Investigation of Object Oriented Programming Techniques for Embedded Generation Switchgear Design

Paul Robert McCabe

Thesis submitted for the degree of
Doctor of Philosophy
The University of Edinburgh
September 2000

Abstract

The low environmental impact of embedded generation, encouraged by government legislation and private enterprise, has resulted in a large number of such installations being developed in recent years within the United Kingdom electricity supply industry. Unfortunately, a significant proportion of these installations have failed to achieve their full potential as a direct result of poor initial design. This situation has been precipitated by limited capital investment and lack of appropriate design experience. Affected installations are unnecessarily compromised, reducing the overall reliability, efficiency and maintainability of the constructed plant.

An area of particular concern is the design of electrical switchgear within embedded generation systems. Currently, the electrical design for embedded generation installations are prepared without the use of any specialist design tools, software based or otherwise. This situation renders the switchgear design process reliant upon bespoke, ill-defined and un-optimised methods. Such practices are labour intensive, error prone and require substantial expertise in all aspects of power protection and distribution systems.

This thesis investigates the use of object oriented programming to develop a software tool that assists with the complete design and specification of embedded generation switchgear. By capturing the design rationale in such a tool, fast, accurate and checked switchgear designs may be produced by developers without extensive previous switchgear design experience. The thesis describes how the design process for switchgear may be rationalised based upon design methodology, taking account of legal and regulatory codes of practice, such as G59. A complete and general review of artificially intelligent techniques and programming paradigms considered suitable for capturing design reasoning are presented. All aspects of the switchgear design are modelled, including protection and instrumentation equipment, auxiliary power supplies and sundry components. Internal component selection, connection and loading is automated ensuring that a complete, fully specified switchgear installation is produced.

The techniques investigated illustrate that cooperating, interacting networks of software objects may be used to assist and perform switchgear design. The techniques presented could be readily adapted for use in other, non-related design domains, within which complex interdependent architectures and relationships exist.

Declaration

I declare that this thesis has been completed by myself and that, except where indicated to the contrary, the research documented is entirely my own.

Acknowledgements

The work represented in this thesis is only a small fraction of the knowledge and growth that I have accumulated during the course of my study. Without the help, encouragement and support of the following people, this work would have never been completed.

Fortunately for me, Dr. Ewen Macpherson, my supervisor, has great patience. It has been Ewen's ability to work with me at a pace more akin with geological time scales while continuing to provide moral support, ideas and comments has proved invaluable. I am deeply indebted, Ewen.

For rants, raves, news, views, in-depth technical knowledge and enthusiasm for all aspects of power engineering, Dr. Robin Wallace (my second supervisor) has no equal! Thank you Robin for all your help. One day, I hope I have the opportunity to return your generosity and share your craic over a good curry and a pint.

I am indebted to Bob Clark, Inveresk plc, Inverkeithing, for allowing me to gain invaluable practical experience of switchgear installations and their operation.

Many thanks to Peter Watson, Scottish Power plc, Glasgow, for providing me with great insight to the commissioning and operation of embedded generation schemes.

The final stages of this project where completed using the internet and the department's computing facilities. Many thanks to all the computing officers, especially David Stewart and Bruce Hassal.

My thanks and best wishes go to all the members of the Energy Systems Group, especially Gary Connor who has put up with me! Thank you for your generosity and friendship, Gary.

I would also like to thank all my family for their support. Especially my parents, without their emotional and financial support this thesis would have never been completed.

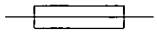
Finally, I have to thank Anne for her love and support. Anne's relentless faith in me, great patience and words of encouragement, not to mention her excellent proof reading skills, have *finally* paid off! Thank you, Anne.

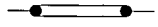
Symbols and Abbreviations

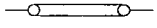
Symbols

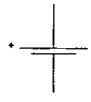
The following symbols have been used throughout the thesis in circuit diagrams:

General devices

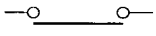

Fuse


Link - bolted contacts

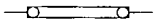

Link - readily separable contacts


Direct current source
(*eg* battery)

Auxiliary switch or protection device contacts

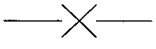

Make contact

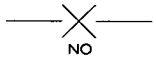

Make contact with delay


Break contact

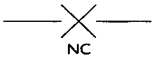

Break contact with delay

Switching


Circuit breaker



Circuit breaker - normally open

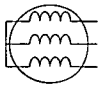


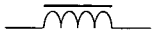
Circuit breaker - normally closed

Machine Windings


AC generator

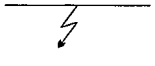
AC motor


AC star connected generator


Transformer or reactor winding
(core indication optional)

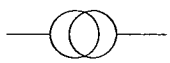
Earthing

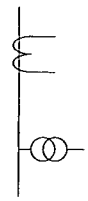

Earth


Earth fault

Power and Measurement Transformers

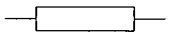

Two windings

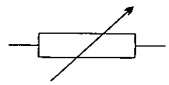

Simplified form

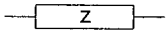

Current transformer

Voltage transformer

Loading


Fixed resistive load


Variable resistive load



Impedance load

Connections

Single phase connection

Three phase connection

Abbreviations

The following abbreviations have been used throughout the thesis:

Symbol	*Definition*
k	kilo - 10^3
M	Mega - 10^6
T	Tera - 10^9
kB	kilobyte (unit of computer storage space)
m	meter (unit of distance)
MB	Megabyte (unit of computer storage space)
n/a	non-applicable
e.g.	*exempli gratia* - for example
et al	*et alii, et alia...* - and others
ibid	*ibidem* - in the same book or passage
i.e.	*id est* - that is to say
p.	page
pp.	pages

Contents Summary

Contents

List of Figures

List of Tables

Return, good Catesby, to thy lord again;
Tell him, myself, the mayor and citizens,
In deep designs and matters of great moment,
No less importing than our general good,
Are come to have some conference with his grace.

–Buckingham, King Richard III, Act III, Scene VII, William Shakespeare.

Chapter 1

Introduction

For the long term development of modern civilisation, mankind must become reliant upon sustainable electricity production. This situation was recognised by all European member states at the Kyoto protocol on climate change. At Kyoto, the United Kingdom government committed to substantially increasing the capacity of embedded generation over the next decade. Within the United Kingdom, there are recommendations and statutory requirements that apply to the construction and operation of any embedded generation scheme. However, the rapid expansion of embedded generation in the near future will result in a profusion of differing design philosophies and implementations. If the new generation of embedded generation plant is to gain credibility, through efficient deployment and consistent performance, the design and specification of such plant must be rationalised. This thesis describes the development of object oriented programming techniques that may be applied to construct software capable of performing the organised design of embedded generation plant. The techniques developed could be applied to the design of any multi-component artefact.

1.1 Background and Motivation

The term *embedded generation* describes the connection of independent, limited capacity (<10MW) generators connected to the electricity distribution network. The exact technical definition of embedded generation varies, as the boundary between transmission and distribution networks varies between countries throughout the world. This thesis will focus upon the United Kingdom electricity supply industry within which embedded generators are generally regarded as having ratings less than 10MW and are connected at 33kV or below.

Embedded generation is not a new development within the electricity supply industry [226]. However, the last two decades have seen an increased expansion

of embedded generation schemes as government legislation[1] [33] and private enterprise have responded to growing public concern for the environment. The beginning of the 1990's witnessed the United Kingdom government taking an active involvement in environmental concerns, developing interrelated policies regarding energy efficiency programs, pollution reduction – especially carbon dioxide emissions – and the encouragement of renewable energy schemes.

Political incentives have resulted in the substantial and sustained development of embedded generation within the United Kingdom, based upon two electricity generation technologies: combined heat and power (CHP) schemes and renewable energy (RE) electricity generation. Typically, these technologies have either a relatively small electrical capacity and/or a remote location, and hence are required to be connected at distribution voltages. Thus they are classed as embedded generation.

Combined heat and power schemes allow the simultaneous production of heat and electricity in an efficient manner. By using a gas or steam turbine to drive an electrical generator, the exhausted gases or steam are utilised for heat intensive industrial processes, *e.g.* the manufacture of paper, or for commercial or domestic water and space heating. The overall efficiency of a combined heat and power plant is typically between 85% – 90%. In comparison, thermal processes, including power generation, have characteristic efficiencies between 30% – 40% [103].

The high efficiency of such plants has resulted in the technology being a crucial component of the government's energy efficiency policies. The current government's target of 5GWe of combined heat and power electrical capacity by the end of the year 2000 appears to be on course. A new target of 10GWe by the year 2010 as part of the current government's Climate Change Program will be announced later this year. However, the setting of such targets has not been met with universal approval [194]. In 1998 combined heat and power schemes supplied 6% of the electricity generated in the United Kingdom, with over 90% of these schemes (1304 in number) having ratings less than 10MWe and therefore classed as embedded generation [198, p.20].

Renewable energy resources have substantial environmental benefits [273] and as a result have been subject to substantial government support. In 1989 the United Kingdom government passed the Electricity Act to transfer ownership of the electricity supply industry back to the private sector [195]. The act also included, under Section 32, the obligation of the industry to comply with renewable orders. In 1990 the government announced the first Non-Fossil Fuel Obligation (NFFO

[1]Such changes in energy policy have not only been reflected within United Kingdom legislation. The United Nations Conference on Environment and Development, also known as the *Earth Summit*, in Rio de Janeiro, June 1992, reflected world wide concern and agreement on climate change. More recently, as a result of the Kyoto conference, December 1997, such policies have been re-enforced by all European member states.

1) to encourage the development of renewable energy resources in England and Wales. The success of this initiative has resulted in a subsequent four additional obligations (NFFO 2–5), three Scottish Renewables Obligations (SRO 1–3) and two Northern Ireland Non-Fossil Fuel Obligations (NI NFFO 1–2); collectively referred to as the *renewable obligations*.

The contracts awarded to developers under the renewable obligations are guaranteed a fixed price per unit of electricity generated for a fixed term. The additional cost of this generation over and above fossil fuelled generation is met by the Fossil Fuel Levy which is paid by all United Kingdom electricity consumers. An indication of the number and size of schemes developed under the renewable orders is presented in Table 1.1.

		Projects Contracted		Projects Generating		Projects Terminated		Projects To Be Commissioned		Completion Rates	
		Number	*Capacity (MW DNC)*	*Number*	*Capacity (MW DNC)*	*Number*	*Capacity (MW DNC)*	*Number*	*Capacity (MW DNC)*	*Number (%)*	*Capacity (%)*
NFFO 1	*1990*	75	152.1	61	141.5	14	7.6	0	0.0	81	93
NFFO 2	*1991*	122	472.2	82	172.6	40	298.5	0	0.0	67	37
NFFO 3	*1994*	141	626.9	70	248.0	2	1.9	69	377.0	50	40
SRO 1	*1994*	30	76.5	13	27.8	0	0.0	17	48.7	43	36
NI NFFO 1	*1994*	20	15.6	13	14.6	0	0.0	7	1.0	65	94
NFFO 4	*1997*	195	842.7	44	98.1	0	0.0	151	744.6	23	12
SRO 2	*1997*	26	113.9	3	6.7	0	0.0	23	107.2	12	6
NI NFFO 2	*1996*	10	16.3	4	0.8	0	0.0	6	15.5	40	5
NFFO 5	*1998*	261	1177.0	9	14.7	0	0.0	252	1162.3	3	1
SRO 3	*1998*	53	144.2	0	0.0	0	0.0	53	144.2	0	0
Total:		933	3637.5	**299**	**724.8**	56	308.0	578	2604.7	-	-

Table 1.1: Current status of United Kingdom Renewable Orders [195, 196, 197, (Adapted)].

Further restructuring of the electricity supply industry planned by the current government will affect future renewable orders.[2] However, NFFO 5 and SRO 3 provided substantial evidence that renewable generation is continuing to converge towards market prices [195, 196]. Indeed municipal and industrial waste projects are already economic without NFFO support, thus for the foreseeable future, the prospects for the continued development of embedded generation appear to be excellent. This is also encouraged in the longer term by the government's target to produce 10% of the United Kingdom's electricity supply from renewable sources by the year 2010 [195, p.5]. To meet this target, the government will have to encourage over a twelve fold increase in the total generation capacity available from renewable energy sources and a four fold increase in combined heat and power capacity within the next decade [20].

