordance with the requirements of the degree of Bachelor of Engineering (Honours) in the division of Software Engineering, I present the following thesis entitled

"GridIQ - A Test Bed for Smart Grid Agents".

This work was performed under the supervision of Mr Jorn Guy Suess.

I declare that the work submitted in this thesis is my own, except as acknowledged in the text and footnotes, and has not been previously submitted for a degree at the University of Queensland or any other institution.

Yours sincerely,

COLIN JOHN BANKIER.

Acknowledgments

I would like to thank my supervisor, Mr Jorn Guy Suess, for his enthusiasm and guidance throughout this project. His input and advice have not only been invaluable to the success of this project, but have also provided me with a deeper level of understanding as an engineer.

I would to thank my wife Angela for her immense support and for the sacrifices that she has made that have enabled me to complete this degree.

Finally, I would like to thank my father Neale for inspiring me to achieve anything that I could dream of, and my mother Jenny for her love and support to ensure that I could do so.

Abstract

The power industry is facing new challenges due to escalating demand and environmental issues such as climate change. The application of information technology to power systems to create intelligent electricity networks, labelled smart grids, has been proposed as a means to address these issues. Agent-based systems have been identified as a suitable paradigm for developing software for smart grids. Tools used by researchers and practitioners developing smart grid systems include agent platforms and power system simulations tools, yet no open source tool currently exists which enables these tools to inter-operate. This research describes the development of GridIQ, an open source software system which enables agents executing in an agent platform to interact with devices in a power simulation tool during simulation, providing a test bed for smart grid agents. The JADE agent platform and the PSAT power simulation tool were selected as the external tools supported by GridIQ. The architecture and behaviour of the system are presented, as well as the management and development processes adopted. An example scenario is also described which demonstrates the GridIQ system successfully enabling JADE agents to monitor and control devices in a PSAT network during simulation.

Contents

List of Figures

List of Tables

Nomenclature

CSIRO Commonwealth Scientific and Industrial Research Organisation

FIPA Foundation for Intelligent Physical Agents

IEEE Institute of Electrical and Electronics Engineers

JADE Java Agent Development Framework

MAS Muli-agent system

MATLAB A commercial software system for numerical computing.

PSAT Power Systems Analysis Toolkit

SCADA Supervisory Control and Data Acquisition

UML Unified Modeling Language

Chapter 1

Introduction

The electricity industry in many countries is facing new pressures due to escalating demand and environmental concerns such as climate change. Electricity networks which are more efficient, can accommodate greater amounts of distributed energy resources, and provide demand-side management have been proposed as a possible solution to these issues. The application of information technology to power systems in order to make electricity grids intelligent is gaining momentum as a means to achieve these goals. These electricity networks have been termed smart grids.

Software systems consisting of autonomous collaborating entities, referred to as agents, may be suitable for providing intelligence to smart grids. A substantial body of research exists exploring the application of agents to various aspects of power system control. Existing agent middleware platforms are commonly used by researchers in this area, yet no free or open source tools exist to interface these systems with power system simulation tools. The hypothesis underpinning the current research is that an open source tool can be developed which allows existing agent middleware systems to integrate with power system simulation tools. The aim of this research is to develop such a tool, titled GridIQ, which would provide a test bed for smart grid agent systems.

The following outlines the contents of the remainder of this document. An overview of challenges facing the power industry and an introduction to the concept of smart grids are presented in the remainder of this chapter. Chapter 2 introduces relevant power systems concepts as well as the concept of agent based software systems and reviews existing literature on applying such systems to power system control. Tools utilised by researchers are discussed and the lack of tools that enable agent systems to inter-operate with power simulation tools is identified, providing motivation for current project. The GridIQ system, including the goals of the project, project management methodologies

including open source and agile software development, the system architecture and behaviour as well as integration with external tools are presented in Chapter 3. Chapter 4 presents an example scenario demonstrating the GridIQ system successfully enabling an agent platform to interact with a power simulation tool. Finally, a discussion on the outcome of the project and future research areas are presented in Chapter 5.

1.1 Power: A Changing Industry

This section discusses challenges facing the power industry and introduces the concept of a smart grid. Key technologies that are proposed to address these challenges are discussed.

World energy consumption has increased dramatically in the last century to the point where resource availability and ecological pressures on human society are being perceived more keenly than before. An abundance of fossil fuel resources has enabled civilisations to be based around electricity grids which passively distribute power from centralised generation plants. Environmental concerns such as climate change and resource constraints are driving a greater need for energy efficiency and a diversification of energy sources. Current grid systems are proving inadequate to address these issues and have been termed both inefficient and unreliable [1]. One key issue is the fact that there is a growing trend to introduce renewable energy sources as well as other types of electricity generation distributed throughout the grid [2], but existing systems were designed only to support centralised remote generation sites [3]. The need for increased contribution from renewable energy sources is largely driven by two main factors: the need to reduce carbon dioxide emissions produced by electricity generation; and energy security for regions reliant on imported fossil fuels [4]. These distributed resources present a substantial change to way the electricity infrastructure is managed and controlled [2]. Making electricity grids more intelligent through the use of information technology has been proposed as a means to cope with such issues.

1.1.1 The Smart Grid

The term smart grid refers to the application of information technology to power systems, and is expected to address many of the shortcomings of the current grid [1]. A smart grid integrates advanced sensing technologies, control methods and integrated communications into the current electricity grid in order for grids to be able to accommodate all generation and storage options and to optimise assets

and operations efficiently [5]. Electricity grids are expected to evolve into networks in which the nodes become active and intelligent and a vast number of system parts communicate with and influence each other [2]. Additionally, they are proposed to become self-healing and resilient to system anomalies in order to increase both efficiency and reliability [1]. Several key technologies in such a system are described below.

Smart Meters

Smart meters, otherwise referred to as Advanced Metering Infrastructure (AMI) or two-way meters, are electric meters which utilise two-way communications between the meter and the utility company and enable two-way power flows to consumers. This technology enables users to monitor their consumption patterns in real-time and enables features which include time stamping of meter data, outage reporting, communication into the customer premise and on-request reads [6]. It also enables the consumer to supply power back to the grid from solar panels or other local renewable sources and enables load management by responding to fluctuations in demand.

Sensors

The use of sensors monitored by intelligent systems throughout the network is intended to allow the system to automatically adapt and respond to changing conditions [4]. It is expected that distribution devices will become intelligent remote agents on communication networks providing data collected through sensors back to operations control centres [3]. With current technology it is difficult to monitor power flows throughout the distribution grid as measurements are typically only available at the distribution substations. Greater levels of information are critical to understanding the impact and benefit of connecting renewable energy sources to the grid [7]. Sensors located throughout the network along with smart metering will be able to collect this information.

Demand Response

Demand response is a key feature of a smart grid that allows power utilities to manage the demand on the electricity grid [8] and empowers consumers to manage their energy use [1]. It allows consumers to adjust their consumption patterns in response to changes in the price of electricity over time allowing load to be shed from the network during times of peak demand [6]. This is achieved through smart appliances which operate in conjunction with a smart meter to adjust their operation based on

a signal from the grid. Such appliances may also allow the utility to remotely control their operation to balance network load.

Energy Storage

The variable nature of renewable energy sources can cause problems with the conventional system balancing methodologies [4]. Storing excess energy provides a solution to managing variable power sources such as wind or other renewable sources. Stored energy can then be returned to the grid when supply drops off. Technologies include batteries, compressed air storage, high-energy flywheels and electrochemical capacitors [9]. Plug-in hybrid electric vehicles are proposed as another potential storage system which would be enabled by a smart grid [10]. Using current infrastructure such vehicles could cause major disruptions to power supplies [11].

This chapter has provided an introduction to the current research topic and presented an overview of the remainder of this document. It has introduced the concept of a smart grid as a means of addressing challenges faced by the power industry. Key technologies that are designed to enable greater renewable energy resource penetration in power grids and enable grids to become smarter and more efficient such as demand management, sensor networks and storage have been discussed.

Chapter 2

Applying Agents to Power Systems

The previous chapter introduced the concept of a smart grid and discussed intelligent devices which may become part of future grids. A common technique to provide this intelligence is to utilise small autonomous computer programs called agents. This chapter presents background theory and existing research related to applying agents to power systems. First, an overview of key concepts in power systems is provided in Section 2.1. Section 2.2 introduces the concept of agent systems, and explains why the agent paradigm is relevant to smart grids. A review of existing research literature on applying agents to power system control is presented, and tools used for research in this area are discussed. The lack of available tools which enable agent systems to integrate with power simulation tools is identified, providing motivation for the current project.

2.1 Power System Analysis and Control

This section introduces basic power systems concepts and terminology relevant to the current research. The major components of a power system are described along with the concepts of power system analysis and control.

A power system consists of four major components: generation, transmission, distribution and loads [12]. The generation component includes power generated from any source such as the burning of coal or gas, or from other sources such as nuclear, wind, solar and others. Transmission networks are responsible for transporting electricity long distances, typically at very high voltages to minimise losses, from generation sites to the distribution system. Distribution networks are responsible for

supplying the load within a well-defined geographical area to industrial, commercial and residential consumers.

Power flow analyses are a key aspect of power system analysis and design, necessary for planning, operation, economic scheduling and exchange of power between utilities [12]. A power flow analysis is a mathematical tool used to determine the voltage magnitude and angle as well as real power and reactive power flows for all buses in a network given some initial conditions [13]. A node on the network where voltage or other quantities are evaluated is commonly referred to as a *bus*. Saadat notes that buses are generally classified into three types and provides the following definitions [12].

Slack Bus One bus in the network is taken as a reference bus where the voltage magnitude and phase angle are specified. This bus makes up the difference between loads and generated power caused by losses in the network. This is also known as a *swing bus*.

Load Bus Buses where active and reactive powers are specified but the voltage magnitude and angle are unknown. These are referred to as *P-Q Bus*es.

Regulated Bus Generator buses where the real power and voltage magnitude are specified. These are referred to as *P-V Buses*.

Quantities such as power, voltage, current and impedance in a power system are commonly expressed using the *per-unit system* (p.u.). This system was devised to avoid "cumbersome transformations" necessary when solving systems with several different voltage levels, instead expressing quantities as fractions or multiples of some base quantity [12].

The key aspect of power systems to which intelligent software systems are being applied in smart grids is power system control. This area is concerned with maintaining a power system within a normal operating state. Saadat states that

> "The objective of the control strategy is to generate and deliver power in an interconnected system as economically and reliably as possible while maintaining the voltage and frequency within permissible limits" [12].

As discussed is Section 1.1, increased penetration of renewable and distributed energy resources presents new challenges in optimising this control strategy, and is a key driver of smarter grids.

This section has introduced some basic power system concepts, including the major components of a power system, power systems analysis, power system control and different bus types. The following section introduces the concept of agent systems.

2.2 Agent Systems

This section introduces the concept of an agent system. Key properties are defined and why they are suited to the control of complex systems is explained.

The concept of an agent system in software engineering refers to a paradigm for designing software systems as a set of components called agents. Wooldridge defines an agent as

> "an encapsulated computer system that is situated in some environment and can act flexibly and autonomously in that environment to meet its design objectives" [14].

Agents exhibit four key properties: autonomy (they operate without human intervention), sociality (they interact with other agents), reactivity (they perceive and react to their environment) and pro-activity (they exhibit goal-oriented behaviour by taking initiatives) [15]. Jennings and Bussmann elaborate upon this definition by adding that agents are problem-solving entities with well-defined boundaries and interfaces, have partial control and observability over their environment, have control over both their internal state and their own behaviour and are capable of exhibiting flexible problem-solving behaviour in pursuit of their design objectives [16].

A software system composed of agents is referred to as a multi-agent system (MAS) or agent-based system. Such systems have been identified as being suited to control of complex systems which consist of a large number of interacting parts and operate in dynamic environments [16]. The complexity of these systems necessitate that the supervisory function be distributed throughout the system with some controllers supervising processes directly while others supervise other controllers [17]. This enables complex behaviours to be decomposed into more manageable sub-functions, detailed tasks to be aggregated into high-level tasks, multiple time-scales and levels of detail to be considered, as well as enabling spatial distribution throughout the system. Agent-based systems provide a decentralised solution based on local decision making that gives the system a high degree of flexibility and robustness. The use of autonomous components for decision making, however, can make it more difficult to predict overall system behaviour, and is key topic of current agent research [16].

This section has introduced the concept of an agent system and suggested that agents are suited to the control of complex systems. The following section discusses the application of agent systems specifically to smart grids.

2.3 Agents in the Smart Grid

This section presents a review of previous research applying agent-based software systems to power system control and smart grids. A common goal of existing research in this area is identified, and research applying agents in both simulation and real-world environments are discussed.

Previous research has suggested that agent-based systems may be more suited to smart grid control than current Supervisory Control and Data Acquisition (SCADA) systems. Vale et al [18] discuss the limitations of current SCADA systems and their inadequacy at providing the control required in smart grids. Pipattanasomporn et al suggest that there is a growing trend towards the utilisation of automated multi-agent systems and discusses research investigating the applications of agents to various aspects of power systems such as system disturbance diagnosis, system restoration, system visualisation, micro-grid operation and distributed energy resource management [19].

A common goal of existing research is to demonstrate that agent systems can successfully monitor and control a power network under specific conditions. Many present designs for agent systems to achieve specific goals. Pipattanasomporn et al present a design and implementation of an agent system designed to secure critical loads in a grid during outages [19]. Work by Jiang presents an agent-based control framework to coordinate distributed energy resources and to manage power and voltage profiles [20]. Morejon et al present an agent system design for control of shipboard power systems [21].

Australia's Commonwealth Scientific and Industrial Research Organisation (CSIRO) has been developing multi-agent technology for deployment within the Australian National Electricity Market [22, 23]. One aspect of this research presented three different agent designs for electrical load management and explored their performance under various circumstances through both comparative experiments and theoretical analysis [22]. Another aspect of this research presented an algorithm for reinforcement learning of agents in order to optimise both system stability and energy cost [23].

Research also exists describing the deployment of agent systems in real-world power grids rather than in simulation environments. Cohen describes a commercial multi-agent system currently being deployed within Con Edison of New York's distribution system [24] and Dimeas and Hatziargyriou describe the installation of an agent system for micro-grid operation in Europe [25].

This section has reviewed existing research applying agent systems to smart grid control. The following section discussed tools commonly used by researchers in this area.

2.4 Tools for Smart Grid Agent Research and Development

This section discusses tools used for research and development of smart grid agent systems. First, the role of simulation in exploring complex systems is discussed. Next, agent platforms which provide development tools and run-time environments for agent systems as well as simulation tools for the analysis of virtual power systems are discussed. Finally, the integration of these types of tools is discussed and the lack of available tools providing this functionality is identified.

2.4.1 Simulation: Learning by Experience

Simulation is an important tool in the development of smart grid technologies. The large scale deployment of such technologies to date has been limited due to an inability to accurately model their effects or to quantify their potential benefits [26]. In order to provide this necessary information, smart grid research commonly involves simulating agent systems in a virtual environment as in research previously discussed by Pipattanasomporn et al, Jiang, the CSIRO and Morejon et al. Several key reasons exist for using models and simulations for answering questions about complex systems [27]. Experimentation on the real system may be too expensive, too dangerous, or the system may not actually exist yet. The time scale of the system may not be compatible with the experimenter (e.g. millions of years). A simulation allows all variables to be studied and controlled where in a real system many variables may be inaccessible. Furthermore, a simulation also enables disturbances and second order effects to be suppressed, allowing a better understanding of the primary effects to be gained. All of these factors may be applied specifically to smart grid research. It is, however,

important to consider that many aspects of the real system are not represented by a model and that a model can only represent a real system under certain conditions.

2.4.2 Agent Platforms

An agent platform is a set of tools that provides both a run-time environment to execute agent programs as well as a development framework which aids developers to create agents. An agent platform is typically a distributed computing system, with agents executing on multiple physical computers connected by a network. Agent platforms can typically be described as middleware because they hide the complexities of underlying heterogeneous operating systems and networks and provide services and protocols for high-level communication between agent programs. In order to promote the inter-operation of heterogeneous agents and the services that they can represent, a standards organisation, The Foundation for Intelligent Physical Agents (FIPA) exists as part of the IEEE (Institute of Electrical and Electronics Engineers) Computer Society.

Pipattanasomporn et al [19] present a list of open source agent platforms which they considered applying to smart grid research. These include Aglets software development kit [28], Voyager, Zeus [29], JADE (Java Agent Development) framework [30], Tracy, SPRINGS [31] and SkeletonAgent. Such frameworks do not provide any specific features for developing agents for power systems, however they have been utilised by previous researchers for this purpose. The CSIRO [22, 23] and Dimeas and Hatziargyriou [25] utilised JADE for power system control, while Pipattanasomporn et al [19] selected the Zeus framework. In order to utilise these agent platforms for smart grid simulation however, additional software is required as discussed is Section 2.4.4.

2.4.3 Power System Simulation

Simulation is commonly used in the design of power systems and many simulation tools exist which allow complex analysis of virtual power systems. Commercial tools include Siemens PSSE [32] and DIgSILENT PowerFactory [33]. Milano and Vanfretti present a review of open source software for this purpose [34]. Tools such as these have been utilised by previous researchers to simulate agents for power systems. Jiang [20] describes the use of Virtual Test Bed, a simulation package developed by the University of South Carolina [35], to simulate an agent-based system for power systems but does not discuss integration with an agent development framework. The United States Department of

Energy has also developed a simulation environment specifically for smart grid systems, GridLAB-D [26]. The applicability of these tools to the current research is discussed in more detail in Section 3.3.

2.4.4 Integrating Agent Platforms and Power Simulation Tools

Previous researchers such as Pipattanasomporn et al and Morejon et al have integrated agent platforms and power systems in a simulation environment. The agent platforms, however, do not provide any features enabling integration with a power simulation tool, requiring researches to develop their own solutions to integrate these tools. Pipattanasomporn et al describe the development of a test bed in MATLAB, a commercial software system for numerical computing [36], for integration with the Zeus agent system, while Morejon et al describe developing a system using the CORBA distributed computing platform to integrate the JADE agent platform and Virtual Test Bed to simulate agent control of shipboard power systems [21].

The software developed by previous researchers for this purpose has not been released, so there is no such test bed for agent systems available for future researchers and students to utilise. Some power simulation tools do support some agent-based simulation features, such as GridLAB-D. These features differ, however, from the ability to integrate agents developed for an agent platform in that they do not allow the simulation and testing of agents that can be deployed in a real-world distributed environment. There does not currently exist any publicly available or open source tool which enables the integration of an agent development framework and a power system simulation tool.

This section has discussed tools utilised for the research and development of smart grid agent systems. The role of simulation in the exploration of complex systems was discussed and an overview of agent platforms and power system simulation tools was provided. Finally the integration of these two types of tools was discussed as means for testing smart grid agent systems in a simulation environment.

This chapter has provided background theory and reviewed existing research relating to applying software agents to power systems. First, power system analysis and control concepts were discussed, then the concept of agent-based software systems were introduced. Existing research exploring the application of agents to smart grids was reviewed, and tools commonly used by researchers in this area were identified. The lack of tools which enable agent platforms to inter-operate with power

simulation tools was identified. The following chapter presents the GridIQ system, a tool which aims to provide this functionality.

Chapter 3

The GridIQ System

This chapter details the development of GridIQ, a test bed for smart grid agents. The motivation and goals of the project are presented in Section 3.1 along with an overview of the systems functionality and development tools and technology. Section 3.2 describes the management processes employed in the development of GridIQ, including an introduction to open source software and a description of the resources available for users and developers. This section also discusses the development process used which adopted key agile software development techniques. Section 3.3 discusses the selection of external components supported by GridIQ and Section 3.4 describes the usage and running of the system. Finally, Section 3.5 presents the architecture and implementation of GridIQ including the system structure and system behaviour. Unified Modelling Language (UML) diagrams are provided throughout this chapter to illustrate aspects of the systems behaviour and design.

3.1 Overview

This section introduces GridIQ by providing the motivation and goals of the project. The relationship between the GridIQ system and other components is presented and tools and technology used in development are discussed.

3.1.1 Motivation and Goals

Chapters 1 and 2 have detailed the changes faced by the power industry, and discussed research indicating that agent-based software systems may be a key mechanism in creating the smart power

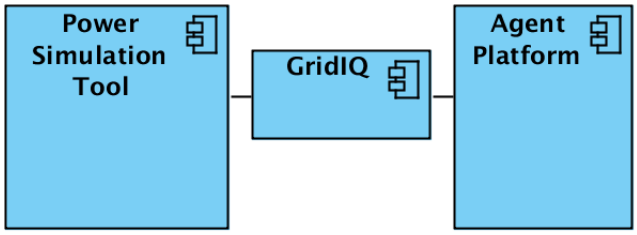

Figure 3.1: GridIQ in Relation to External Components

grids of the future. Agent platforms used by researchers in this area were discussed in Section 2.4 along with the need to use these platforms in a simulation environment in order to advance smart grid agent systems. There are, however, no open source tools available which enable an agent platform to integrate with a power simulation tool. Previous researchers have created such tools specific to their purposes, but these tools are not available for others to use. This provides the underlying motivation for developing GridIQ.

The goal of GridIQ is to provide a bridge between an agent platform and a power simulation tool. It will act as a test bed for agent systems by enabling agents executing within the agent platform to interact with devices in a virtual power network during simulation. GridIQ aims to leverage existing systems and technology where possible by utilising an existing agent development framework as well as an existing power system simulation tool. Figure 3.1 illustrates the relationship between the GridIQ system and these external components. Collaboration will be encouraged through open source development and clear documentation. The aim is to deliver a tool which can be used by researchers to develop the next generation of smart grid agents.

The open source goals of the project include avoiding any dependencies on proprietary external tools or applications, in order to enable researchers and students to use and collaborate on development of the system without restrictive licencing fees or limitations. Additionally, the aim is to for GridIQ to be cross-platform and not tied to a particular operating system.

3.1.2 Development Tools and Technology

GridIQ has been developed in the Java programming language. This technology was selected because it allows cross platform compatibility via the Java virtual machine run-time environment. Java is also a widely used programming language used in both commercial and academic environments,

maximising the potential that users and collaborators of the GridIQ project will be familiar with the development language. The platform also supports a strong open source ecosystem, meaning many open source tools, including agent platforms, have been developed in Java. Selecting Java for the development of GridIQ provides greater possibilities for integration with these existing tools.

The Eclipse integrated development environment (IDE) [37] has been utilised for the development of GridIQ. Eclipse was selected for the primary reason that it is the most commonly used IDE for Java development, maximising familiarity to users and contributors.

This section has provided the motivation for developing GridIQ and described the system in relation to external components. The Java platform and Eclipse IDE have been discussed as tools and technology used in development. The following section describes management processes adopted for the project.

3.2 Project Management and Development Methodologies

This chapter describes the methodologies used in the development of the GridIQ software. The concept of open source software is introduced along with a rationale for developing GridIQ under this paradigm. Resources maintained during the project in order to facilitate open source development are then described. Finally, the concept of agile software development is introduced and agile techniques adopted for the project are discussed.

3.2.1 Open Source: Community Development

The term open source software refers to software which provides liberties to users to use, study, modify, and redistribute the software by making the source code freely available. This paradigm is in contrast to proprietary software which usually places restrictions on a user's right to copy, modify, and redistribute it. The terms free software and open source software have different definitions, provided by the Open Source Initiative [38] and the Free Software Foundation [39] respectively, but are closely related and are commonly combined under the inclusive term free and open source software (FOSS). The term free software does not refer to the price of the software, but to the freedoms granted to users of the software . Open source software is often developed under a peer to peer model in which anybody can contribute to the development of the software.

The open source phenomenon raises questions about what motivates developers to contribute to software without compensation for their work [40]. Additionally, critics argue that open source projects cannot compete with commercial products on stability, reliability, and innovation [41]. Open source software is, however, becoming a key driver of the software industry. Internet giants like Google, Yahoo and Facebook have built their infrastructures using primarily open source technologies [42]. Norris describes the use of open source software in mission-critical operations at NASA, assessing open source components used to be of higher quality than commercial products used or components developed in-house [43]. It has been argued that the growing strength of open source software in the industry drives greater innovation and has changed the way software is developed by influencing current software engineering processes [44].