[2]On the 28th of July, 2000, the *Utilities Act* became law, resulting in the replacement of the existing Electricity Pool with the new electricity trading arrangements (NETA) [102]. Under the NETA all electricity suppliers must produce electrical power from renewable energy sources in accordance with set targets. To encourage the use of electricity generated from renewable or combined heat and power plants, the *Climate Change Levy* will become law by April 2001. The levy requires business customers to pay an additional 0.43 pence per unit of electricity consumed which has not been produced from either renewable energy or combined heat and power sources [71].

With the majority of large primary resources already exploited, future projects will have to become increasingly small (in terms of electrical generating capacity) to exploit remaining renewable energy resources or opportunities for utilisation of combined heat and power plants [280, 72].[3] There is already evidence to suggest that this trend is apparent with the mature renewable energy technologies, especially hydro power [268]. Furthermore, European Union's Electricity and Gas Directives are forecast to encourage the construction of small, clean and localised electricity generation stations, with combined heat and power plants becoming wide spread [264]. The development and availability of automated electronic control and switching equipment for inexpensive, small generating plants have assisted this trend.

Due to the marginal economics of renewable embedded generation schemes the initial design process is a crucially important factor in the economic viability and performance of any proposed embedded generator, but particularly so on smaller scale schemes. Indeed, the importance of decisions taken during the early stages of the design process and their later consequences has been widely recognised [224, 75, 147, 78]. Furthermore, Stewart [257] has indicated that up to 90% of civil structure failures, malfunctions or poor serviceability are attributed to human error, where at least 33% of such failures may have been avoided if design checking had been adequate. Unfortunately, similar figures for embedded generation do not currently exist.

1.2 Design of Multi-Component Systems

The advancement of modern technology has precipitated the development and construction of artefacts of escalating complexity. The design of such artefacts has only been made possible by employing techniques to manage the complexity inherent within these artefacts. This is particularly true in engineering. In order to allow the development of modern artefacts, designers employ design strategies to allow them to manage complexity, the most common of which is divide and conquer, frequently referred to as *top-down design*.[4]

The application of the design strategies, especially top-down design, frequently results in boundaries being formed between constituent aspects of the artefact. Often the boundaries formed allow the final artefact to be divided into two or more components with a common interface. However, it is usual that the design of one

[3]Such schemes alluded to here would not apply to renewables obligations for support as they would not be economic; they would be privately funded and in direct discussion with the local public electricity supplier.

[4]Refer to Section 2.2 for a more complete discussion of design methodology.

component impacts upon several of the other constituent components of the artefact. Within modern engineering design, such as embedded generation systems, the number of constituent components and their interdependencies can be very large indeed. The result is that it can be exceedingly difficult for designers to accurately manage the design process.

1.3 Design of Embedded Generation Systems

Presently, no commercially available, industry standard software application exists that assists or performs the necessary design reasoning and specification production common to the majority of embedded generation schemes. Furthermore, there currently exists no generic, flexible software architecture that allows the capturing and reproduction of design reasoning for multi-component artefact design. The development of such a software architecture is necessary if the electrical design of embedded generation systems is to be modelled in a computationally interpretable and flexible form. The design and specification of multi-component assemblies can be captured through the creation and development of a general software design architecture, based upon the object oriented programming paradigm, combined with the metaknowledge of the application domain. This allows design and specification tasks to be performed in shorter time frames with greater accuracy and reduced effort. The application of such a software structure to embedded generation design, that includes a graphically driven user interface, would allow developers with limited experience to produce high quality specifications that are consistent and accurate.

Renewable embedded generation utilises a wide range of diverse engineering technologies with which natural forces or materials are harnessed to produce energy. By examination it is apparent that the switching, instrumentation and protection requirements for many embedded generation schemes are similar, both in electrical and operational terms. These are also the areas where the majority of developers have significant difficulties, as they are frequently familiar with the technology associated with the energy source or fuel and the selection of an appropriately sized machine to exploit the resource. The detailed electrical design, however, including the switchgear selection, protection equipment requirements and instrumentation, are not typical skills of such developers.

This project seeks to develop a framework that allows the design rationale for multi-component systems to be captured in a computational form. Embedded generation switchgear is a good example of a multi-component system and has been selected in this project to demonstrate how multi-component system design can be modelled. By concentrating upon switchgear design for embedded generation

(which includes instrumentation and electrical protection), capturing the design process for these artefacts in software, efficient, consistent and accurate switchgear specifications applicable to the majority of new embedded generation schemes can be achieved.

1.4 Scope of the Thesis

The investigation presented in this thesis bridges two fields of research which are part of an ongoing series of projects in the Energy Systems Group, the Department of Electronics and Electrical Engineering at the University of Edinburgh. These two research fields are:

1. Embedded generation and renewable energy systems (focusing particularly upon small scale hydro power), and
2. Artificially intelligent techniques in design.

The exploration of artificially intelligent techniques in design commenced by capturing design rationale utilising a procedural programming language, namely C. However, the limitations of this approach, especially its inflexibility, became apparent and subsequent research focused upon the use of expert systems. More recent studies have utilised genetic algorithms (combined with geographical information systems) to tackle more complex design problem domains. The most closely related projects are summarised below in chronological order.

Pelton Turbine Design: The requirement for decentralised power generation in the rural areas of developing countries is vast, and extensive hydrological resources available for small-hydro power development have been identified. However, the design procedure for turbine design is non-trivial. This project developed a new method of designing multi-jet vertical Pelton turbines and estimating their performance. This design method was developed and written, using the procedural programming language C, into a comprehensive software package. A test facility was constructed to verify the software's output.[5]

Micro-Hydro Design: Engineering knowledge in developing countries, both experimental and professional, is particularly valuable, especially for micro hydro power design and development. Unfortunately, such knowledge is easily lost. The assessment of micro hydro sites requires a wide range of skills. This project developed an expert system coupled to a database that allowed

[5]Investigated by Robin Wallace [276].

this assessment to be assisted and a decision made as to whether an installation should be undertaken. This research was targeted for Nepal where such knowledge is in short supply and the potential for hydro development is very large [9].

Power Electronic Design: The design of power electronic systems requires considerable expertise in exploring a wide range of complex and tedious tasks. Faster design times and more efficient design are among the advantages that can be achieved using artificially intelligent technologies, such as expert systems, to assist with such tasks. This project explored the use of an expert system, linked to simulation tools to assist the design of switch mode power supplies, developing techniques that may be applied to the design of other electronic systems.[6]

Embedded Generation Simulation: The introduction of embedded generation to rural distribution networks can compromise overall system operation, security and integrity. The technical issues associated with embedding generation plants through numerical simulation was explored, resulting in the production of hierarchical rules to be used to rapidly identify whether plants may be successfully embedded at particular locations.[7]

Wind Farm Design: Both traditional and external costs are associated with wind generation plant. In order to provide the optimum true cost wind turbine layout a large number of technical and social factors must be considered. An optimal design can be determined utilising genetic algorithms combined with geographical information systems (GIS). The software developed efficiently produced wind farm layouts significantly better than those designed by humans.[8]

The research on embedded generation simulation was ongoing at the commencement of this project. However, it was envisaged that the rule base produced could be, at some juncture, incorporated into this research. This would not only provide assistance with the design of the switchgear and associated components, but also the resulting specification could be assessed to ensure stability and integrity of the plant when connected to the distribution network.

The experience acquired from the construction of a software application to design multi-jet Pelton turbines has indicated, in concordance with other research,[9] that the selection of a purely procedural programming approach to capturing design reasoning has significant limitations. Therefore research projects subsequent

[6] Investigated by Amarnath Reddy [161, 217].
[7] Investigated by Steven Stapleton [255, 254].
[8] Investigated by Gary Connor [52].
[9] Refer to Section 4.4.7.

to this work have utilised artificial intelligence software technologies to capture design knowledge concerning a particular domain.

The research exploring power electronic design was nearing completion as this project commenced. The experience and knowledge gained, however, highlighted several difficulties and limitations of capturing design reasoning in a computer interpretable form.[10] Therefore this project also attempted to explore alternative implementation technologies.

1.5 Project Aims

During the next decade, if the government is to achieve its Kyoto commitments, the expansion of renewable embedded generation will be rapid and widespread. The success of embedded generation will depend upon an industry standard, rationalised, efficient design process that ensures consistent performance of such plant.

The development of software capable of encapsulating and performing the design necessary to allow the automation of embedded generation systems design may be achieved through the creation of an object oriented hierarchy. Such a hierarchy must reflect the organised and interrelated nature that exists within the multiple components that comprise any embedded generation installation. However, the software architecture developed must be flexible, both in terms of applicability to other design domains and future adaption and/or maintenance. Hence the aim of this project may be summarised thus:

> ***To develop an object oriented architecture that allows the capturing of design rationale for multi-component artefacts in a computationally interpretable form, such that the effects of individual component parameter alterations are propagated automatically throughout the various representative objects, ensuring that an accurate and consistent specification for the overall artefact is achieved.***

In order to achieve this aim, the project was partitioned into the following divisions:

1. A study of currently installed embedded generation, assessing design attributes, methods and possible process development.
2. A study of design methodology and mechanisms that allow human practitioners to produce designs.

[10] These issues are explored in greater detail in Section 4.4.1.

3. A review of possible software technologies for the development of a design tool, considering each technology's advantages, limitations and use.

4. From the prior studies, select an implementation technology that allows the construction of a flexible architecture for capturing design rationale within which switchgear design may be captured.

5. Finally, the design tool created should be demonstrated to verify the integrity of the tool and investigate the implications arising from its use.

The fundamental premise on which this research hinges is:

> The object oriented programming approach allows the encapsulation of design rationale within a flexible programming language. The hierarchal arrangement of objects and the creation of a complex network of interrelationships between objects allow intelligent design reasoning to be exhibited by a software application embodying this approach.

1.6 Impact Areas

The nature of the research presented in this thesis crosses several diverse disciplines, resulting in a voluminous literature survey. The diversity of documents and approaches within one field of research frequently leads to differences of opinion upon the critical evaluation of results. When several fields of research are involved, as in this instance, this problem is intensified.

This thesis focuses upon three areas of knowledge which, when combined, allowed the prototyping of a tool for embedded generation switchgear design:

1. The development of a set of design practices and the modelling of leading design dimensions which thus allow the design reasoning of switchgear and associated components to be ordered into a hierarchal sequence. Within this process, legal, regulatory and common codes of practice have been incorporated. This process provides a demonstration of how an engineering task may be rationalised in order for it to be modelled in software.

2. A comprehensive review of the field of artificial intelligence and key programming paradigms with a view to their practical application for artificial design has been undertaken and reported in this thesis. This has allowed the appropriate selection of an implementation technology to be selected not only for the purposes of this research, but for future projects in design process modelling in other domains.

3. The object oriented programming techniques described in this thesis, although not novel, had not been applied directly to a power engineering design task before. Furthermore, this research attempts to combine design knowledge and processes specific to an engineering design domain into a hierarchy of cooperating objects without relying upon external data or knowledge bases. Although this approach introduces some limitations, it is hoped that the techniques developed will be of use to other areas of engineering design or that the model presented will be extended.

The thesis concludes with a discussion of the implemented design tool. Exploration of the object oriented design hierarchy, including limitations and possible alternatives to alleviate these issues, are described. Possible further extensions of the model are discussed resulting in the identification of a number of areas of future research.

1.7 Thesis Outline

The research presented in this thesis spans a four year period of study into artificially intelligent design practices of switchgear and associated components for embedded generation schemes. The work was performed in several stages and this situation has been reflected in the structure of the thesis.

Initially a study of the electrical installation design of embedded generation schemes was explored and this work is presented in Chapter Two, combined with a review of design methodology, describing the fundamental theory of this embryonic discipline. This part of the thesis considers the design methods employed by human practitioners and how these techniques may be rationalised and emulated. Having considered design rationale, the discussion returns to the embedded generation design process and how this may be rationalised, and formulated into a design hierarchy, in accordance with statutory regulations.