A key aspect of open source software is community development. It is driven by engineers using the software who actively engage in its evolution by critiquing and reviewing source code as well as having strong input on features and implementation [45]. With an active community, an open source project can quickly evolve into a mature product without the financial investment required to develop a commercial product. It is the potential to have a high number of participants using and testing the software and submitting both bug reports and fixes that enables a high level of quality to be achieved. The feature set can also evolve to more directly suit the needs of those using the software, as users participate actively in development.

The choice to manage the GridIQ project as open source has been made for two reasons. The first is to take advantage of the community development model to gain input from other researchers and potential users with the intention that GridIQ might evolve into a mature project used in future smart grid research and development. The prevalence of open source software in the industry motivated the second reason: to gain learning experience managing an open source project.

3.2.2 Open Source Project Resources

In order to manage the GridIQ project as open source, a set of resources has been provided to enable collaboration with a user community. These resources, discussed below, were provided via Sourceforge.net, an online portal for hosting open source projects.

Project Web Site

The project web site provides a general overview of the project and its purpose. Its purpose is to act as a first source of information for users seeking information on GridIQ, providing links to the other resources. The website is available at [46].

GridIQ Sourceforge Web Site

The GridIQ Sourceforge site provides resources for enabling open source collaboration on the project and as available at [47]. It provides downloads of GridIQ related files, and provides mechanisms for users to obtain support using GridIQ as well to engage in development. Key features provided are listed below.

User Forums The user forums provide the primary means of communication and discussion for the user and developer community. It is also the primary means for users to request help using GridIQ.

Bug Tracking/Feature Requests Sourceforge.net provides a tracker system for bug tracking and feature requests. It allows bugs or requests to be categorised and prioritised, allocated to specific developers for resolution, and assigned a status. This is an essential tool for managing tasks and responsibilities for contributors.

Source Code Version Control System A version control system tracks changes to project source files and is a critical tool for developing software in teams. Changes made by individuals are merged back into a central source code repository, and a full history of changes is maintained. The GridIQ source code has been managed using Subversion, a popular version control system, on Sourceforge.net. Anonymous users may check out a copy of the source code from the repository, but only authorised developers are permitted to commit changes back to the code base.

The long-term vision for these resources is that smart grid researchers may use them to collaborate and continue to develop GridIQ to suit to their requirements.

3.2.3 Agile Development

The term agile software development refers to a group of related software development methodologies with an emphasis on iterative development, accommodating changing requirements and team communication. To compare agile with other methodologies Boehm and Turner suggest that all development processes fall within a continuum from 'adaptive' to 'predictive' with agile processes falling on the 'adaptive' side [48]. Predictive methodologies focus on planning in detail, identifying all features, tasks and delivery dates at the outset of the development process. Adaptive processes on the other hand focus on adapting to change by dividing tasks into small increments with short time frames. Planning occurs throughout development, enabling the project plan to change when necessary.

Agile development methodologies were developed in recognition of the fact that it is often impossible to posses perfect knowledge about a project at its outset, and that the inability to adapt to change can be a major cause of software project failures. It was for this reason that agile techniques were deemed applicable to the GridIQ project. The GridIQ project involved unfamiliar technologies and external components, and increased knowledge of the system requirements was continually gained throughout the research and development process.

It has been suggested that the rising acceptance of agile methods is strongly correlated with the rise in open source software [44]. Open source development requires global collaboration between developers, distributed change management and iterative development. Several key software engineering techniques associated with agile development such as continuous integration, test-driven development, automated unit testing, and code refactoring are common in open source development. In order to facilitate the open source development of GridIQ, several of these techniques were adopted including test-driven development, automated unit-testing and code refactoring. These techniques are described below.

Test-Driven Development

Test-driven development is an approach in which tests are written which specify the required behaviour of a particular feature of the system before the system code is written. This approach is the reverse of traditional software engineering processes where testing is usually the last step in the development cycle. Writing tests before the system code serves two purposes: the tests provide a verifiable specification of the software's functionality and allow the system to be extended and modified with

confidence that existing functionality has not been compromised [49]. Test-driven development is typically utilised in conjunction with automated unit testing, described below.

Automated Unit Testing

A unit test refers to a test which exercises a particular unit, typically a single class, of the system code. Automated unit testing refers to the use of tools to automatically run all units tests for a system. The benefit of automating the process is that tests can be run regularly to identify whether changes to the system have compromised existing functionality. JUnit is a common unit testing framework for Java which was used to create and run automated unit tests for GridIQ. The set of tests is maintained in the source code repository with the system code, but test code is excluded from the build for releases.

Code Refactoring

Code Refactoring is the process of changing the design of existing code. The purpose is to create a series of small incremental changes which do not modify the external behaviour of the system, but that result in better code [50]. Refactoring is a key technique in developing systems which evolve. The philosophy is that it is impossible to design a system up front to support all possible future requirements. Instead, the most simple design which achieves the current known requirements should be implemented, when new requirements become known existing code can be refactored to support them. This approach was employed through the development of GridIQ. At the outset of the project it was not possible to identify all the details relating to the integration of external components. As more knowledge was gained about the components, code could be refactored to accommodate new methods of interactions.

This section has discussed open source and agile software development methodologies adopted for the GridIQ project. An introduction to the open source software movement and the benefits of community development has been provided, as well as a description of resources maintained to facilitate open source development of GridIQ. Additionally, the concept of agile software development has been introduced and the techniques adopted for the GridIQ project explained.

3.3 External Components

GridIQ acts as a bridge between two types of external tools: an agent platform and a power simulation tool. Within the scope of the current project, only one tool of each type is supported. This section describes the process of selecting the supported tools and provides a description of those selected. The selection process aimed to identify the tools which best matched the following criteria: are open source; are mature projects under active development with an active community; and are currently used by researchers in power systems and smart grids. The criteria for external components to be open source was included as the open source nature of GridIQ would be undermined if it relied on proprietary components.

3.3.1 Agent Platform

The agent platform supported by GridIQ should provide basic services for agent development and execution such as communication and life-cycle management as well as support the development of systems which comply with FIPA standards for agent systems. Agent platforms used or discussed in previous research were investigated. Several platforms mentioned in research appeared to have been discontinued. Those that were still active included JADE [30], SPRINGS [31] and Aglets [28].

The maturity of these projects was the primary factor used in selection. Norris [43] suggests using the amount of time the development team has invested and the number of releases the team has made as metrics to assess the maturity of an open source project. JADE (Java Agent Development Framework) was found to be the most mature project, with an active user and developer community and a comprehensive feature set, and was hence selected as the agent platform supported by GridIQ.

JADE is a middleware for the development and run-time execution of peer-to-peer applications which are based on the agents paradigm [30]. An instance of the JADE run-time is called a container, while the set of all containers is referred to as the platform and provides a homogeneous layer that hides the complexity and heterogeneity of underlying hardware, operating systems and networks. JADE provides mechanisms for agents to dynamically discover and communicate with other agents and supports asynchronous messaging between agents which complies with Agent Communication Language (ACL) specifications defined by FIPA. An ACL defines the protocol for communication between agents. The JADE community [51] provides an active group of users and forums, good documentation and tutorials for using JADE, as well as regular releases.

3.3.2 Power Simulation

A range of power simulation tools were investigated in order to determine the most appropriate tool to be supported by GridIQ. Tools discussed by Milano and Vanfretti in their review of FOSS for power systems [34] were considered, as well as tools mentioned by other smart grid researchers. Based on the maturity and stated features of each project, five tools were selected for further evaluation. A description of each of these is provided below.

Virtual Test Bed

Virtual Test Bed (VTB) is a simulation package developed by the University of South Carolina that supports design, analysis, and virtual prototyping of electric systems [35]. The fact that the integration of this tool with the JADE agent platform is discussed by Morejon et al [21] suggested that it could be highly suitable for the current project. After further analysis however, this tool was rejected for the following reasons. VTB is developed for the Microsoft .NET platform, while JADE is developed for the Java platform. In order for these systems to inter-operate a distributed computing platform such as CORBA [52]is required, adding significant complexity to the GridIQ project. VTB is also only compatible with the Microsoft Windows operating system, and the goals of GridIQ include making the system compatible with other major operating systems.

InterPSS

InterPSS [53] is a Java based simulation tool which provides both a graphical user interface and Java libraries which can be used directly from other Java applications. This allows the tool to natively inter-operate with the JADE agent platform, a feature which initially suggested it could be very suitable to the current project. After further evaluation, this tool was rejected for the following reasons. Source code for the core power simulation libraries was not available and the documentation for the API was poor. This made the tool difficult to use and many other users on the forum reported difficulties relating to the quality of the tool.

Pylon

Pylon is an implementation of the MATLAB simulation tool Matpower in the Python programming language [54]. This tool provides a command line interface which would allow it to be executed by

GridIQ. Although this tool does support the required functionality, it is maintained by a single developer and an active user base or community is not evident. The original MATLAB implementation, Matpower, appeared to be a more mature project with a wider user base.

Matpower

Matpower [55] is a package of MATLAB M-files for solving power flow and optimal power flow problems. It is compatible with Gnu Octave, an open source alternative to MATLAB. Octave is a command line only tool, which means integration with GridIQ could be achieved by executing an external process. This tool also appeared to support the required functionality, but another Octave/-MATLAB based tool was selected as described below.

PSAT (Power Systems Analysis Toolkit)

The Power Systems Analysis Toolkit (PSAT) [56] was selected as the power simulation tool supported by GridIQ. It is a MATLAB based tool which, like Matpower, is also compatible with GNU Octave so is not reliant on the the proprietary MATLAB. MATLAB is, however, commonly used in power systems research and education, so a high proportion of the GridIQ intended audience would be familiar with this tool. Although only the Octave version of PSAT will initially be supported by GridIQ, support for the MATLAB version is discussed as a possible future extension in Chapter 5. PSAT was selected in preference to Matpower as it appeared to provide a greater set of functionality including conversion of data files from several formats, small signal stability analysis, as well as wind turbine and user defined models. PSAT also provided a community forum and appeared to be more widely used by power engineering students and researchers.

This section has presented the process of selecting external components supported by GridIQ and provided a description of JADE, the agent platform selected, and PSAT, the power simulation tool selected. A brief description is also provided of other tools assessed along with the reasons that they were not selected.

3.4 System Operation

This section describes the functionality and operation of the GridIQ system. Prerequisites, project configuration, as well as simulation execution are discussed. Use Case Modelling has been utilised

to describe the system from a user's point of view in order to capture the system's behavioural requirements. An overview of system use cases is presented in Subsection 3.4.2, with each use case described in detail in subsequent subsections. A sample project included in the GridIQ release in order to demonstrate use of the system is discussed in Section 3.4.11.

3.4.1 Prerequisites

As discussed in Section 3.3, GridIQ utilises external components. All components are free and open source. The components required as prerequisites for using GridIQ are presented below.

Java Development Kit

GridIQ has been developed for the Java platform, and requires a Java run-time environment (JRE) to run. Use of GridIQ also requires the user to compile their own Java source code which requires a Java Development Kit (JDK). The JDK includes the JRE, but also contains additional tools such a Java compiler. GridIQ requires JDK version 1.6 or higher.

PSAT

As discussed in Section 3.3.2, the Power Systems Analysis Toolkit (PSAT) is the supported external tool for performing power flow simulations. PSAT is a library of MATLAB files which require either MATLAB or Gnu Octave in order to run. Separate distributions of PSAT are available for each of these tools. The currently supported version is version 2.1.6, and only the Gnu Octave version is currently supported by GridIQ.

Gnu Octave

Gnu Octave [57] is required to be installed on the local system in order to run the PSAT library. The currently supported version is Gnu Octave 3.2.

JADE

The JADE platform is distributed as a set of Java Archive (JAR) files. These files are required by GridIQ in order to run. The currently supported version is JADE 3.7.

Gnuplot

Gnuplot [58] is the currently supported tool for plotting simulation results. Use of this tool is optional as it is not required in order to run GridIQ, but is recommended to enable users to graphically view results. The currently supported version is Gnuplot 4.2.5.

3.4.2 System Use Cases

Use Case Modelling is a technique used to describe a system from a user's point of view in order to capture the system's behavioural requirements. This subsection presents the interactions between the user, GridIQ, and other systems during operation.
Running GridIQ requires several steps to be performed. These steps are summarised below.