Chapter Three describes a comprehensive review of artificial intelligence technologies and fundamental programming paradigms. The artificial intelligence community have developed several technologies that attempt to emulate human reasoning. Since the practice of design is a distinctly human process, these technologies are presented to explore their suitability to modelling the art of design. Computer programming languages are also presented to allow a comparison to be drawn between artificial intelligence and other software technologies.

Chapter Four develops the discussion presented in Chapter Three, by analysing the suitability of the software technologies previously presented with a view to capturing a limited set of human design skills; a process which will be referred to as

artificial design. Examples of the construction of artificial design systems, classified by software technology type is also presented. The closing discussion in this chapter describes the reasoning underpinning the final software technology selection.

The methodology of object oriented programming adopted for the construction of a tool for switchgear design is presented in Chapter Five. This chapter includes a description of the various design techniques which the software emulates as well as the component hierarchy, dependencies and attributes modelled in objects. A discussion of various approaches to modelling individual components is also included throughout the chapter.

The operation of the software on a typical switchgear design situation is presented in Chapter Six. This chapter indicates how the hierarchy and network of design objects discussed in Chapter Five, combined with a graphical user interface, allows the user to quickly and accurately design switchgear for embedded generation plant.

Finally, Chapter Seven discusses and summarises the work described in this thesis. Several areas of future research are suggested for the development of the software tool for switchgear, protection and instrumentation design. Final conclusions are drawn about the suitability of object oriented programming techniques for use in the development of artificial design tools and the general design hierarchy developed through this project.

1.8 Chapter Summary

This chapter introduces and defines embedded generation. The motivation of this thesis of automating the design process for embedded generation switchgear has been presented utilising a generalised object oriented design hierarchy. The areas of contribution provided by the thesis and an overview of its structure has also been presented.

Chapter 2

Embedded Generation, Design Methodology & Rationalisation

This chapter is composed of three distinct but interrelated areas. Initially, the technical background of embedded generation is presented, describing the equipment required for installation and the purpose of the individual components which constitute an installation. Secondly, a review of design methodology is presented, exploring the methods utilised by human practitioners with a view to incorporating these techniques into the software tool. Finally, a rationalisation of the design process of switchgear and associated components for embedded generation is explained, indicating how aspects of the process may be automated or inferred. The chapter closes with a summary of the key concepts presented from each area.

2.1 Embedded Generation

Embedded generation is not a recent development. At the turn of the last century, the United Kingdom electricity supply industry was in its infancy, consisting of independent generating stations connected to local distribution networks. Reliability and performance of supplies varied, therefore in a bid to standardise and unify the industry the 132kV National Grid was constructed allowing larger, more efficient power stations to be developed and the bulk supply of electric power to occur. The electricity was then transmitted to areas of high demand and supplied to loads via the distribution network. In 1947 the electricity supply industry was nationalised [172], and, coupled with the rapidly growing demand for electricity throughout the United Kingdom [138, 139], resulted in the *supergrid* being constructed. Operating at 275kV or 400kV, the supergrid provided the bulk transmission of electricity to the then well established distribution networks, produced by very large centralised power stations [130].

Independent generation still existed, directly connected to local distribution networks at either 132kV or 33kV, operating to increase both national and local supply security [226]. Such plants, however, were utilised to meet the specialist requirements normally associated with the industrial sector, which could justify, economically, the high costs associated with such plants. No commercial bias then existed concerning the dispatch of embedded generators, hence no regulatory controls were required. Typical industrial applications included stand-by or peak load generation normally utilising steam turbine or diesel generation sets [100].

However, the mass production of compact, efficient and low cost generation equipment combined with reliable electronic and microprocessor control has made it feasible to construct and remotely operate embedded generation. Coupled with growing environmental awareness, the economical advantages of combined heat and power plants and improved electricity supply reliability have all been principal factors driving embedded generation development [134, 133].

In contrast to the operation of large scale generation which is scheduled to meet demand, embedded generation operates as determined by either the heat requirements of a combined heat and power scheme or by the energy available from a renewable energy resource. Embedded generators exporting less than 10MW to the public distribution network are not, as a rule, centrally despatched by the National Grid Company (NGC) [126]. There are, however, some embedded schemes which respond to electronic tariff signals or other charging mechanisms; all such installations are dependent upon fuel or energy availability.

A block diagram of a typical generation station is depicted in Figure 2.1; although this is a typical layout for most generating stations, it is not universal. For example, fuel cell technologies convert their respective energy sources directly into electricity [81], thus only requiring control gear, switchgear and protection gear. Given the wide range of embedded generation technologies with their large range of specialist equipment for prime moving, generating and control, it is the focus of this project to attempt to automate the common sections in Figure 2.1; namely the design of the switchgear and protection equipment.

The following sections will describe the legal and regulatory documents associated with the implementation of embedded generation, the common electrical components required to construct such a plant and current design practice of developers of embedded schemes.

2.1.1 Legislation and Regulations

Until the 1960's many industrially based embedded generators cooperated with the public electricity supply industry to maintain security of supply. Without any

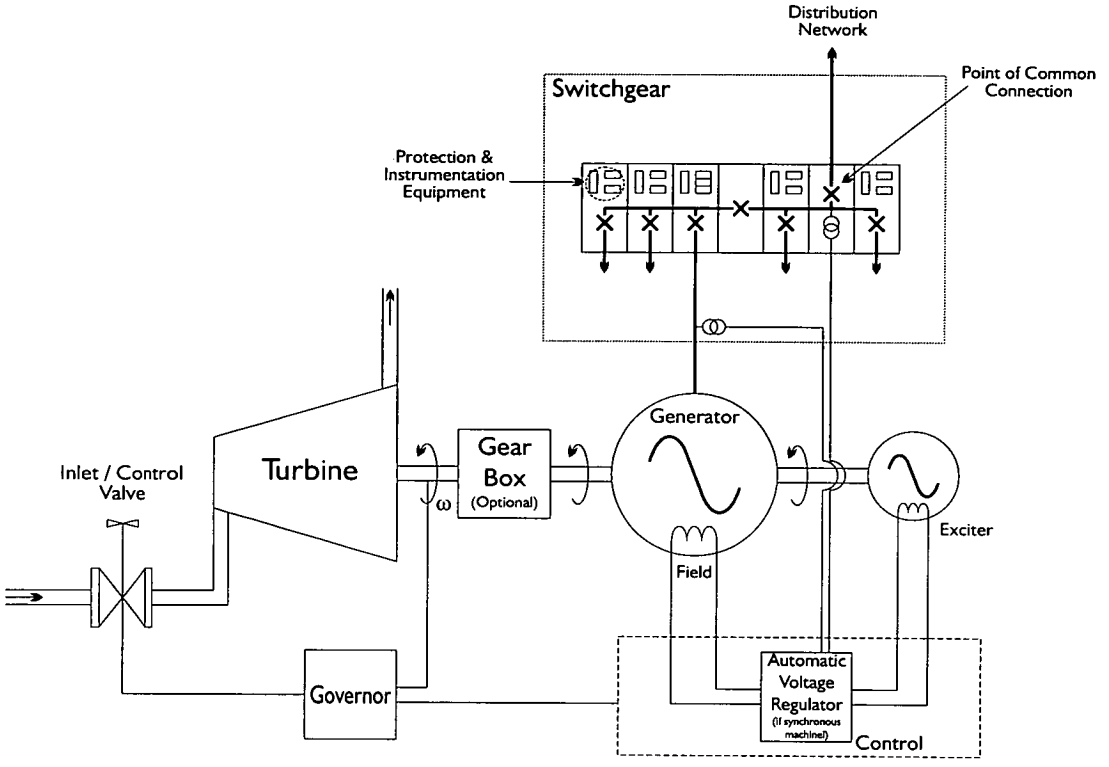


Figure 2.1: A block diagram of a generation station.

commercial bias, no regulatory controls were required since embedded generators were only allowed to operate in parallel with the grid supply and not allowed to export electricity [226]. The expansion of distribution networks during the same period developed on the basis that power flowed from the grid supply points (GSP) into lower voltage networks.

It was not until the 1983 Energy Act that this situation changed by allowing the commercial generation and export of electricity across public networks. The technical difficulties of exporting power from independent generators across distribution networks were highlighted, resulting in the formation of the Electricity Association, a representative body for the industry. This saw the introduction of the *Engineering Recommendation G59* [15] which stated the basic minimum safety requirements that embedded generators are required to meet. More specifically, this recommendation, henceforth referred to as G59,[1] describes the protection equipment, earthing arrangements and switching required for safe operation of any embedded generator.

Engineering Recommendations produced by the Electricity Association are reg-

[1]Since its introduction in 1985, G59 has been reviewed once, in 1991, and had two minor amendments subsequently added, in 1992 and 1995. As a result of the 1991 review, G59 is officially called *G59/1* but since many authors refer to this document as G59 and for the sake of brevity, the current version of the document, G59/1, with both amendments will be referred to as *G59* throughout this thesis.

ulatory in nature; that is, they are produced to develop a consistent approach to matters of safety across the electricity supply industry. Although not statutory, the Engineering Recommendations are required by all public electricity suppliers to be adhered to by third parties wishing to connect to their networks. However, due to the wide range of equipment and variable operational practices that occur in the industry, no single document could discuss all possible situations without excessive specification, therefore such documentation is open to interpretation.

To assist embedded generation developers, the Electricity Association produced the *Engineering Technical Report 113* [17], frequently referred to as ETR113, in 1988 to provide greater technical guide-lines for the implementation of G59. In 1995, ETR113 was reviewed to precipitate the better understanding and simplification of the interpretation and implementation for embedded generation developers. This revision also reflected the changes detailed by the review of G59 in 1991.

The scope of G59, and ETR113, was intended for generators connected at or below 20kV and where the output of the generator does not exceed 5MW. However, the United Kingdom government has actively supported through the *Renewables Obligations*[2] embedded generation on an economic basis. This policy has encouraged larger embedded schemes to be developed, and in 1996 the Electricity Association introduced *Engineering Recommendation G75* [16] for generators between 5MW and 100MW in capacity, connected above 20kV. Due to the large capacity of such embedded generators and their impact upon the distribution and transmission networks, they may require a licence to generate (dependent upon local circumstances) and thus be centrally dispatched by the National Grid Company.

Although G75 includes guidance on issues that affect larger embedded generators such as security, stability and synchronising, the document indicates that G59 and ETR113 should be used as sources of guidance on the technical aspects of connecting to the public electricity network. Such large schemes are, however, not relevant to this thesis as they are of sufficient size to afford to be professionally designed and engineered. However, G59 is applicable to small scale embedded generation, the economics of which are marginal. Thus a software application which can alleviate costs, especially during the initial stages of an embedded generation project, will be of great benefit. Therefore, this thesis will focus upon modelling the technical aspects associated with G59.

Another important issue associated with all types of generation is power quality. This issue is particularly relevant to embedded generators since they can affect local consumers. There are two important areas of concern: transient network voltage disturbance, frequently referred to as *flicker* due to its effects upon lighting loads, and *harmonic distortion* of the mains voltage waveform. Typical causes

[2] As previously described in Section 1.1, page 1.

of voltage flicker include the connection and disconnection of generators or transient torques produced from the prime mover, an effect specifically associated with wind turbines [56]. Harmonic distortion of the mains waveform may generally be introduced by power electronic devices or interfaces which are becoming increasingly common in embedded generation schemes; fuel cells and photovoltaics are examples of renewable energy sources that utilise power electronic devices directly [131], though wind farm developers are selecting static VAr compensators (SVC's) to improve voltage regulation as an economic alternative to distribution network reinforcement [236, 235].

Due to the safety implications that voltage flicker and harmonic distortion introduce into power systems, both issues are covered by Electricity Association guidelines; namely P28 [14] and G5 [13] respectively. Techniques already exist for the prediction and reduction of harmonic distortion, and the implications of a scheme in terms of voltage flicker may be obtained by the analysis of existing schemes or detailed numerical computer simulations. The inclusion of such analytical techniques is beyond the scope of this thesis but could be included in future development as a basis upon which design decisions may be founded upon. However, it should be noted that Jenkins and Strbac indicate that embedded generation may have the ability to improve power quality in certain circumstances [132]. Furthermore, Hodgkinson suggests that the impact of embedded generation upon the quality of the local electricity supply may be minimised. This may be achieved by avoiding the connection of embedded generation at low voltages[3] where fault levels are at their minimum [117].

Other legal and regulatory documents applicable to embedded generators include the *Grid Code* and the *Distribution Code*. Both these codes of practice were introduced by the 1989 Electricity Act, as a result of privatisation of the United Kingdom electricity supply industry, to define working relationships between public electricity suppliers and third parties, such as embedded generation developers [226]. Due to regional variations, in both demographic and geographical terms, which in turn reflect variations in the type and nature of local electricity networks, both the Grid and Distribution Codes have minor regional differences. These variations are also reflected in public electricity suppliers' interpretation of G59, but such differences are relatively minor and thus have a minimal impact upon this project.

In summary, the principal regulatory document that directly affects the design of switchgear, protection and instrumentation equipment for embedded generation

[3]The terms 'low voltage' and 'high voltage' have no universal definition within the electricity supply industry. For the purposes of this thesis, the definitions stated by G59 [15] will be adopted, as defined as follows. *Low voltage (LV):* A voltage normally exceeding extra-low but not exceeding 1000VAC or 1500VDC between conductors or 600VAC or 900VDC between conductors and earth. *High voltage (HV):* A voltage exceeding 1000VAC or 1500VDC between conductors or 600VAC or 900VDC between conductors and earth.

is G59, combined with the auxiliary material presented in ETR113. Therefore a key role of the design software must be to ensure G59 compliance is achieved by the provision of all necessary protection equipment.

2.1.2 Typical Equipment

Although all embedded generation schemes require both switchgear and protection equipment, both sets of equipment fulfilling very different purposes, they are both housed in the same metal enclosures, that are referred to as panels or *sections*. Each section contains a circuit breaker, a set of busbars, instrumentation and protection equipment; the collective term for one or more sections being *switchgear* or a *switchboard*. A typical switchgear installation is depicted in Figure 2.2. The components that constitute a section are schematically illustrated for two manufacturers in Figures 2.3 and 2.4.

The circuit breakers depicted in Figures 2.2 and 2.3 are described as *vertically isolating*, with the property that circuit breakers may be changed (if faulty) or serviced without isolating or de-energising the entire switchboard. Manufacturers also produce horizontally isolating, horizontally withdrawable circuit breakers, as shown in Figure 2.4, and fixed circuit breakers which must be de-energised for maintenance; they are, however, less expensive to purchase.

Basic switchgear assemblies are manufactured to be general purpose pieces of switching equipment, thus their application to embedded generation schemes, or to any other task, requires such general specifications to be chosen by the developer, detailing control, protection and instrumentation equipment. The manufacturers of switchgear supply basic sections complete with circuit breakers and busbars as these items are heavily integrated into the section enclosure. Additional components, such as current and voltage transformers along with control, instrumentation and protection equipment, must be installed into each section to the purchasers specification; either by the manufacturer or a specialised third party company. These additional components, with the exception of the current and voltage transformers, may be supplied by any suitable equipment manufacturer. The arrangement of components with a typical section is indicated by Figures 2.3 and 2.4.

2.1.3 Section Types

As previously mentioned, switchboard sections are intended for a variety of purposes, and embedded generators are frequently located on sites which require the generator switchgear to be integrated with other switching as part of a larger switchboard. An example of such a situation would be a combined heat and power

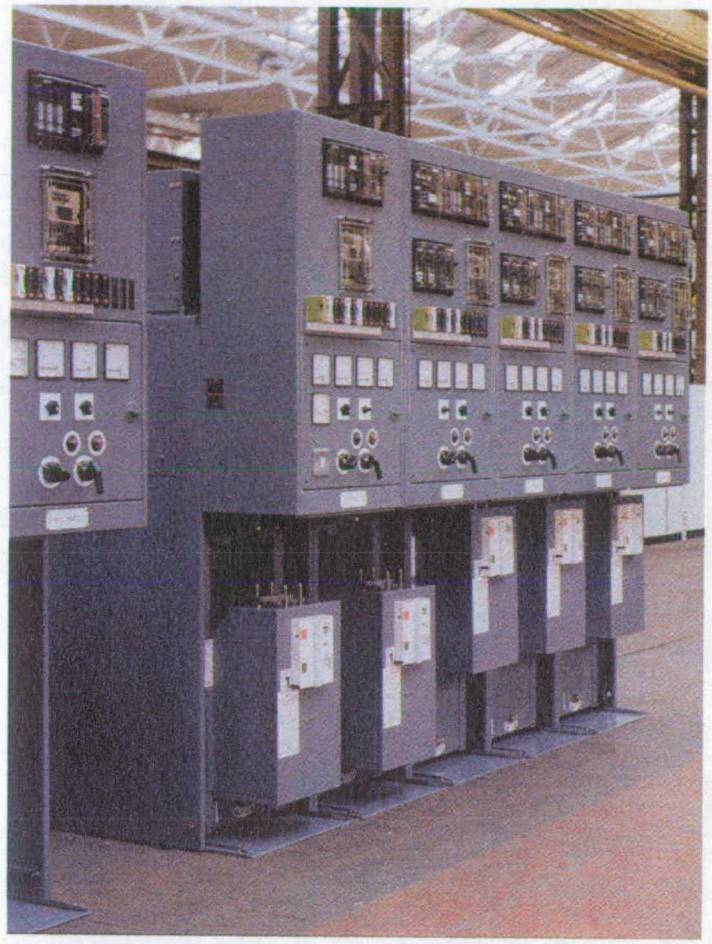


Figure 2.2: A typical switchgear installation [150]. Note the individual electromechanical protection devices mounted at the top of the front panel (two rows), below which a series of low voltage fuses, instrumentation meters, control switches and indicator lamps are mounted. The circuit breakers are mounted at the bottom of each section; two of which can be seen withdrawn from service.

scheme located within a paper mill; the generator, when operational, would supply electricity to reduce the grid demand of the on-site industrial machinery with any excess power being exported to the distribution network. In such a situation the switchgear sections illustrated in Figure 2.5 would be typical.

After consideration of the spectrum of embedded generation installations and manufacturers data, it becomes clear that there are six distinct types of switchgear section, as illustrated in Figure 2.5. These are listed below with an explanation of their respective function:

Incomer Section: The purpose of this section is to switch the incoming grid connection. Typically there is only one grid connection to a switchboard, although double and occasionally triple connections, separated by one or more *bus sections* (see below), are not uncommon *(section 4)*.

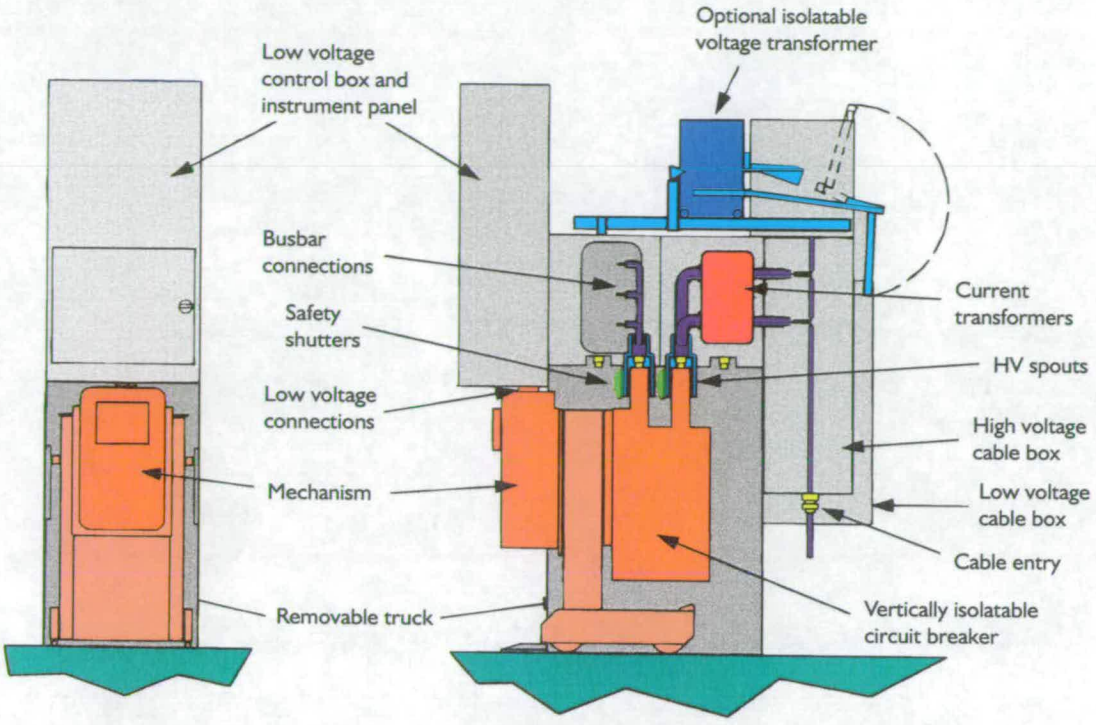


Figure 2.3: An illustration of the front and side views of a vertically isolating, horizontally withdrawable switchgear section, indicating the arrangement of construction and components [150].

Load Section: These sections switch any loads connected to the switchboard; in a typical industrial setting, a load section would correspond to a production area *(sections 1,3,6 and 7)*.

Generator Section: This section switches a single generator. If there are multiple generators connected to a single switchgear set, then each generator must be connected via its own section *(section 8)*.

Bus Section: The purpose of a bus section is to allow groups of sections to be switched or to separate sources of energy; *e.g.* grid connections and generators *(section 5)*.

Operationally Spare Section: To allow for future expansion or a rapid connection of a load, the developer may include a fully specified load section which is not connected during the installation, commissioning or initial operation of the switchgear *(section 9)*.

Spare Section: These sections contain only a set of live busbars – no other equipment. Developers may include such sections to allow for future expansion of the switchgear *(section 2)*.

The function of a section determines the constituent equipment required to fulfil its purpose. This equipment must be selected and specified by the switchgear de-

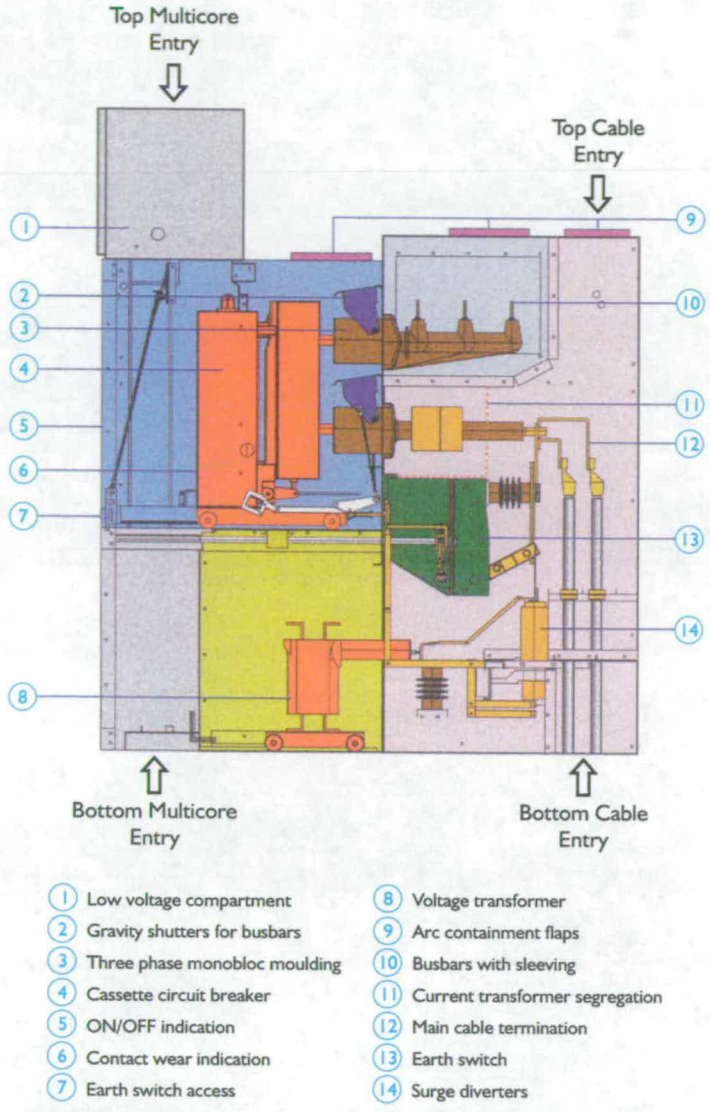


Figure 2.4: An illustration depicting the construction of a horizontally isolating, horizontally withdrawable switchgear section, from the side point of view [152].

veloper. Within each section, with the exception of spare sections, the following categories of equipment are required:

- Circuit breakers,
- Protection equipment,
- Instrumentation equipment,
- Current and voltage transformers, and
- Fuses and links.