- A network model must be defined in the PSAT data format.
- A set of agents must be defined using the JADE framework.
- Load and generation profiles for network devices, referred to as disturbances, may optionally be defined.
- A project file specifying a mapping of agents to buses in the network and other simulation settings must be defined.
- The simulation is then executed using GridIQ.
- Simulation results may optionally be viewed using a plotting application.

Figure 3.2 presents a UML Use Case diagram illustrating the set of actions able to be performed using GridIQ. The use cases presented define the scope of the system. Actions performed using GridIQ are contained within the GridIQ system boundary. There are, however, actions that must be performed outside of this boundary in order to use GridIQ, such as defining network models and plotting results. These actions are depicted within the File System boundary.

3.4.3 Defining Global Configuration

The global configuration file stores settings which relate to a GridIQ installation as a whole, rather than to individual simulations. This includes details of where external components are installed in

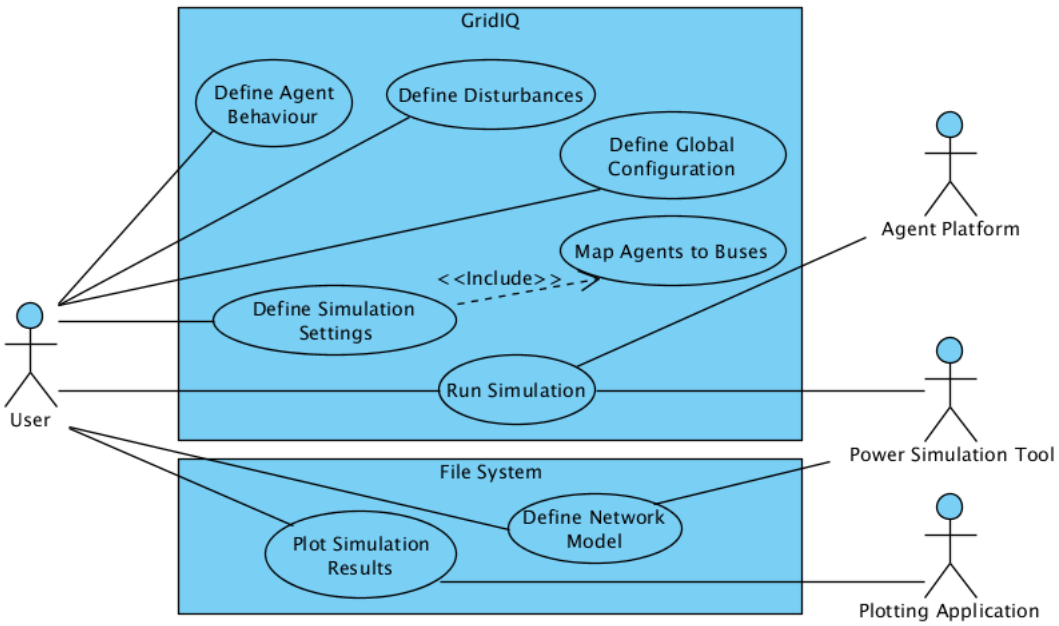

Figure 3.2: Use Case Diagram of the GridIQ System.

order for GridIQ to interact with these components. GridIQ will expect to find an XML file named config.xml in the current working directory which specifies global properties such as the location of the Octave executable file and the PSAT library files. The XML schema of the configuration file is presented in Appendix A. A use case description of defining global configuration settings is presented in Table 3.1.

3.4.4 Defining Simulation Settings

Simulation settings define the configuration for a particular simulation. These settings are defined in a project file containing the following data:

- The network model file to use.
- The number of time steps to run the simulation for.
- (Optional) The Java class defining disturbances to be applied to the network.
- A mapping of agents to buses the network model.

Use Case Number	1
Use Case Name	Define Configuration Settings.
Use Case Description	A user defines global configuration settings specifying paths to external components.
Primary Actor	The User.
Preconditions	None.
Trigger	The user edits or creates a global configuration file.
Basic Flow	1. The user edits or creates the XML configuration file using a text editor or XML editor of their choice. 2. The user specifies file paths to the external component executables or libraries. 3. The user saves the configuration file.

Table 3.1: Use Case 1 Description - Define Configuration Settings

Table 3.2 presents a use case description for defining simulation settings. The schema for the project XML files is presented in Appendix A. The process of mapping agent to buses is described in more detail in the following subsection. Disturbances are described in more detail in Subsection 3.5.2.

3.4.5 Mapping Agents to Buses

A GridIQ project file contains a mapping of agents to buses in the network model, allowing the user to specify a Java class to be instantiated for each agent in the system, and which bus in the network the agent will control. The fully qualified class name must be provided for each agent class, and buses are referred to by the name defined in the network definition file. The schema for the project file and agent mappings is presented in Appendix A. Table 3.4.5 presents a use case description for mapping agents to buses.

3.4.6 Define Network Model

The power network on which to run a simulation must be defined in the PSAT data format. PSAT network files are text files containing a set of MATLAB array definitions specifying properties of devices on the network, and can be created and edited using any text editor. Alternatively, MATLAB

Use Case Number	2
Use Case Name	Define Simulation Settings.
Use Case Description	A user defines project specific settings for a simulation.
Primary Actor	The User.
Preconditions	None.
Trigger	The user edits or creates a project configuration file.
Basic Flow	1. The user edits or creates the XML project file using a text editor or XML editor of their choice. 2. The user specifies a mapping of agents to buses in the network model and other simulation settings. 3. The user saves the configuration file.

Table 3.2: Use Case 2 Description - Define Simulation Settings

Use Case Number	3
Use Case Name	Map Agents to Buses.
Use Case Description	A user specifies which agents are to control which buses during simulation.
Primary Actor	The User.
Preconditions	None.
Trigger	The user edits or creates a project configuration file.
Basic Flow	1. For each bus in the network to be monitored or controlled by an agent, the user specifies the agent class to be instantiated in the project configuration file.

Table 3.3: Use Case 3 Description - Map Agents to Buses

visual editing tools can be used to create network files, but this process is not discussed within the scope of the current project. The PSAT data format is described in detail in the PSAT reference manual [59], and sample networks are distributed with the PSAT library.

Each array in the network file defines a set of related attributes for any number of buses in the network. For example a Bus.con array defines only base properties for all buses, a PQ.con array defines the load attributes for a set of buses and a PV.con array defines regulated generator properties of a set of buses. The PSAT data format defines fifty-four different array structures for defining network attributes. The current implementation of GridIQ supports the following nine array structures required to parse the three and six bus example networks from the PSAT distribution.

Array Name	**Description**
Bus.con	Base attributes for all buses
Line.con	Power Line attributes
PQ.con	Non-regulated load attributes
PV.con	Regulated generator attributes
SW.con	Slack generator attributes
Shunt.con	Shunt Admittance attributes
Supply.con	Power supply attributes
Demand.con	Power demand attributes
Bus.names	Names of buses in the network

A use case description for defining network model files is presented in Table 3.4.

3.4.7 Defining Agent Behaviour

Agents are defined in the Java programming language in order to be compatible with the JADE agent platform. The GridIQ library provides a base agent class *gridiq.agents.GridIQAgent*, which all agents must extend. This class extends the JADE *jade.core.Agent* class, adding functionality allowing agents

Use Case Number	4
Use Case Name	Define Network Model.
Use Case Description	A user defines a network model in a supported format.
Primary Actor	The User.
Preconditions	None.
Trigger	The user edits or creates a network definition file.
Basic Flow	1. The user edits or creates the network definition file using the power simulation tool or a text editor. 2. The user saves the network definition file.

Table 3.4: Use Case 4 Description - Define Network Model

Use Case Number	5
Use Case Name	Define Agent Behaviour.
Use Case Description	A user defines agents using the agent platform.
Primary Actor	The User.
Preconditions	None.
Trigger	The user edits or creates agent source code.
Basic Flow	1. The user edits or creates source code defining agent behaviour using a source editor of their choice. 2. The user compiles the agent code.

Table 3.5: Use Case 5 Description - Define Agent Behaviour

to interact with GridIQ. In order to define agents a user may use a source code editor of their choice, such as the Eclipse IDE. Agent development for the JADE platform is discussed in various tutorials provided on the JADE community website [51]. The compiled agent classes must be placed on the GridIQ classpath when running a simulation as discussed in subsection 3.4.9. Table 3.5 presents a use case description of defining agent behaviour.

Use Case Number	6
Use Case Name	Define Disturbances.
Use Case Description	A user defines disturbances in Java.
Primary Actor	The User.
Preconditions	None.
Trigger	The user edits or creates disturbance source code.
Basic Flow	1. The user edits or creates source code defining disturbances using a source editor of their choice. 2. The user compiles the disturbance code.

Table 3.6: Use Case 6 Description - Define Disturbances

3.4.8 Defining Disturbances

Disturbances define activity on the network over time and are used to set time variant attributes such as load or power generation profiles. Their purpose is to provide underlying network activity that agents can monitor and react to. At each time step, the disturbances defined for the simulation are applied to the buses in the network. Disturbances are defined in the Java programming language. GridIQ provides two classes which can be extended to add disturbances to a simulation. The *Disturbance* class represents a disturbance on a single bus and contains a method, *onTimeStep*, which may be overridden by child classes to implement disturbance logic. This base *Disturbance* class holds a reference to the *Bus* object the disturbance is to be applied to, enabling user implementations to read and set properties of the bus. The *DisturbanceSet* class represents a collection of all disturbances in the network. By providing a subclass of *DisturbanceSet* containing user defined disturbances, the underlying activity on the network can be defined for a particular experiment. The *DisturbanceSet* class contains a *Setup* method which a user may override to add *Disturbance* instances to specific buses in the network. The *DisturbanceSet* implementation loaded by GridIQ at run-time is specified in the project file. This class is dynamically loaded and instantiated when the simulation is initialised. A use case description of defining disturbances is presented in Table 3.6.

Use Case Number	7
Use Case Name	Run Simulation
Use Case Description	A user executes a simulation
Primary Actor	The User
Preconditions	External components are installed and a valid global configuration file, project configuration file and network definition file exist.
Trigger	The user wishes to initiate a simulation for a given project file.
Basic Flow	1. The user runs the gridiq.GridIQ class using the Java run-time environment from a command terminal. The JADE libraries and user defined agent or disturbance class files must be placed on the Java classpath. 2. GridIQ interacts with the power simulation tool and JADE agents to perform the simulation for the required number of iterations. 3. GridIQ outputs the simulation results.

Table 3.7: Use Case 7 Description - Run Simulation

3.4.9 Running a Simulation

A simulation is initiated by running the *gridiq.GridIQ* class from the command line using the Java run-time environment. The main method of this class expects a single argument specifying the project file to use. Any user defined classes defining agents or disturbances must be added to the Java classpath when running GridIQ. GridIQ runs the simulation for the number of time steps specified in the project file and then terminates. As the program executes the current time step is printed to the terminal and power flow analysis results are written to an output directory in a format that can be graphed using the plotting application Gnuplot. This is discussed in more detail in the following subsection.

3.4.10 Plotting Simulation Results

In order to gain a visual representation of network changes throughout a simulation, results can be graphed using a plotting application. The default results format supports plotting using Gnuplot. A set of properties for each bus in the network is output to a separate file in a directory named “output” in the current working directory. Each file contains a time series of the bus voltage, real power load

Use Case Number	8
Use Case Name	Plot Simulation Results
Use Case Description	Results of the simulation are plotted to graphically view the change in bus properties over time.
Primary Actor	The User.
Preconditions	A simulation has been executed and a results file has been generated.
Trigger	The user executes the the plotting application.
Basic Flow	1. The simulation results are opened using the plotting application. 2. The results plot is displayed on screen.

Table 3.8: Use Case 8 Description - Plot Simulation Results

and real power generated for each iteration of the simulation. The GridIQ sample project includes an example script demonstrating how to plot results from multiple files in a single graph. A use case description of plotting simulation results is presented in Table 3.8.

3.4.11 Sample Project

The GridIQ release includes a sample Eclipse project demonstrating a basic configuration for a simulation. The project includes example global and project configuration files, an example PSAT network description file, as well as source code for example agents and disturbances. The sample project also includes step-by-step instructions on running the simulation and viewing the results. These additional instructions are included in Appendix B.

This section has described the usage and operation of GridIQ, discussing prerequisites and use cases of the system. Global and project configuration files were discussed, including the mapping of agents to buses, the definition of network models, agents and disturbances were discussed, as well as the general process of running a simulation using a command terminal. The following section presents the design and implementation details of the system.

3.5 System Design and Implementation

This section details the design and implementation of the GridIQ system. Both structural and behavioural aspects of the design are presented along with details of their implementation. Object-oriented design patterns have been applied in order to utilise existing solutions where common software design problems were encountered. The patterns used are discussed throughout this section.

3.5.1 System Structure

This section describes the structure of GridIQ in terms of classes and components. The relationships between key components in the GridIQ system are illustrated in Figure 3.3. A description of each of these components is given below. A full list of classes implemented is provided in Appendix C.

SimulationController

This component provides a single interface to the rest of the system by implementing the Façade design pattern. The *SimulationController* maintains the current time step of the simulation and interacts with other components in the system to load project and configuration files, execute simulations for the appropriate number of steps, and persist simulation results.

PowerAnalysis

Classes in this component are responsible for directly interacting with the power simulation tool. The Adapter design pattern is utilised to decouple the actual simulation tool used from the rest of the system by defining an interface *PowerFlowSolver*. *PowerFlowSolver* provides a single point of access to the *PowerAnalysis* component for the *SimulationController* by implementing the Façade design pattern. A concrete implementation of the interface, *PSATSolver*, is provided to interact with the PSAT tool. This component is also responsible for reading network definition files and constructing a network domain model, serialising the current network state back to a file, and reading simulation results. Again the Adapter design pattern is used to decouple the network and results file formats from the rest of the system by defining an interface *NetworkParser*. In order to handle network and results files in the format required by PSAT, a concrete implementation *PSATParser* is provided.

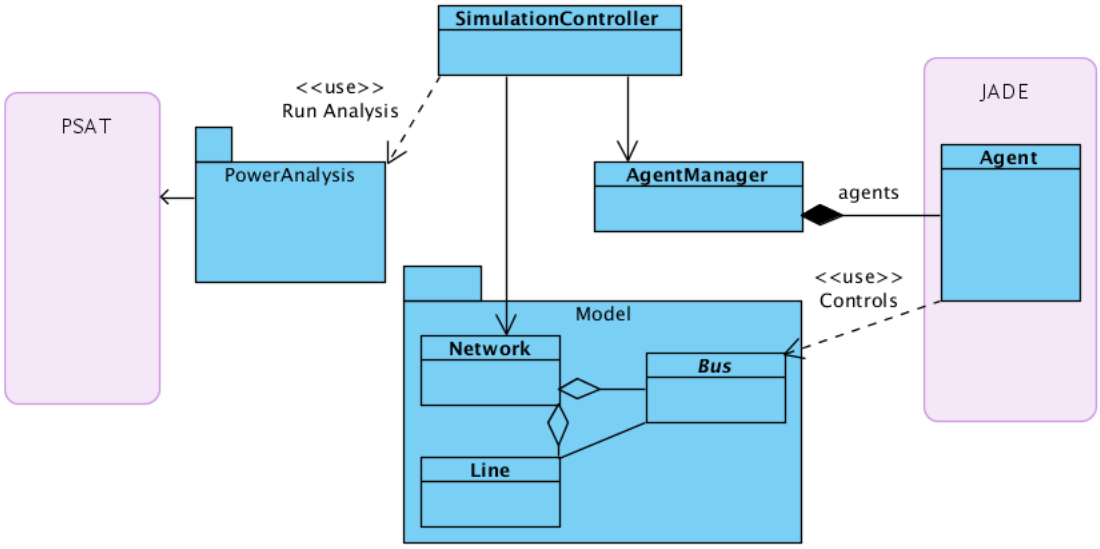

Figure 3.3: Class Diagram illustrating key components in the GridIQ structure.

Model

This component defines the power network model used by the system. Classes within this component store the structure and current state of all devices in the network. A network is comprised of a set of *Bus* objects connected by *Line* objects. Lines are defined by a set of attributes such as length, resistance, reactance and power, voltage and current ratings. A bus is defined by multiple attribute sets defining attributes such as power, voltage and current ratings, load, power generation, shunt admittance, slack bus attributes, and power demand and supply properties. The bus attributes supported correspond to the supported PSAT network attributes as discussed in Section 3.4.6.

AgentManager

The *AgentManager* employs the Façade design pattern to provide a single point of control for all agents in the system. The adapter pattern is also employed by defining an *AgentManager* interface in order to decouple the system from the actual agent system used, allowing other agent systems to be supported in the future. A concrete *JADEAgentManager* is provided to support the JADE agent platform. The functionality of this component is discussed further in Subsection 3.5.2 Managing Agents.

3.5.2 System Behaviour

This section describes the behaviour of classes and components within the GridIQ system, including the sequence of actions performed in order to run a simulation. The implementation of key elements of functionality such as the parsing of PSAT network files, running power flow analyses with Gnu Octave and PSAT, Managing Agents, and persisting results is also described.

Simulation Execution

This subsection describes the sequence of actions performed by GridIQ in order to execute a simulation. Figure 3.4 presents a UML sequence diagram illustrating the flow of execution. A detailed description of this sequence of events is given below.

1 A simulation is initiated The user runs a simulation. The global and project configuration files are parsed to determine simulation settings.

1.1 The network is loaded The network description file specified by the project settings is parsed to construct a network model.

1.2 Agents are started The set of agents specified by the project settings are instantiated with a reference to a bus object in the network model.

1.3 A power flow analysis is executed The power simulation tool is run to solve the network and results are returned to the *SimulationController*.

1.4 Agents are notified of the simulation results The *AgentManager* passes the simulation results for each bus to the relevant agent.

1.5 Agents modify network properties Agents use the simulation results in their decision making process to communicate with other agents and modify the properties of their bus.

Steps 1.3 to 1.5 are repeated for the number of iterations specified in the project settings. Each time the updated network structure is used as the basis for the power flow analysis in the next iteration.

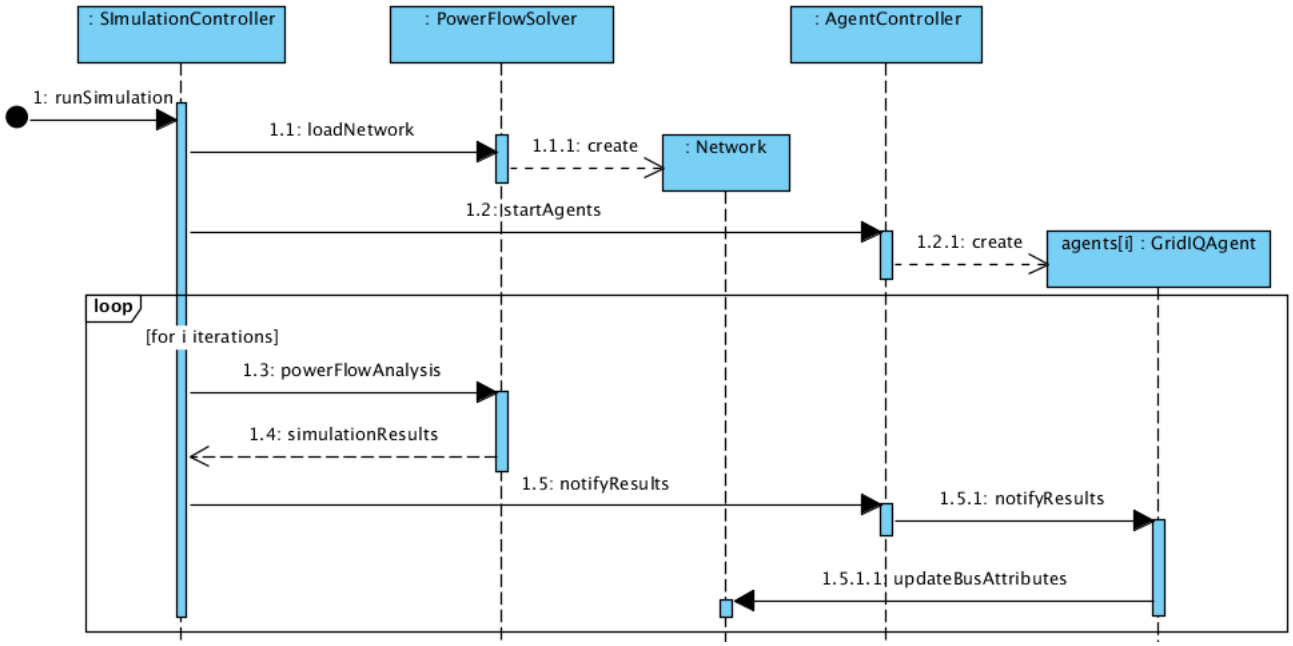

Figure 3.4: Sequence Diagram of Simulation Execution.

Parsing and Serialising PSAT Network Files

Section 3.4.6 introduced the PSAT data format, discussing the fact that it is comprised of over fifty-four array structures describing device properties of which nine are currently supported by GridIQ. When a network model is loaded, GridIQ parses the network file and assembles a data structure representing the state of all devices on the network. Each of these array structures may contain a different number of attributes of different types, and different logic is required to correctly apply each array to the network model. In order to allow a different algorithm to be used to parse each array structure, the Strategy design pattern was employed by defining an interface *DeviceParser*. A class implementing this interface was created for each of the array structures, allowing the parser class to switch its algorithm each time it encountered a new array in a network file.

Before a power flow analysis can be performed on the current network state, the network is serialised to a PSAT network file. As well as defining methods to parse network files, the *DeviceParser* interface also defines functionality for generating the correct text representation of a network object when serialising each device back into a MATLAB array structure.

Running Power Flow Analyses with GNU Octave and PSAT

Integration with Gnu Octave is achieved by executing a system command as an external process from the Java run-time. This functionality is implemented by the *PSATSolver* class. After the network has been serialised to file, a MATLAB script file is generated containing commands to load the PSAT libraries and execute a power flow analysis on the serialised network and output simulation results to a file. This script is then executed with Octave as an external process. GridIQ waits until this external process terminates, at which time the results file is parsed and results are applied as the current network state. This process is performed for every iteration of a simulation.

Managing Agents

The *JADEAgentManager* class is responsible for initialising, destroying, and communicating with agents. On initialisation it creates a JADE run-time container and starts all agents defined for the simulation with a remote reference to the bus that the agent controls. The agent can then directly read and set attributes of the bus. After each simulation step, the agent manager passes a simulation result containing data for a single bus and its outgoing lines to the agent controlling each bus. To achieve this, JADE provides a communication mechanism which enables a remote process to pass an object to an agent called *Object2Agent* communication. Without this mechanism, the only way of communicating with an agents is from another agent using ACL messaging.

Applying Disturbances

As discussed in Section 3.4, the *DisturbanceSet* implementation to be used for a simulation is specified in the project configuration file. An instance the specified class is created by the *SimulationController* when a project is initialised using the default class loader provided by the Java run-time. As discussed in Section 3.4.8, the *Disturbance* class contains a method, *onTimeStep*, which may be overridden to provide disturbance logic. The *DisturbanceSet* class is responsible for calling this method on contained disturbances when instructed by the simulation controller. Disturbances are applied at each time step before the power flow analysis is performed.

Results Persistence

After each simulation time step, the results of the power flow analysis can be stored so that they may be analysed or plotted. The Adapter design pattern is utilised to decouple the actual method used to store results from the system by providing an interface *ResultsWriter*. As discussed in Section 3.4.10 the current system provides a concrete implementation, *GnuplotWriter*, which writes results to a text file in a format that can be used by the plotting application Gnuplot. The system could easily be extended to support other persistence mechanisms such as a database by creating other implementations of *ResultsWriter*.

This section has presented the structural and behavioural design and implementation details of the GridIQ system. The relationships between key components were discussed and their responsibilities described. Object-oriented design patterns have been utilised where design challenges matched well known problems. Key components included the *SimulationController*, *PowerAnalysis*, *Domain*, and *AgentManager* components. The behavioural aspects of the system were presented by illustrating the sequence of actions performed in order to execute a simulation. The implementation of key individual functions such as parsing network files, running external power analyses, managing agents, applying disturbances, and persisting results were also described

This chapter has provided a detailed description of the GridIQ system. The motivation and goals the system were introduced along with the tools and technologies used for development. The open source and agile management processes adopted were discussed, and resources maintained to facilitate open source development were described. External components supported by GridIQ were discussed, as well as an overview of using the system. Furthermore, the system design and implementation was presented in detail.