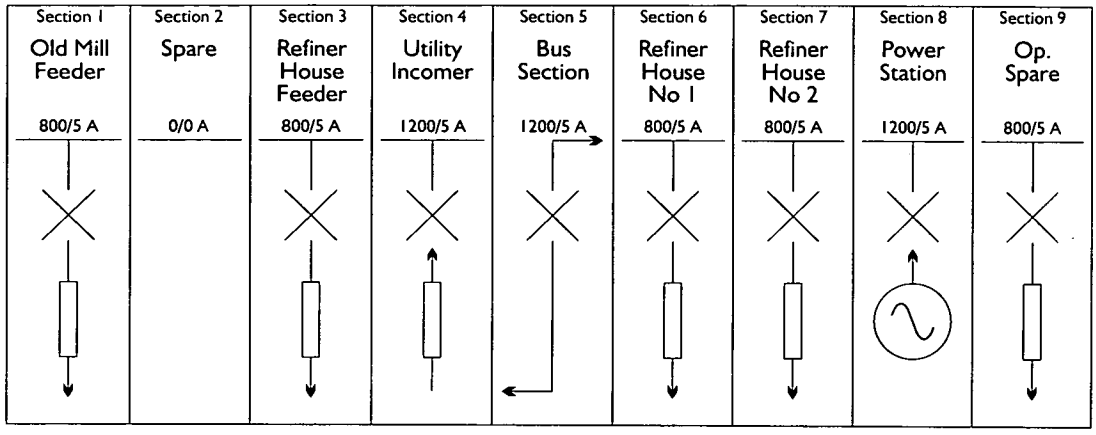


Figure 2.5: A series of switchgear sections installed at a paper mill [166].

By knowing the intended application of a section, the switchgear developer may then select instrumentation and protection equipment to meet the sections intended purpose. This act in turn determines the current and voltage transformer requirements and specifications, finally leading to fuse specifications.

The above list of equipment categories will be examined in more depth in the following sections, along with the specification of auxiliary power sources. These sources are required to ensure that instrumentation and protection equipment operate correctly.

2.1.3.1 Circuit Breakers

The most important functional component of any switchgear section is the circuit breaker. The purpose of this device is to perform the switching of electrical circuits during normal as well as abnormal operating conditions. Circuit breakers connect or disconnect circuits by the mechanical compression or separation, respectively, of metallic contacts. Typically, circuit breakers are manufactured to switch loads between 200A and 3000A at voltages between 3.3kV and 36kV. Higher capacity circuit breakers (<5000A) can be manufactured but are only utilised for specialist purposes due to the technical difficulties relating to their construction.

All circuit breakers manufactured within the United Kingdom must conform to British Standard 5311 [39] which describes, in detail, all aspects of circuit breaker rating, design, installation and testing. Within British Standard 5311, twenty four electrical quantities of a particular type of circuit breaker must be specified by the manufacturer. However, only the circuit breakers rated voltage, current, frequency, fault level and the auxiliary supply's voltage and frequency ratings directly affect or have a bearing upon other aspects of the overall electrical design of a switchgear installation.

As part of their specification, all circuit breakers must be able to operate, that is disconnect circuits, during fault conditions. The maximum fault that a circuit breaker can withstand is indicated by the device's fault rating. A particular switchboard section is manufactured to accommodate a specific type of circuit breaker. As a result the fault rating of a circuit breaker will generally dictate the fault rating for a particular type of switchboard section. Other components within a switchboard that are required to be fault rated will be constructed to meet or exceed this fault rating. Such components include busbars and any device connected to them without the inclusion of a fault limiting device, such as a busbar or circuit fuse [216].

Having considered the electrical ratings of a circuit breaker, the *interrupting medium* is the next most important factor in the specification of these devices. During switching, a transitory stage of arcing occurs between the electrical contacts. Such arcing allows the flow of electrical energy to continue until the discharge ceases. The discharge also results in the surface of the switching contacts being damaged. The purpose of the interrupting medium is to accelerate the extinguishing of the discharge. During the initial development of circuit breakers, air was the only interrupting medium available. The limited extinguishing properties of air circuit breakers resulted in the development of other interrupting mediums; these included compressed air, mineral oil, sulphur hexafluoride (SF_6) and most recently, ultra high vacuum [8]. Historically, mineral oil circuit breakers have been popular, but due to the safety issues associated with these devices [51, 21], modern circuit breakers utilise either sulphur hexafluoride or vacuum as their interrupting medium as such devices are physically compact, have lower foundation requirements, eliminate the bulk handling of mineral oil, and are extremely reliable [85]. The interrupting medium throughout an individual switchboard installation is always the same.

In order for a circuit breaker to open and close, the device's operating mechanism must be physically energised. There are various means by which this is achieved, such as [85]:

- Manually,
- Direct current solenoids,
- Compressed air,
- High pressure oil,
- Charging spring, and
- Electric motors.

The universal practice for the operation of indoor metal clad switchgear is the use of a charged spring mounted as an integral part of the circuit breaker's switching mechanism which is mechanically compressed during the closing operation, with the moving contacts being latched in the closed position. Modern switchboards are intended for automatic or remote control, therefore favour the use of solenoids or electric motors to energise the switching mechanism. The electric power to energise either of these devices is obtained from the switchboard's auxiliary supply. For circuit breakers that are operated infrequently, a manual charging mechanism can be installed [54]. In addition to the operating mechanism, circuit breakers also require two solenoids to cause them to open, a *trip*, and to close after the close springs have been compressed. These two solenoids are named the *shunt trip coil* and *spring release coil* respectively. Both of which require connection to the switchboard's tripping supply at either 110 or 240 Vdc [160].

2.1.3.2 Protection Equipment

Protection equipment, also termed protective gear or protective relaying, has the duty initiating the isolation of any component of an electrical system in which an abnormal condition occurs, either instantaneously or, in some cases, after a predetermined delay. The abnormal conditions in which protection equipment may be required are summarised as follows [160, 275]:

1. The overloading of equipment, which if persistent, results in the overheating of equipment, such as transformers or rotating machinery windings, cables or damage to other equipment. Such faults are called *overcurrents*.

2. The failure of insulation resulting in a hazardous leakage of current to earth, either a single phase to earth, phase to phase to earth or three phase to earth fault; such faults are collectively referred to as an *earth fault*.

3. The failure of insulation resulting in a short circuit between two or three phases; called *phase to phase* or *three phase fault* respectively.

4. The complete loss of, or sustained drop in, system voltage resulting in the malfunction of equipment, *e.g.* electronic equipment 'crashing' or electric motors slowing or stopping; such situations are referred to as *loss of mains*, or *undervoltage* conditions respectively.

5. The operation of the grid is unstable, or beyond statutorily operational limits; faults arising from such connections are referred to as *under-* or *over-frequency* and *overvoltage*.

Protection equipment for embedded generators was originally simply a set of electrical fuses [280]. However, the requirement for a more sophisticated approach to allow coordination and division of electrical networks into areas or *zones*, as depicted in Figure 2.6, precipitated the development of primitive protection devices [170] continuing to the present with the development of complex digital protection equipment.

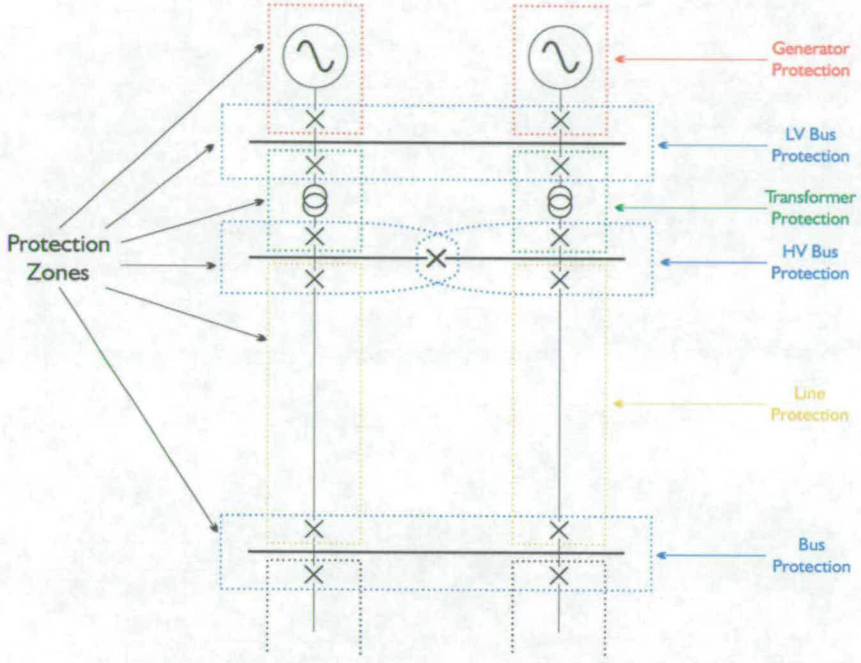


Figure 2.6: One line diagram of a power system indicating the primary protection zones [8, p.10(Adapted)].

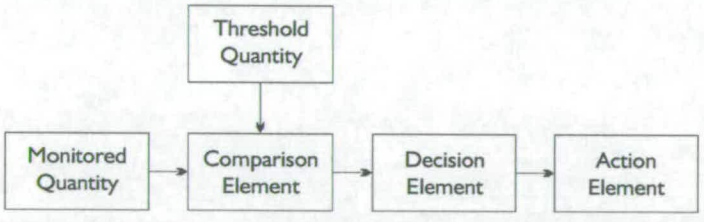


Figure 2.7: A schematic the functional elements constituting a protective device [8, p.7].

Protection devices require electrical connections to either *sensing transformers* or *transducers* to allow them to monitor a specific aspect of the electrical network or power station. Typically sensing transformers are utilised, which is a collective term for both *voltage transformers* and *current transformers*. These devices electrically isolate and reduce the magnitudes of their respectively measured quantities to manageable levels. However, other protection devices, especially those associated with motor or generator protection, require temperature, pressure, vibration and speed transducers. The purpose of all protection devices is to monitor a spe-

cific quantity, via a transducer, ensuring that this quantity remains within a predetermined range. Should the said quantity exceed the given range, the protection device will operate providing a contact change that opens the circuit breaker that is intended to clear the fault. This operation also initiates either shutdown of the generating station or an alarm signal to be issued. This process is schematically illustrated in Figure 2.7. A protection device may also be time delayed to determine if the monitored quantity has exceed the given range for a significant period of time; again such action is predetermined and defined by the protection engineer.

Protection equipment or individual devices are frequently referred to as protection *relays* as early development of such devices was based upon mechanical relay technology. Since their introduction electromechanical relay devices have been refined to become reliable and accurate pieces of protection equipment [191]. Due to the frequent and costly maintenance required to ensure that such electromechanical relay devices operated correctly, the first electronic protection devices were developed in the 1930's based upon vacuum tube technology. However, such electronic protection devices were not extensively used as they required similar or higher levels of maintenance compared to their electromechanical equivalents. The development of the transistor resolved this issue and by the mid 1960's electronic protection equipment had become extensively utilised in the power industry [8, p.99].