Chapter 4

Example Scenario

An example scenario has been developed to demonstrate that the GridIQ system does achieve its intended functional objective to enable agents executing within an agent platform to interact with devices in a virtual power network during simulation. This Chapter describes the goals, implementation and results of the scenario. The scenario discussed in this chapter has been released as the sample project discussed in Section 3.4.11 to provide users with an example of using GridIQ.

4.1 Goal

The purpose of the example scenario is to demonstrate that agents implemented using the JADE agent platform can monitor and influence devices in a PSAT network via GridIQ. A specific scenario has been designed in which a set of buses in a network continuously increase their power consumption. The goal of this experiment is to apply agents to devices in the network which will coordinate with each other to cause other buses in the network to modify their power generation to share this increasing demand.

4.2 Procedure and Results

This section describes the methodology used to construct the example scenario and presents the results of the simulations performed. Two simulation cases were developed: a base case in which only disturbances were applied to a network, and a target case in which agents were also applied to the

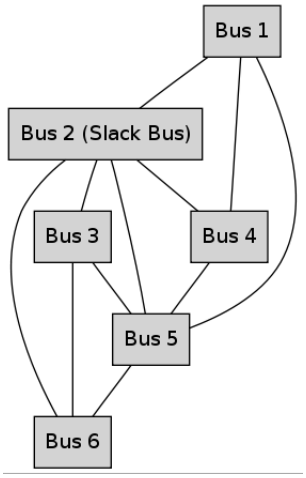

Figure 4.1: The Example Scenario Network

network. This enabled simulation results with and without agent interaction to be compared in order to determine whether the agents are able to interact with the network during simulation.

A network consisting of six buses which is distributed with PSAT as an example network was used as the basis for the scenario. The topology of this network is illustrated in Figure 4.1. Each bus in the network has a base voltage rating of 400kV, while each power line connecting the buses has a power rating of 100MVA and a voltage rating of 400kV. The following subsections describe the base and target scenarios and present the simulation results.

4.2.1 Base Scenario

The goal of the base scenario is to apply disturbances to a set of buses in the network so that activity without agents applied to the network can be determined. A disturbance which causes the real power consumption of a bus to increase at steady rate was defined by extending the GridIQ *Disturbance* class. Disturbances were applied to buses four, five and six to linearly increase their power consumption, each at a different rate. The modified network topology with disturbances applied is illustrated in Figure 4.2.

When the base scenario simulation is run, the bus designated as the slack bus, bus 2, increases its real power to match the increasing demand as expected. Figure X shows the results of running the base scenario. It can be seen that the real power load of buses four, five and six increase linearly due to

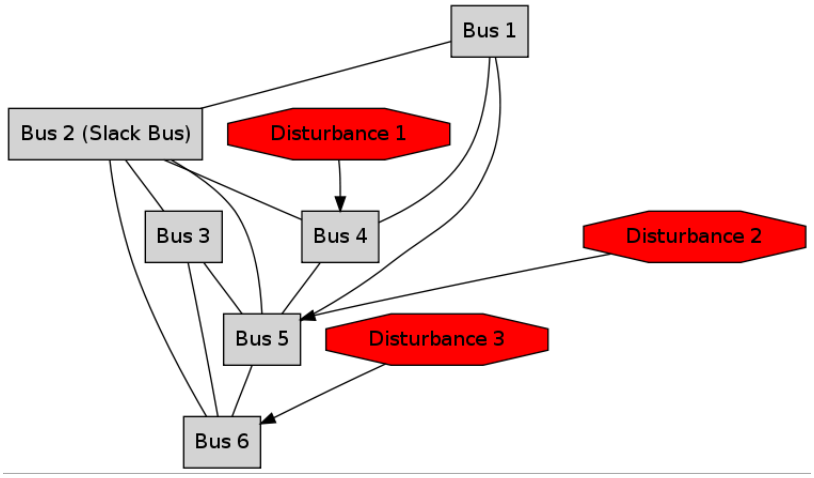

Figure 4.2: The Example Network with Disturbances Applied

the disturbances applied. The power generation of bus two increases to match this demand, while the power generation of buses one and three remains static.

The application of agents into this network is discussed in the following subsection.

4.2.2 Target Scenario

The goal of the target scenario is to apply agents to the network that influence buses to increase their power generation in order to share the increasing demand on the network. Agents were developed by extending the *GridIQAgent* class which enables agents to interact with devices on the PSAT network. Two types of agents have been defined: a warning agent and a control agent. The warning agent reads the real power load at its bus and sends a warning message if it is above a certain threshold. The control agent listens for warning messages and when one is received increases the real power generation at its bus. Warning agents were applied to the buses with disturbances applied: buses four, five and six. Control agents were applied to buses one and three. Figure X presents the example network for the target scenario with both disturbances and agents applied.

The results generated when running this scenario are presented in Figure 4.5. In comparison to the base scenario results shown in Figure 4.3, it can be seen that the power generation of buses one and three now increase to share the increasing demand with bus two. This demonstrates that the warning agents have successfully monitored the increasing load on their assigned buses. The agents

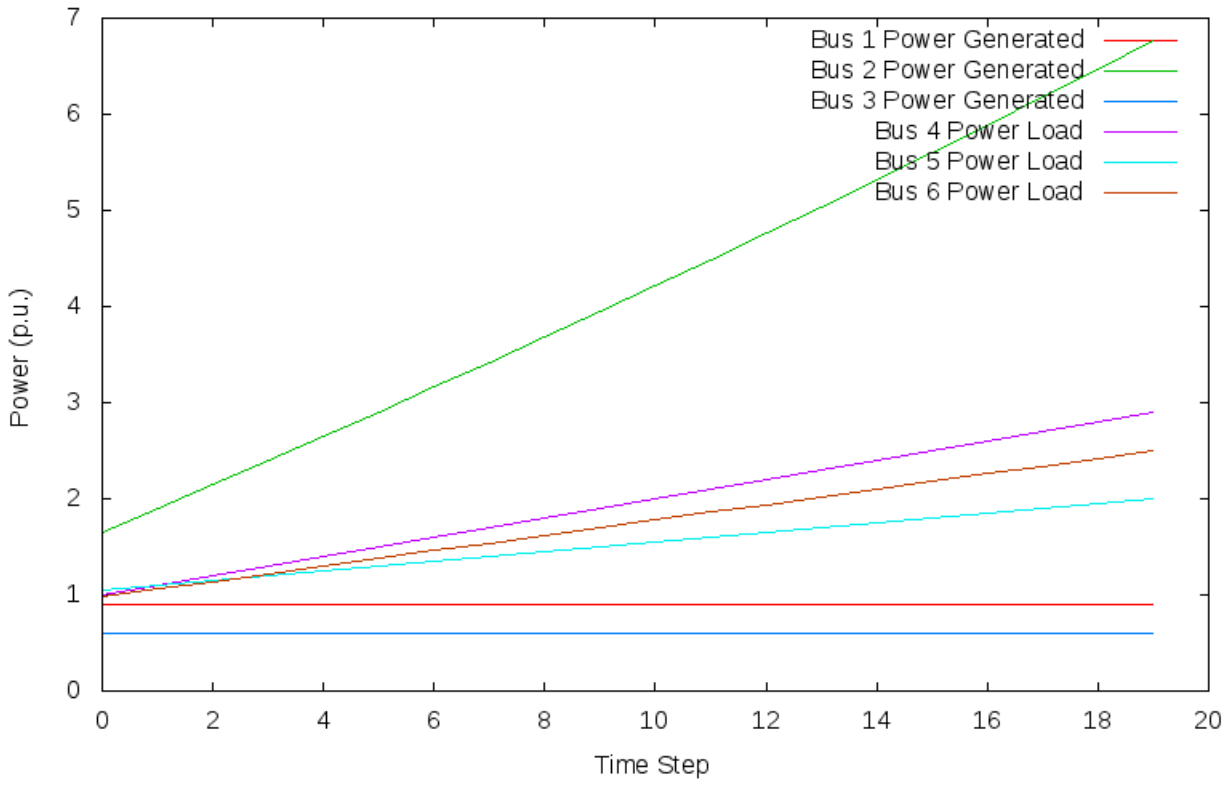

Figure 4.3: Results of the Base Scenario Simulation.

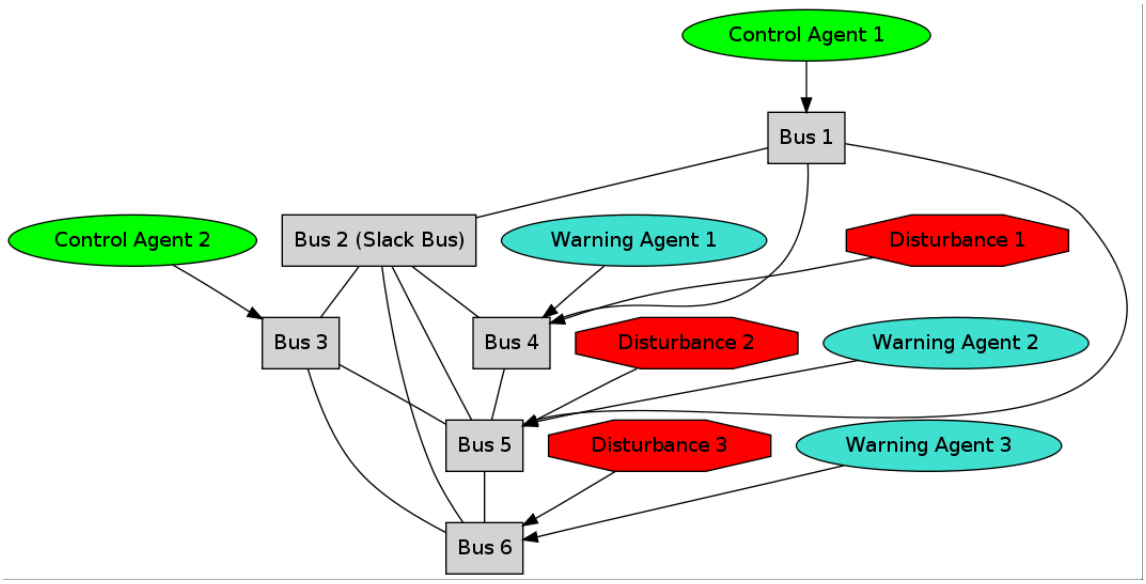

Figure 4.4: The Example Network with Agents Applied

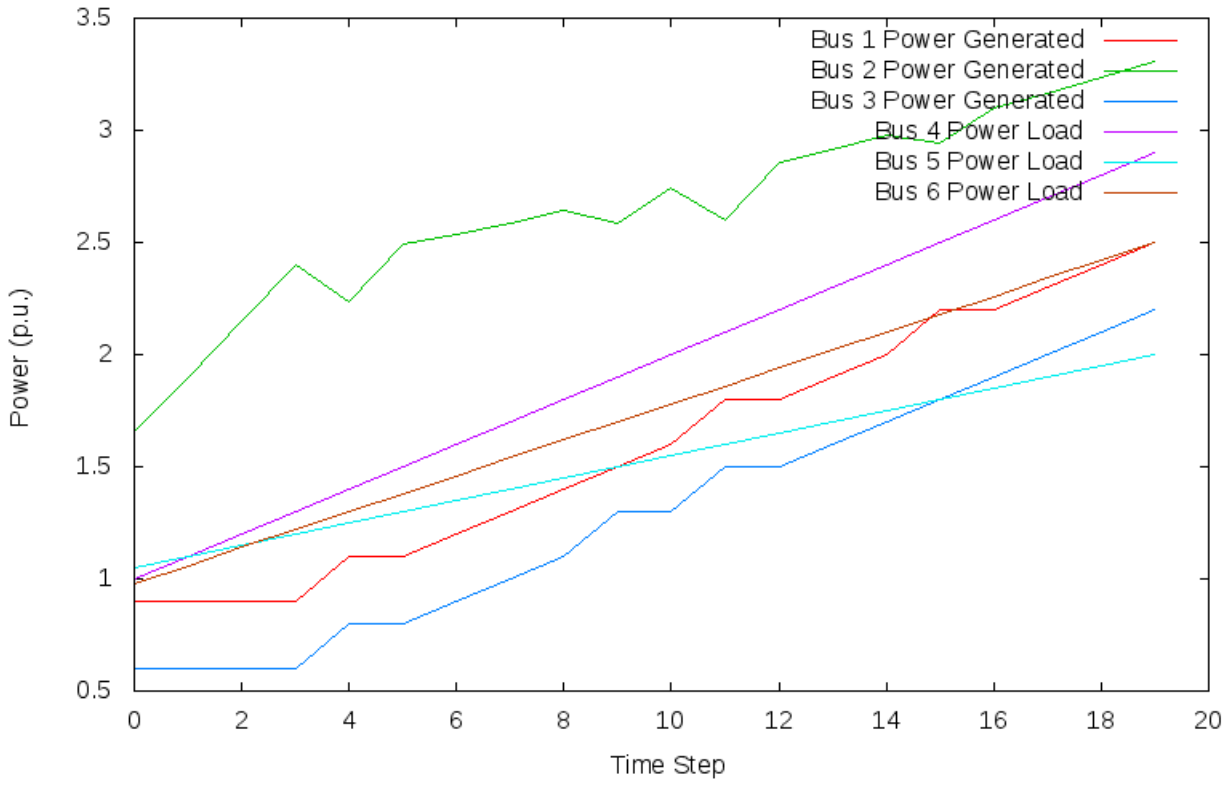

Figure 4.5: Results of the Target Scenario Simulation.

have successfully communicated with each other to transmit warning messages to the control agents, causing them to increase the power generation at their buses.