The evolution of protection equipment continued with the development of digital protection devices which was first proposed in 1969 by Rockefeller [222], but it was not until the advent of cost effective microprocessors that the development of digital devices was considered seriously by the power industry. Digital protection devices convert the transducer or input quantities presented to it into digital signals by the processes of sampling and quantisation. The resulting digital information may then be processed by the microprocessor, thereby allowing numerical and Boolean logic decisions to be performed under software control; typical operations include tripping, closing timing, reclosing, blocking, metering and synchronising [25, 191]. By allowing protection signals to be digitally represented within a single digital protection device, a significant range of protection functions may be implemented; such devices are termed *multi-function* or *integrated* protection relays and are becoming increasingly popular due to the range and flexibility of features offered [110].

The different areas of protection required by an embedded generator are depicted in Figure 2.8, indicating the three categories of protection equipment that together combine to form a complete protection scheme. The *generator protection* monitors for internal physical faults on the generating unit, such as overheating, bearing failure and excessive vibration, but may also include electrical quantities, such as overcurrent, circulating currents, earth faults and voltage/frequency operated protection.

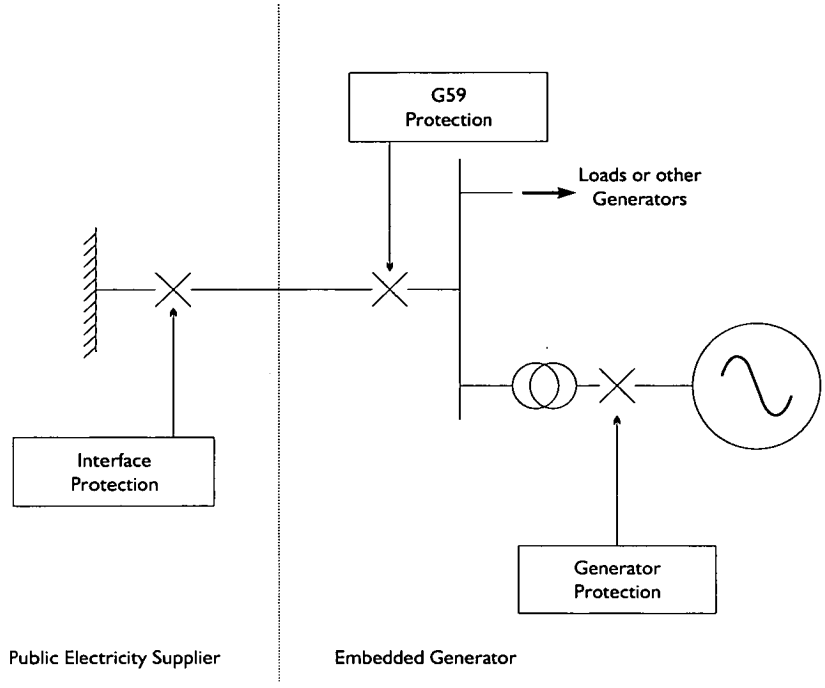


Figure 2.8: Embedded generator protection [131, 17, p.237].

The *G59 protection* ensures that the embedded generator does not disturb the operation of the public distribution network. Two situations in particular are monitored: that the generator does not continue to supply a fault on the network,[4] and that the generator does not continue to operate isolated from the distribution network, a situation termed *islanding*. The main hazards of islanding are [225, 200]:

1. The reduction of fault levels such that the performance of protection equipment is severely reduced or completely fails to operate,
2. Reduction in the quality of supply due to the reduction of fault levels,
3. Manual, remote or automatic operation of public electricity suppliers' circuit breakers resulting in the re-connection of unsynchronised generation,
4. The operation of generation without a system earth, and
5. The energisation of the high voltage supply network which may or may not be healthy – a dangerous situation called *backfeeding*.

All of these issues substantially reduce or completely nullify safety precautions resulting in possible damage of the embedded generator or, more importantly, expose personnel, from either the utility and/or the operator, to partially energised, unearthed equipment.

[4]This is a situation that can easily occur due to most embedded generators not having sufficient fault capacity to operate distribution protection and trip network circuit breakers.

Finally, the *interface protection* acts as a backup to the G59 protection, ensuring that faults are cleared and that the generator will not continue to supply a faulted or de-energised network. However, this protection equipment is owned and operated by the public electricity supplier, or otherwise referred to as the local *utility*.

The associated costs with a complete and comprehensive protection scheme are considerable. Since there is a certain degree of duplication between G59 protection and the interface protection, it may be possible to come to an agreement with the local public electricity supplier on a more economic design. Such an agreement, if at all possible, is dependent upon the size of the proposed scheme and the conditions prevailing on the local distribution network [130]. However, developers are keen to reach an agreement since the cost of interface protection has a major impact on the viability of a scheme, especially as the electrical capacity of an embedded project decreases [63].

As previously mentioned, G59 prescribes the protection equipment required for an embedded generation scheme, in terms of electrical quantities to be monitored; Table 2.1 lists these protection requirements. The first four protection devices indicated in Table 2.1 are present to ensure the fundamental safety of the embedded generator during operation and the last device in the list, the *parallel limit timer*, ensures that a plant certified for only short term operation in parallel with an energised distribution network disconnects once a short 'test' period has been completed.[5] Such plant is intended to operate as standby generation.

The protection devices *reverse real and reactive power* and *directional overcurrent* are required to ensure that a generating plant does not *motor*, that is, to be synchronised and connected to the grid but instead of producing power, the generator draws power from the grid. The final two protection devices, namely *loss of mains* and *neutral voltage displacement* are employed to ensure that the generator does not become islanded, or operate with the neutral other than at earth potential.[6] These two conditions merit further consideration as they both require the installation of expensive protection equipment.

Under G59 all generators larger than 150kVA, *i.e.* those that export electricity to the distribution network, have to employ a loss of mains protection device. The function of this device is to ensure that if the generator is operating and becomes

[5]This classification of embedded generation is intended to back-up the public electricity supply. Such generators are installed in commercial or industrial premises where a sustained loss of mains power would incur significant costs to the company concerned. Installations typically utilise diesel engines as prime movers and to ensure the plant operates correctly they are tested frequently (normally at least once month).

[6]It should be noted that reverse power protection may be included within the interface protection to provide complementary protection to loss of mains situations [117], however, due to it's associated cost and difficult implementation, such a decision on its provision may only be decided by the developer. G59 and ETR113 provide no guidance on the assessment of the necessity for such a protection device.

	Permanent Parallel Operation				
	HV Generating Plant		LV Generating Plant		
Protection Required	No Export	Export	Small (<150kVA)	Medium (150 - 250kVA)	Large (>250kVA)
Under / Over Voltage	✓	✓	✓	✓	✓
Under / Over Frequency	✓	✓	✓	✓	✓
Overcurrent	✓	✓	✓	✓	✓
Earth Fault	✓	✓	×	✓	✓
Loss of Mains	✓	✓	×	✓	✓
Reverse Real & Reactive Power	✓	‡	×	×	‡
Directional Overcurrent	‡	‡	×	×	‡
Neutral Voltage Displacement	✞	✞	×	✞	✞
Parallel Limit Timer	×	×	×	×	×

	Short Term (Test) Parallel Operation (<5 minutes)			
	HV Generating	LV Generating Plant		
Protection Required	Plant No Export	Small (<150kVA)	Medium (150 - 250kVA)	Large (>250kVA)
Under / Over Voltage	✓	✓	✓	✓
Under / Over Frequency	✓	✓	✓	✓
Overcurrent	✓	✓	✓	✓
Earth Fault	✓	×	✓	✓
Loss of Mains	×	×	×	×
Reverse Real & Reactive Power	×	×	×	×
Directional Overcurrent	×	×	×	×
Neutral Voltage Displacement	✞	✞	✞	✞
Parallel Limit Timer	✓	×	✓	✓

Key:
- ✓ Protection required by G59
- ✞ For application refer to ETR113
- ‡ Optional protection (for application refer to ETR113)
- × Not required

Table 2.1: The minimum protection requirements for generating plant operating in parallel with a Public Electricity Supplier network [17, 225].

islanded, it will trip the generator circuit breaker, disconnecting it from the distribution network and shut down the prime mover safely.

To guard against islanding, loss of mains protection devices operate on the principle that should the distribution network to which the embedded generator is connected become de-energised, then a substantial fluctuation in the voltage, frequency, or, proved by recent research work [200], impedance, will result at the point of common connection. Two types of loss of mains protection devices have become widely accepted by the power industry; they are referred to as *voltage vector shift* and *rate of change of frequency* (ROCOF). Both these devices are expensive to purchase and have been proved unable to readily distinguish between normal and abnormal system frequency variations in operation. This is especially the case for ROCOF based devices, as sudden changes in load or generation capacity in a local area can cause nuisance tripping [280, 117, 226]. The introduction of multifunction digital protection devices has resulted in manufacturers [110] producing complete

G59 protection relay packages in one unit. However, such devices have yet to be generally accepted by the industry due to their unproven track record and high cost. Other research work has been conducted for loss of mains protection devices for specialist energy sources; for example Crossley *et al* [61] have developed a loss of mains device for the protection of wind farms.

Embedded generators with a capacity of 1000kVA or below will typically employ a low voltage generator, connected via the distribution network by a star/delta transformer, illustrated in Figure 2.9(a). If the generator backfeeds via such a connection arrangement, then any earth fault on the faulted supply network will not produce an earth fault current but will unbalance phase voltages, resulting in placing extra stress onto the unfaulted phases. This situation completely compromises the safety of both generator and distribution networks.

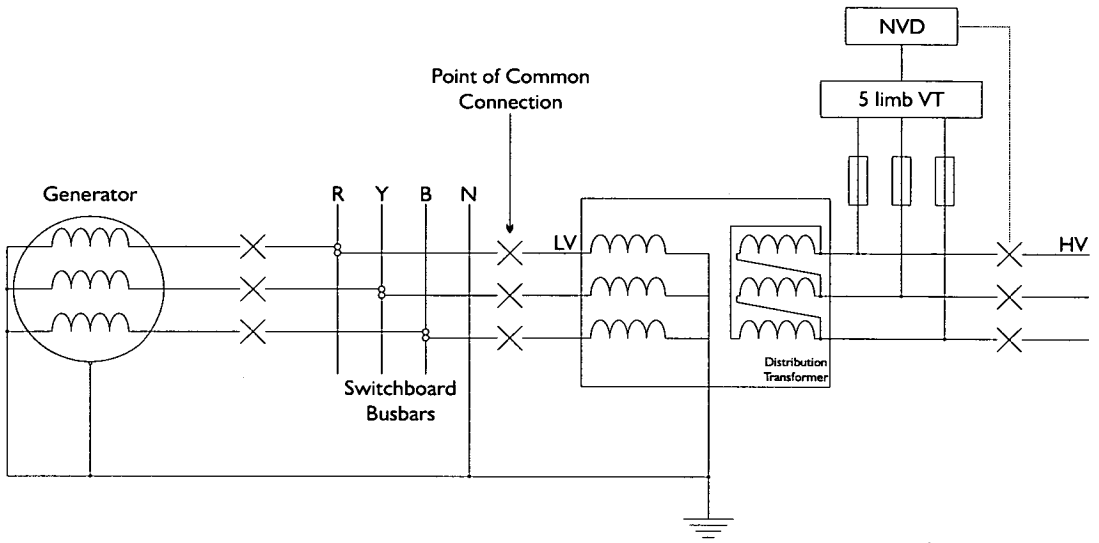
A similar situation may occur with high voltage generators ($\gtrapprox$1500kVA) which normally obtain their earth connection via the high voltage distribution network, depicted in Figure 2.9(b). Thus sustained generation during a loss of supply fault results in the unearthed operation of the generator.

Both the aforementioned situations may be detected by *neutral voltage displacement* protection, connected via a *five-limbed voltage transformer*. This transformer is connected either to the high voltage side of the distribution transformer, in the case of low voltage generators represented in Figure 2.9(a), or directly to the incoming supply for high voltage generators, portrayed in Figure 2.9(b). In both instances the neutral voltage displacement device and five-limb transformer must be connected to the utility's distribution network, beyond the developer's *point of common connection* – the boundary between the developer's or customer's electrical network and the distribution network. If deemed necessary, this equipment has to be purchased and installed at the expense of the developer but with ownership transferred to the utility.

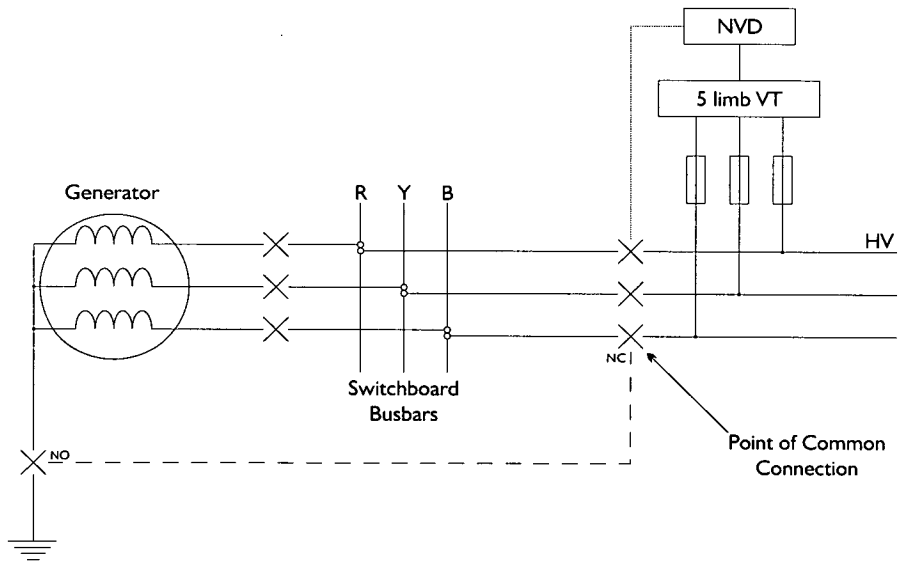
To conclude, G59 requires some form of loss of mains protection to be included within the protection scheme for an exporting embedded generator with a capacity greater than 150kVA.

2.1.3.3 Instrumentation Equipment

The purpose of switchgear instrumentation equipment is to provide metering and a visual indication, in real time, of various operations or electrical quantities concerning the status of the section to which it is fitted. Instrumentation equipment is not required to enable a switchgear installation to operate safely; such important functions are performed by either control or protection equipment. Therefore in-



(a)



(b)

Figure 2.9: Typical connections for embedded generators [17, (Adapted)]; (a) Low voltage connection, (b) High voltage connection.

strumentation devices, if any, fitted to a section are at the discretion of the design engineer.

The number and type of instruments fitted to a particular section depends upon the section's function; an indication of typical instrumentation facades and quantities measured are depicted in Figure 2.10. As under and over current situations present the most danger, almost all sections of any switchgear installation will be fitted with at least a single phase current meter; the exception being spare sections. However, in situations were load imbalance is common and hence an important quantity to be monitored, either three individual current meters, or a single meter with a phase selector switch, may be fitted.

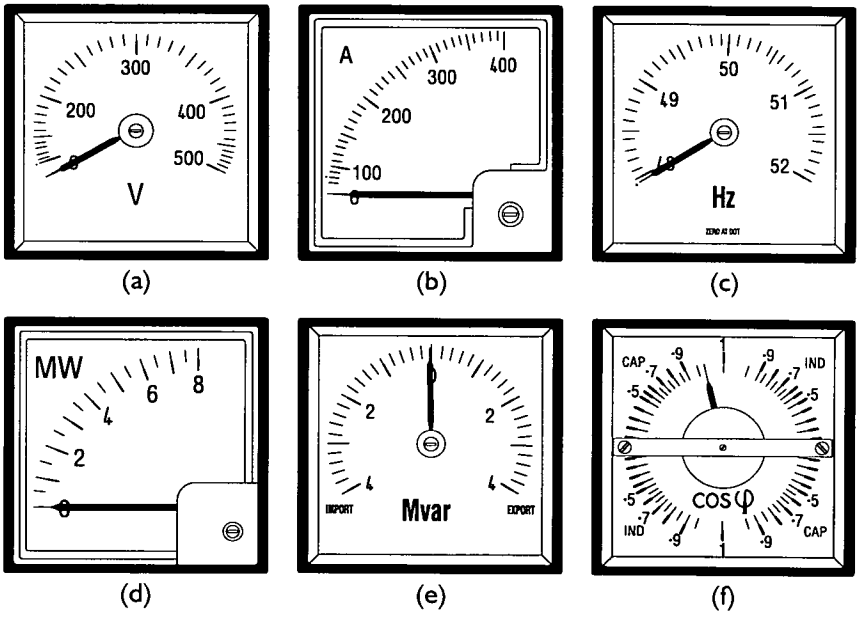


Figure 2.10: A selection of typical instrumentation facades [151]; (a) 240°volt meter, (b) 90°ampere meter, (c) 240°mains frequency meter, (d) 90°real power meter, (e) 240°reactive power meter, (f) 360°power factor meter.

Due to the safety implications introduced by the switching of any source of electrical energy, incomer and generator sections frequently require additional instrumentation. The instrumentation included on incomer sections also frequently includes volt meters though generator sections additionally include power factor, active and reactive power and synchronisation indicators as well. The connection of instruments that require mains derived current or voltage signals must be connected to measurement grade current or voltage transformers.

Instrumentation equipment also includes indicator lamps which are used to signal important events or status, such as if the section circuit breaker is either open or closed or if the section busbars are live. Other non-electrical but important events, for example the indication of the position of the main inlet valve for a hydro

generator or the boiler temperature of a steam turbine installation, may also be included within a section's instrumentation but this is a non-standard procedure – such equipment is normally mounted in separate, purpose made steel cabinets.

2.1.3.4 Current and Voltage Transformers

A transformer is a device that changes an alternating current at one potential difference to another through the action of a magnetic field linking the two coils of a transformer together [48]. This property to transform voltages and currents, determined by the turns ratio between the two coils of a transformer, enables the large voltages and currents associated with power systems to be linearly reduced to manageable magnitudes, thereby allowing instrumentation and protection equipment to be mounted on low voltage panels within the switchgear. The benefits of operating equipment at low voltages also allows the physical size of the equipment to be reduced and personnel to calibrate and test such equipment safely [8]. The reduction of electrical quantities for measurement and protection purposes is accomplished by two categories of device, namely *current transformers* (CT's) and *voltage transformers* (VT's). Both these categories of transformer are collectively referred to as *sensing transformers*.

Current and voltage transformers are further subdivided into two categories relating to their construction and rating; thus termed either *protection grade* or *instrumentation grade* transformers. All current and voltage transformers must be constructed to British Standards 7626 [38] and 7625 [37] respectively.

Protection grade transformers are used to derive the measurement of currents or voltages from the busbars, or circuit being protected. Such transformers must be fully fault rated and are constructed so as not to saturate during fault conditions until a multiple of their rating has been exceeded; by which stage the protection device should have operated. If saturation did occur, a substantially reduced magnitude representation of the current or voltage measured would be observed by the protection equipment, resulting in the protection device not sending a trip signal. There are two classes of protection grade transformers for both current and voltage measurement, defined by British Standards 7626 and 7625; either 5P or 10P for current transformers and 3P or 6P for voltage transformers. The numerical quantity indicates the multiple of the rating of the transformer within which the device remains accurate during a fault. The British Standard 7626 also allows for an additional classification of protection grade current transformer, called *Class X*. Such devices are utilised for more demanding applications where the device is specified not by an accuracy class, but by detailed specification.[7]

[7] A Class X specification would typically include knee point voltage, turns ratio, exciting current at the stated voltage and secondary resistance.

Measurement grade transformers are used to derive electrical quantities from busbars in the same manner as protection transformers. Indeed the construction of measurement grade transformers is similar to their protection grade cousins, although they differ by being accurate up to only 120% of their rating. Beyond 120% measurement grade transformers saturate, thereby protecting the measurement instruments connected to them from over current and over voltage conditions; such conditions are only experienced during fault conditions. The accuracy classes and relative percentage errors for measurement grade current and voltage transformers are given in Tables 2.2 and 2.3 respectively.

Accuracy Class	*± percentage of current ratio error of rated current*				
	5	*20*	*50*	*100*	*120*
0.1	0.4	0.2	ns	0.1	0.1
0.2	0.75	0.35	ns	0.2	0.2
0.5	1.5	0.75	ns	0.5	0.5
1.0	3.0	1.5	ns	1.0	1.0
3.0	ns	ns	3.0	ns	3.0
5.0	ns	ns	5.0	ns	5.0

Table 2.2: Measurement grade current transformer classifications and their respective percentage accuracy at various burdens, in accordance with British Standard 7626 [38]. Note that *ns* indicates *not specified*.

Accuracy Class	*± percentage voltage ratio error*
0.1	0.1
0.2	0.2
0.5	0.5
1.0	1.0
3.0	3.0

Table 2.3: Accuracy classes for measurement grade voltage transformers in accordance with British Standard 7625 [37]. Note that the above values are only correct for burdens between 25% and 100% of rated output current and between 80% and 120% of rated output voltage.

For both measurement grade current and voltage transformers it is typical that accuracy classes 0.5 or better are used for metering purposes, classes 1.0 or 0.5 for protection purposes and class 3.0 is normally used for indication purposes.

2.1.3.5 Fuses

The simplest and oldest device for the protection of any piece of electrical equipment is the fuse. A fuse is an "overcurrent protective device with a circuit opening fusible part that is heated and severed by the passage of overcurrent through it [129,

p.380]." For the purposes of this project the term 'fuse' describes the total assembly of the fuse element sealed in a cartridge and mounted in a holder in a circuit. The advantages and limitations of fuses are summarised as follows [272]:

Advantages:

1. Low cost,
2. Fast clearance of faults,
3. High reliability,
4. Constant performance, and
5. Fail safe operation.

Limitations:

1. Cartridge requires replacement after operation,
2. Difficulty in protecting against small overcurrents,
3. Materials and logistics limit use at high voltage,
4. Indication of fuse status is not trivial, and
5. Not suitable for sensitive applications.

Fuses are extensively utilised in the power industry for the protection of lines and cables up to 132kV. In a similar manner that protection relays are organised to provide protection zones, fuses may also be zoned by the use of *coordination charts*. The use of fuses at higher voltages is limited to specialist applications due to the difficulty and cost of manufacturing cartridges, although fuses can safely interrupt currents up to approximately 200kA [11].

Within the switchgear installations, fuses are necessary to protect instrumentation and protection devices, and voltage transformers[8] from overcurrents. Therefore all the fuses required for the protection of either instrumentation or protection equipment for a section are mounted on its corresponding front panel or inside. Voltage transformer primary fuses are mounted within the high voltage compartment within the same section but the low voltage secondary fuses may be mounted either inside the low voltage compartment of the section or on the front panel of the section.

[8]Current transformers are never fused since the resulting open circuit from the operation of a fuse will cause the current transformer to become saturated. This results in the mains alternating current waveform being chopped by the saturation effect, and coupled with the winding inductance, produces very high voltage spikes that can damage the current transformer windings and be a danger to personnel.

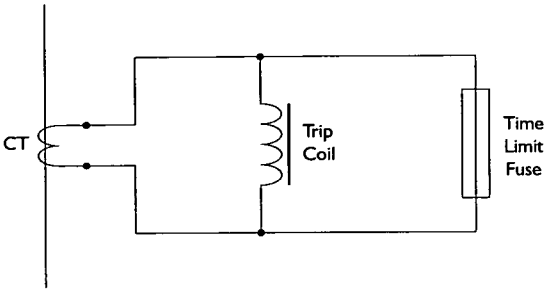


Figure 2.11: A current transformer operated, direct-acting trip coil, with time limit fuse [160, (Adapted)p.445].

2.1.3.6 Auxiliary Power Supplies

The first generation of electromechanical control, instrumentation and protection devices obtained any additional energy required for their operation either directly or indirectly from the generator or grid connection which they served. An example of such an arrangement is illustrated in Figure 2.11, where a current transformer connects directly to the trip coil of a circuit breaker. To introduce a time delay, a time limit fuse is connected across the trip coil terminals with an inverse time/current characteristic. During normal operating conditions the fuse will carry the current transformer secondary current, by-passing the trip coil. During fault conditions, however, sufficient current will cause the fuse to 'blow' and the whole secondary current will be transferred to the trip coil, resulting in its operation.