This chapter has described an example scenario which demonstrates the GridIQ system facilitating interaction between the JADE agent platform and the PSAT power simulation tool. A base scenario was discussed first to show network activity without agent interacts, followed by a target scenario in which agents monitored and influenced the activity of buses in the network.

Chapter 5

Discussion

This chapter provides a discussion on the outcomes of the GridIQ project. The success of the project is assessed against the initial objectives and future work which could be undertaken to expand the project is discussed. Finally, a summary of this document and conclusion are provided.

5.1 Project Success

Overall, the GridIQ project can be labelled a success. The example scenario presented in Chapter 4 has demonstrated the software being applied as specified in the initial objectives outlined in Chapter 3. The scenario shows that GridIQ can successfully enable agents developed using the JADE agent platform to interact with devices in a PSAT power simulation. The objective of managing the project as open source as also been realised. Resources for users and contributors have been created on the Sourceforge.net website and project source code has been maintained in a publicly accessible Subversion repository. The example scenario has been released along with documentation instructing users on creating and running simulations using GridIQ.

5.2 Future work

As with many software systems, in particular open source software, there is no final complete state for the GridIQ project. It is intended to continually evolve to meet the needs of its users. The current

project has successfully released a version of GridIQ which provides basic functionality for applying agents to power simulations. There are, however, a great deal of aspects in which the current implementation could be improved and extended.

An important aspect of future work on the GridIQ project is that it should be driven by the needs of power systems engineers and researchers applying agents to smart grid systems. GridIQ should be applied to a range of problems and scenarios where simulation of agents is required in order to identify key features for further development. User and community involvement are also needed to evaluate the usefulness of current features and to suggest future directions.

Several potential features from which GridIQ could benefit in future releases are described below.

5.2.1 Graphical User Interface

GridIQ is currently run from the command line only and project and configuration files must be edited directly in a text editor. A graphical user interface for the software would enhance its usability and would enable other new features. It could allow users to view a graph of the network and assign agents to buses graphically. It could allow users to locate resources such as the Octave executable and PSAT libraries using a file browser window, rather than entering the path directly into configuration files. Additionally, it could provide a simulation control panel containing graphical buttons to perform simulation actions such as to run the next step of the simulation, run all remaining steps to the end of the simulation, and to stop or pause the simulation while it is running. Results plotting could also be incorporated into the user interface, allowing results to be viewed progressively as a simulation runs.

5.2.2 Performance and Scalability

The example scenario demonstrated in Chapter 4 contained only a small number of buses and agents, and the simulation was performed for only a small number of time steps. Smart grid systems, however, typically consist of a very high number of nodes, and simulations of large time scales may be necessary. Providing greater performance and scalability is a key feature for future development. This may involve support for specialised power simulation tools, the ability to deploy a simulation over multiple networked machines, or optimisations to the iterative approach used by GridIQ.

5.2.3 Support for Other Power Simulation Tools

The current release of GridIQ only supports power simulation using PSAT and GNU Octave. Users may, however, require features of other power simulation tools or other network file formats. This might include features of PSAT that are available when used with MATLAB but not with Octave, or features and formats supported by other open source tools discussed in Chapter 3 or commercial simulation tools. As discussed in Chapter 3, the GridIQ architecture has been designed to allow support for other tools. Future work could implement adapters for such tools.

5.2.4 Support for Additional PSAT Device Types

As described in Chapter 3, GridIQ currently supports only nine of the possible 54 devices specified by the PSAT network format. Future work could extend the domain model classes to support attributes required for these devices. This would enable more complex network designs to be utilised. The PSAT network parser classes would also require extension to read and write these devices to and from the PSAT network files.

5.3 Summary and Conclusions

The introductory chapter of this document has described the challenges facing the power industry and introduced the concept of a smart grid as a potential means to address these challenges. Chapter 2 introduced key concepts in power system analysis and control, and discussed the agent paradigm for software systems as a potential means of providing intelligence to smart grids. A review of existing research applying agents to smart grids and power system control was presented, along with an overview of the tools used in such research. Two key types of tools identified were agent platforms, which provide run time services and development tools for agent systems, and power simulation tools. The fact that there do not currently exist any open source tools that allow agent platforms to integrate with power simulation tools provided the motivation to develop GridIQ, a test bed for smart grid agents.

The details of the GridIQ system were presented in Chapter 3. The goal of the system to act as a bridge between an agent platform and a power simulation tool, as well as the intended set of use cases were described. Managing the project as open source was discussed, including an introduction to the open source software movement and the benefits of collaborative development, as well as a

description of the resources provided for users and collaborators on Sourceforge.net. The concept of agile software development was introduced and the techniques adopted for the GridIQ development process were discussed. The external components supported by GridIQ, the JADE agent platform and PSAT power simulation tool, were introduced along with the process of selecting the supported tools. The operation of the GridIQ software was described in Section 3.4 through detailed descriptions of each use case. The architecture of the GridIQ software was presented describing the relationships between of key classes and components, and the behaviour and implementation of these classes and components was discussed.

Finally, an example scenario was presented which demonstrates the GridIQ system. The scenario shows agents monitoring a six bus network, communicating, and influencing behaviour of the buses. The scenario demonstrates GridIQ successfully enabling interaction between JADE agents and a PSAT power flow simulation. The hypothesis formulated in Chapter 1 that an open source tool can be developed which allows existing agent middleware systems to integrate with power system simulation tools has been supported through the successful development of the GridIQ system. It is hoped that the open source collaborative development model will enable the GridIQ software to continue to evolve into a tool widely used by researchers and practitioners developing agents for the electricity grids of the future.

Bibliography

[1] H. Farhangi, "The path of the smart grid," *Power and Energy Magazine, IEEE*, vol. 8, no. 1, pp. 18 –28, January-February 2010.

[2] M. Hommelberg, C. Warmer, I. Kamphuis, J. Kok, and G. Schaeffer, "Distributed control concepts using multi-agent technology and automatic markets: An indispensable feature of smart power grids," *Power Engineering Society General Meeting, 2007. IEEE*, pp. 1 –7, june 2007.

[3] K. Geisler, "A smarter greener power grid," in *Power Systems Conference, 2009. PSC '09.*, March 2009, pp. 1–3.

[4] C. Potter, A. Archambault, and K. Westrick, "Building a smarter smart grid through better renewable energy information," in *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, March 2009, pp. 1–5.

[5] Modern grid initiative. United State Department of Energy. [Online]. Available: http://www.netl.doe.gov/smartgrid/

[6] M. Albadi and E. El-Saadany, "Demand response in electricity markets: An overview," in *Power Engineering Society General Meeting, 2007. IEEE*, June 2007, pp. 1–5.

[7] D. Hart, "Using ami to realize the smart grid," in *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, July 2008, pp. 1–2.

[8] S. Collier, "Ten steps to a smarter grid," in *Rural Electric Power Conference, 2009. REPC '09. IEEE*, April 2009, pp. B2–B2–7.

[9] A. Nourai, "Large-scale electricity storage technologies for energy management," in *Power Engineering Society Summer Meeting, 2002 IEEE*, vol. 1, July 2002, pp. 310–315 vol.1.

[10] J. Katz, "Educating the smart grid," in *Energy 2030 Conference, 2008. ENERGY 2008. IEEE*, Nov. 2008, pp. 1–3.

[11] P. Fairley. (2010, January) Speed bumps ahead for electric-vehicle charging. IEEE. [Online]. Available: http://spectrum.ieee.org/green-tech/advanced-cars/speed-bumps-ahead-for-electricvehicle-charging

[12] H. Saadat, *Power System Analysis Second Edition*, 2nd ed. McGraw Hill, 2002.

[13] J. J. Grainger, *Power system analysis*. New York : McGraw Hill, 1994.

[14] M. Wooldridge, "Agent-based software engineering," *Software Engineering. IEE Proceedings*, vol. 144, no. 1, pp. 26 –37, feb 1997.

[15] S. Franklin and A. Graesser, "Is it an agent, or just a program?: A taxonomy for autonomous agents," *Proceedings of the third international workshop on agents, theories, architectures, and languages*, pp. 21–35, 1996.

[16] N. Jennings and S. Bussmann, "Agent-based control systems: Why are they suited to engineering complex systems?" *Control Systems Magazine, IEEE*, vol. 23, no. 3, pp. 61 – 73, june 2003.

[17] W. Davis, "The distributed intelligent control of complex systems," in *Intelligent Processing and Manufacturing of Materials, 1999. IPMM '99. Proceedings of the Second International Conference on*, vol. 1, jul 1999, pp. 615 –621 vol.1.

[18] Z. Vale, H. Morais, M. Silva, and C. Ramos, "Towards a future scada," in *Power Energy Society General Meeting, 2009. PES '09. IEEE*, 26-30 2009, pp. 1 –7.

[19] M. Pipattanasomporn, H. Feroze, and S. Rahman, "Multi-agent systems in a distributed smart grid: Design and implementation," *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, pp. 1 –8, march 2009.

[20] Z. Jiang, "Agent-based control framework for distributed energy resources microgrids," *Intelligent Agent Technology, 2006. IAT '06. IEEE/WIC/ACM International Conference on*, pp. 646 –652, dec. 2006.

[21] G. Morejon, S. Srivastava, and D. Cartes, "Integrating virtual test bed and jade agent platform using corba for reconfiguration of shipboard power systems," *Power Engineering Society General Meeting, 2006. IEEE*, p. 6 pp., 0-0 2006.

[22] R. Li, J. Li, G. Poulton, and G. James., "Agent-based optimization systems for electrical load management," *1st International Workshop on Optimisation in Multi-Agent Systems*, pp. 60–69, May 2008.

[23] Y. Guo, A. Zeman, and R. Li, "A reinforcement learning approach to setting multi-objective goals for energy demand management," *7th International Conference on Autonomous Agents and Multiagent Systems*, 2008.

[24] D. Cohen, "Intelligent agent applications for integration of distributed energy resources within distribution systems," *Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy in the 21st Century, 2008 IEEE*, pp. 1 –5, july 2008.

[25] A. Dimeas and N. Hatziargyriou, "Control agents for real microgrids," *Intelligent System Applications to Power Systems, 2009. ISAP '09. 15th International Conference on*, pp. 1 –5, nov. 2009.

[26] K. Schneider, D. Chassin, Y. Chen, and J. Fuller, "Distribution power flow for smart grid technologies," *Power Systems Conference and Exposition, 2009. PSCE '09. IEEE/PES*, pp. 1 –7, march 2009.

[27] P. A. Fritzson and N. Piscataway, *Principles of object-oriented modeling and simulation with Modelica 2.1*. New York: IEEE Press - Wiley-Interscience, 2004.

[28] Aglets agent development toolkit. [Online]. Available: http://aglets.sourceforge.net/

[29] H. S. Nwana, D. T. Ndumu, L. C. Lee, and M. Heath, "Zeus: An advanced tool-kit for engineering distributed multi-agent systems," in *3rd International Conference and Exhibition on the Practical Applications of Intelligent Agents and Multi-Agent Technology, PAAM98*, 1998.

[30] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. Jade: A white paper. [Online]. Available: http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf

[31] Springs agent development toolkit. [Online]. Available: http://sid.cps.unizar.es/SPRINGS

[32] Siemens psse website. Siemens. [Online]. Available: http://www.energy.siemens.com/hq/en/services/power-transmission-distribution/power-technologies-international/software-solutions/pss-e.htm

[33] Digsilent website. DIgSILENT. [Online]. Available: http://www.digsilent.com.au/

[34] F. Milano and L. Vanfretti, "State of the art and future of oss for power systems," *Power Energy Society General Meeting, 2009. PES '09. IEEE*, pp. 1 –7, jul. 2009.

[35] Virtual test bed website. University of South Carolina. [Online]. Available: http://vtb.engr.sc.edu/vtbwebsite/#/Overview

[36] Matlab website. The MathWorks. [Online]. Available: http://www.mathworks.com/products/matlab/

[37] Eclipse website. The Eclipse Foundation. [Online]. Available: http://www.eclipse.org/

[38] K. Coar. (2007) The open source definition. [Online]. Available: http://www.opensource.org/docs/osd

[39] Free software definition. The Free Software Foundation. [Online]. Available: http://www.gnu.org/philosophy/free-sw.html

[40] A. Hars and S. Ou, "Working for free? motivations of participating in open source projects," in *System Sciences, 2001. Proceedings of the 34th Annual Hawaii International Conference on*, 3-6 2001, p. 9 pp.

[41] T. Lewis, "The open source acid test," *Computer*, vol. 32, no. 2, pp. 128, 125 –127, feb 1999.

[42] G. DeKoenigsberg, "How successful open source projects work, and how and why to introduce students to the open source world," in *Software Engineering Education and Training, 2008. CSEET '08. IEEE 21st Conference on*, 14-17 2008, pp. 274 –276.

[43] J. Norris, "Mission-critical development with open source software: lessons learned," *Software, IEEE*, vol. 21, no. 1, pp. 42 – 49, jan-feb 2004.

[44] C. Ebert, "Open source drives innovation," *Software, IEEE*, vol. 24, no. 3, pp. 105 –109, may-june 2007.

[45] J. Dinkelacker, P. Garg, R. Miller, and D. Nelson, "Progressive open source," in *Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on*, 2002, pp. 177 – 184.

[46] Gridiq website. [Online]. Available: http://gridiq.sourceforge.net/

[47] Gridiq sourceforge site. [Online]. Available: http://sourceforge.net/projects/gridiq/

[48] B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed.* Boston, MA: Addison-Wesley, 2004.

[49] K. Beck, *Extreme programming explained: embrace change*. Addison-Wesley, 2000.

[50] M. Fowler, *Refactoring: Improving the design of existing code*. Addison Wesley, 1999.

[51] Jade website. [Online]. Available: http://jade.tilab.com/

[52] Corba website. [Online]. Available: http://www.omg.org/gettingstarted/corbafaq.htm

[53] Interpss website. [Online]. Available: http://community.interpss.org/

[54] Pylon website. [Online]. Available: http://rwl.github.com/pylon/

[55] Matpower website. [Online]. Available: http://www.pserc.cornell.edu/matpower/

[56] Psat website. [Online]. Available: http://www.power.uwaterloo.ca/~fmilano/psat.htm

[57] Gnu octave website. [Online]. Available: http://www.gnu.org/software/octave/

[58] Gnuplot website. [Online]. Available: http://www.gnuplot.info/

[59] F. Milano, *Power System Analysis Toolbox Quick Reference Manual for PSAT version 2.1.2*, June 2008.

Appendix A

Project and Configuration File XML Schema

This chapter presents the XML schemas for the global and project configuration files. The XML schema document (XSD) is presented as well as an example XML document for each configuration file type.

A.1 Global Configuration File

A.1.1 Schema Document

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="GridIQConfig">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="octave"/>
        <xs:element ref="psat"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="octave">
    <xs:complexType>
      <xs:attribute name="path" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="psat">
    <xs:complexType>
      <xs:attribute name="path" use="required"/>
    </xs:complexType>
```

```
  </xs:element>
</xs:schema>
```

A.1.2 Example XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<GridIQConfig>
        <octave path="/usr/bin" />
        <psat path="/home/colin/Applications/psat" />
</GridIQConfig>
```

A.2 Project Configuration File

A.2.1 Schema Document

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="GridIQProject">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="agent"/>
      </xs:sequence>
      <xs:attribute name="disturbanceSet" type="xs:string"/>
      <xs:attribute name="networkFile" use="required" type="xs:string"/>
      <xs:attribute name="timeSteps" use="required" type="xs:integer"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="agent">
    <xs:complexType>
      <xs:attribute name="busID" use="required" type="xs:integer"/>
      <xs:attribute name="class" use="required" type="xs:string"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.2.2 Example XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<GridIQProject networkFile="6_bus_network_mdl.m"
        timeSteps="20" disturbanceSet="sample.MyDisturbances">

        <agent busID="4" class="sample.WarningAgent" />
        <agent busID="5" class="sample.WarningAgent" />
        <agent busID="6" class="sample.WarningAgent" />
        <agent busID="1" class="sample.ControlAgent" />
        <agent busID="3" class="sample.ControlAgent" />
```

</GridIQProject>

Appendix B

Getting Started with GridIQ

This chapter presents the installation and running instructions distributed with the GridIQ sample project. The sample project is discussed in more detail in both Section 3.4.11 and Chapter 4.

```
**************************************
Getting Started with GridIQ
**************************************

GridIQ acts as a bridge between the JADE agent development platform and the PSAT power simulation
    tool.
In order to use GridIQ, you create JADE agents which will control buses in the power network.
    Details on how to develop
agents are provided in the JADE documentation.

Power networks are defined in PSAT format, please see PSAT documentation for details
on defining PSAT networks. The PSAT distribution contains a set of example networks.

This sample project provides an example of using GridIQ. The project contains two simple agents
    controlling a 6 bus network.
The sample project is configured for the Eclipse IDE and is intended to be used
as a starting point for developing more complex agent systems.

**************************************
1. Install Prerequisite Software

GridIQ uses Gnu Octave to run power simulations using PSAT, and Gnu Plot can be used to plot
    simulation results.

Install Gnu Octave, download from http://www.gnu.org/software/octave/ or if using Linux install
    using package installer.

Install Gnu Plot from http://www.gnuplot.info/download.html or if using Linux install using package
    installer.

Download PSAT Octave version from http://www.power.uwaterloo.ca/~fmilano/downloads.htm
Unzip in any chosen location on your computer.

**************************************
2. Open Sample Project in Eclipse
```

```
Unzip SampleProject
In Eclipse File->New Java Project
Create project from existing source - select unzipped location

************************************
3. Configure GridIQ
Edit config.xml file
In the octave path attribute insert the path to the directory where the octave executable is located
    .
Note: this should include the bin directory e.g. [Octave install directory]/bin . Do not include a
    trailing slash.
In the psat path attribute insert the path to the directory PSAT is unzipped

************************************
4. Examine sample project files
The sample project contains 2 agents, sample.WarningAgent.java and sample.ControlAgent.java.
The WarningAgent emits a warning message when the load on its bus exceeds a certain level.
The ControlAgent increases the power generation of its bus each time a warning is received.

The project also defines a set of disturbances. Disturbances define the activity on the network -
    such as loads changing over time.
The file sample.MyDisturbances adds a disturbance to buses 4, 5 and 6 in the network linearly
    increasing their load over time.

The sampleProject.xml file defines the PSAT network file to use and maps agents to buses.
    WarningAgents have been placed on buses 4,5 and 6,
while ControlAgents have been placed on buses 1 and 3.

************************************
4. Run the sample project

Note: Windows users beware of windows firewall preventing JADE platform from working - if a window
    pops up choose "unblock".

In Eclipse:
Create a new Run Coniguration by going:
Run->Run Configurations
Create a new configuration and give it a name
Set Main class: gridiq.GridIQ
Under arguments tab set program arguments: sampleProject.xml

From command line:
navigate to base directory of project, run:
./gridiq.[bat/sh for windows/unix] sampleProject.xml


You should see a set of messages displayed indicating that the simulation is running.
The results are written to the output directory. A .dat file is created for each bus in the system
containing attributes of interest at each time step. Currently voltage, real power generated and
    real power load are output.

************************************
5. Plot results
Results may be plotted with Gnu Plot. An example plot file, plotResults.txt, is provided with the
    sample project.
To plot this file, run:
gnuplot -persist plotResults.txt
from the command line from the sample project directory.

Note: Windows users remember when using gnuplot, you must either use the absolute path to the
    gnuplot executable, or place
the directory contaning the executable in the windows system path variable.
```

```
************************************
This project is still in it's infancy. Any contributions or suggestions from users and developers
    will assist in making it great.
************************************
```

Appendix C

Software Classes in the GridIQ System

This chapter provides a listing of classes and interfaces in the GridIQ system as well as classes which comprise the system tests. Please refer to the API documentation provided at the GridIQ website at [46] for further details on each class.

C.1 System Classes

Classes in the main system are listed below.

```
gridiq.ConfigParser
gridiq.GridIQ
gridiq.ProjectParser
gridiq.agents.GridIQAgent
gridiq.agents.behaviours.ReceiveBusData
gridiq.network.Bus
gridiq.network.Demand
gridiq.network.ElectricityPricing
gridiq.network.Generator
gridiq.network.Line
gridiq.network.Network
gridiq.network.PQLoad
gridiq.network.PVGenerator
gridiq.network.ShuntAdmittance
gridiq.network.SlackGenerator
gridiq.network.Supply
gridiq.poweranalysis.DeviceParser
gridiq.poweranalysis.NetworkParser
gridiq.poweranalysis.ParseException
gridiq.poweranalysis.PowerFlowSolver
gridiq.poweranalysis.PSATSolver
```

gridiq.poweranalysis.disturbance
gridiq.poweranalysis.disturbance.Disturbance
gridiq.poweranalysis.disturbance.DisturbanceSet
gridiq.poweranalysis.psat.parser
gridiq.poweranalysis.psat.parser.BusNameParser
gridiq.poweranalysis.psat.parser.BusParser
gridiq.poweranalysis.psat.parser.DemandParser
gridiq.poweranalysis.psat.parser.LineParser
gridiq.poweranalysis.psat.parser.PQParser
gridiq.poweranalysis.psat.parser.PSATParser
gridiq.poweranalysis.psat.parser.PVParser
gridiq.poweranalysis.psat.parser.ResultsParser
gridiq.poweranalysis.psat.parser.ShuntParser
gridiq.poweranalysis.psat.parser.SlackParser
gridiq.poweranalysis.psat.parser.SupplyParser
gridiq.poweranalysis.results.BusResult
gridiq.poweranalysis.results.LineResult
gridiq.poweranalysis.results.SimResults
gridiq.simulation.AgentManager
gridiq.simulation.GnuPlotWriter
gridiq.simulation.Project
gridiq.simulation.ResultsWriter
gridiq.simulation.SimulationController

C.2 Test Classes

Classes which comprise the system tests are listing below.

gridiq.agents.IncrementPowerAgent
gridiq.agents.SetActivePowerAgent
gridiq.agents.SetVoltageAgent
gridiq.agents.behaviours.AdjustVoltage
gridiq.agents.behaviours.IncrementPower
gridiq.agents.behaviours.SetActivePower
gridiq.poweranalysis.PSATConverterTest
gridiq.poweranalysis.PSATRunTest
gridiq.poweranalysis.disturbance.LoadIncreaseDisturbance
gridiq.poweranalysis.disturbance.LoadIncreaseDisturbanceSet
gridiq.run.RunAgentTest
gridiq.simulation.AgentManagerTest
gridiq.simulation.DisturbanceTest
gridiq.simulation.ProjectParserTest
gridiq.simulation.ResultsOutputTest
gridiq.simulation.SimulationControllerTest
gridiq.test.TestBase