The operation of protection equipment based upon the principle illustrated in Figure 2.11, or other similar arrangements, was dependent upon the consistent and accurate manufacture of fuses. The variability of manufacturing processes that existed at the end of the nineteenth century, combined with the limited functional nature of such schemes and the requirement to replace a fuse after each operation, motivated the development of electromechanical relay devices. The adoption of relay based protection devices substantially improved the accuracy, both in terms of current and time measurement; such a device is schematically illustrated in Figure 2.12. However, the use of such devices and their interconnection to form protection schemes necessitated the use of separate tripping facilities at the circuit breaker, with the energy required for tripping obtained from a separate, reliable source.

More recently the advent and adoption of modern electronic control, for both instrumentation and protection purposes, has resulted in the requirement for an independent electrical energy source to power this equipment. Therefore in modern switchgear installations there are two separate electrical power sources, independent of the main grid connection upon which the switchgear operates; a non-secure supply known simply as the *auxiliary supply* and a secure supply referred to as the

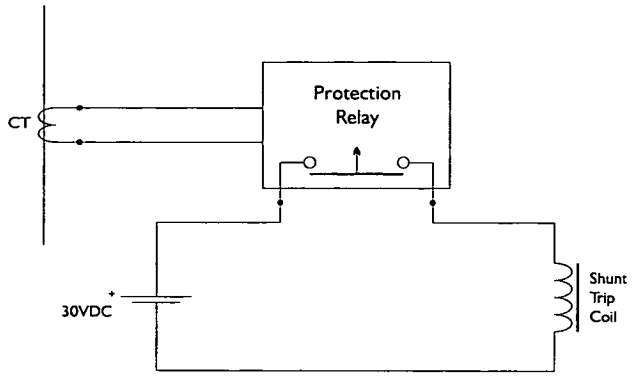


Figure 2.12: A relay operated protection device, with separate circuit breaker tripping facilities.

tripping supply. The main items of equipment that require auxiliary power supplies and their typical voltage requirements are listed in Table 2.4.

Device	*Auxiliary Voltage*	
	AC	*DC*
Circuit breaker (motor/solenoid operated)	230, 400	
Circuit breaker (tripping circuit)	30,50,220	30,50,220
Instrumentation equipment	230	110
Protection equipment	230	24,48,110,125,250

Table 2.4: Switchgear devices and their respective auxiliary power supply voltage requirements [150, 152].

The auxiliary supply generally provides power to all non-essential switchgear devices; these are pieces of equipment that if de-energised or removed from a section would not compromise the safety or affect the operation of that section or adjoining sections. Devices connected to this supply would include instruments, panel illumination and anti-condensation heaters. Such supplies must be connected to another source of electrical power other than the generator or the public electricity supplier connection switched by the switchgear. Typically the auxiliary supply is a single phase connection to the powerhouse's[9] non-secure mains supply; that is, a grid connection without any form of additional security of supply precautions or emergency or backup supply installation. The tripping supply is also normally connected to the powerhouse's mains supply but via a rectifier with a bank of emergency batteries[10] or other energy storage medium. As the name suggests, the tripping supply provides electrical power to all the protection equipment (and critical sections of the control gear) vital to the switchgear and generator's safe operation during the event of a complete loss of mains power. During such an event, the embedded generator must be safely disconnected from the distribution

[9]The building that houses or contains the generator and associated switchgear.
[10]Such an installation is frequently called an uninterruptible power supply (UPS).

network, the prime mover (and therefore the generator) shut down and the switchgear circuit breakers left in a safe setting to await re-energisation. The schematic arrangement of a typical protection configuration is shown in Figure 2.13.

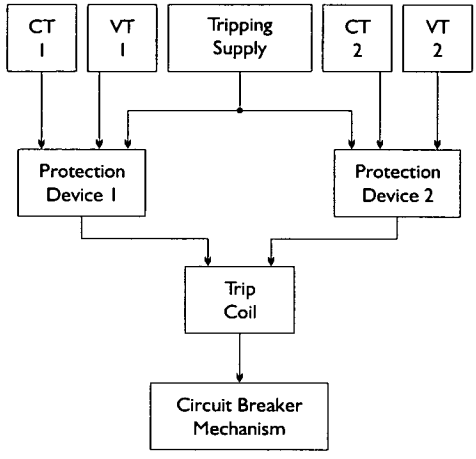


Figure 2.13: A schematic diagram of a typical protection equipment configuration.

Both auxiliary and tripping supplies, unless already installed, are considered as part of the switchgear specification and eventual installation. In certain cases, especially schemes located within industrial sites, both supplies will have already been installed forming part of an existing switchgear installation. In such cases only the additional loading placed on each supply by the new switchgear installation must be specified, assuming that both new and existing switchgear are designed to operate off similarly specified supplies.

2.1.4 Connection Topologies

It could be argued that the installation of an embedded generator, in electrical terms, is substantially different for every site considered. This is most certainly the case in terms of the actual electrical magnitudes involved and minor details, such as the size and location of trapped and local loads or earthing arrangements. But the overall topological features of embedded generation are consistent. This may be observed by considering a variety of frequently encountered connection topologies, as depicted in Figure 2.14.

Most embedded generation schemes will broadly fall into one of the topologies illustrated in Figure 2.14. All G59 compliant installations must apply the same protection philosophy; that is, the incoming, or public supply, connection is always protected from the embedded generator in the event of a fault. The application of G59 protection requires the fitting of protection devices to trip a circuit breaker,

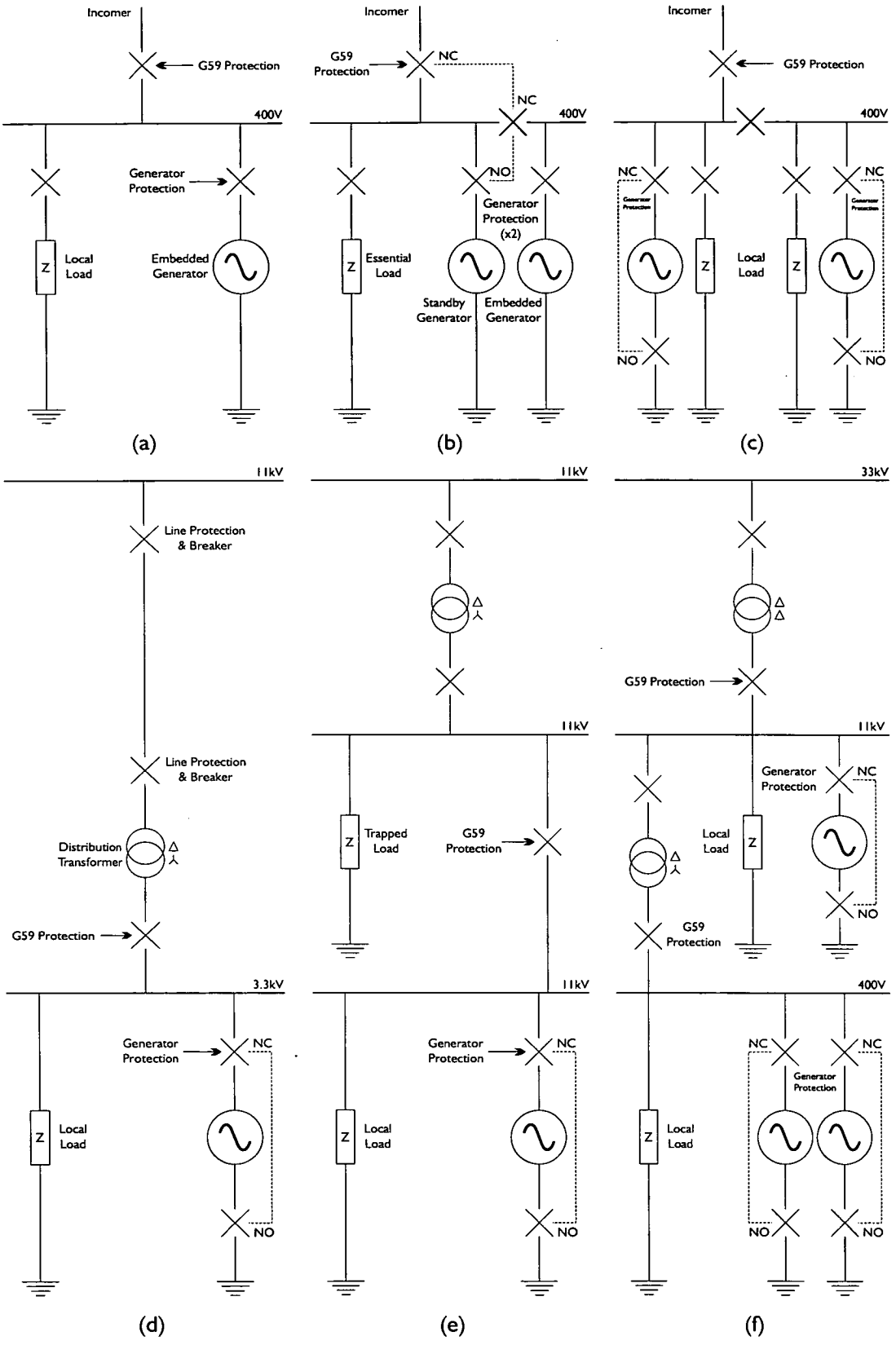


Figure 2.14: Typical embedded generation connection topologies [277, 17]; (a) Basic LV installation, (b) LV installation with standby generation, (c) Dual LV installation, (d) Basic HV installation, (e) HV installation with trapped load, (f) Mixed voltage site.

therefore necessitating the need for a switchgear installation. Furthermore, an incomer section within a switchgear installation must always have G59 protection fitted. Even cases of mixed voltage embedded generation installations, such as the one illustrated by Figure 2.14(f), will require completely separate switchgear installations with separate G59 protection equipment. Such cases, however, are exceptional.

2.1.5 Current Design Practice

At present, the electrical specifications for embedded generation schemes are produced by the majority of developers utilising mental recall and organisational skills, and crude paper based aids, combined with limited design experience. The use of specialist computational tools is largely non-existent. However, word processing or spreadsheet applications are used to aid the documentation of the design process and to produce the final specification that is submitted to the manufacturer [277].

During the design process, there are the inevitable discussions between the developer and the local public electricity company. Such discussions tend to occur once the developer has arrived at a preliminary electrical specification for a scheme. During such discourses it is not uncommon for specific aspects of the proposed electrical specification to be considered and modified. Due to the complexity of switchgear installations such changes result in the omission or lack of consideration of the implications such modifications may have upon the overall proposed electrical design. For example the addition of a specialist protection device may require additional current or voltage transformers to be installed – an expensive omission if not discovered until the commissioning of the plant.

Given that the process of producing an electrical specification for any embedded generation plant may be rationalised. Through rationalisation, the design process for embedded generation switchgear can be defined. If this process can be captured in software, replicating certain aspects of design behaviour, then a considerable improvement in the consistency and accuracy of switchgear design and the final specification can be achieved.

2.1.6 Summary

This concludes the introduction and discussion of embedded generation. This section has discussed the common electrical components that constitute embedded generation schemes, excluding discussion of non-standard components such as prime movers and control equipment. The legal and regulatory issues directly affecting the development of new embedded generation schemes have been given,

indicating the very important nature of G59 and its practical application guide, ETR113. The discussion has continued with a description of the typical equipment required for safely switching and protecting an embedded generator, also indicating the other functions which a section may fulfill as part of a switchgear installation. Other equipment that constitute a switchgear section have also been introduced. Finally, various connection topologies were considered to indicate the validity of considering switchgear installations as a common instance in all embedded generation schemes.

2.2 Design Methodology

The conception, specification and implementation of an embedded generation scheme is not based upon any (clearly) defined procedure or any definite or intuitive series of logical process, documented or otherwise. Thus the development of an embedded generation project from conception through to specification then to physical implementation follows an ill-defined process common to the development of all artefacts – the process of design. This process is principally concerned with the development of an artefact from a mentally conceived idea to a complete, typically written, specification of the artefact which resolves a particular need or function; the process of constructing the artefact from a given specification to a physical, or other manifestation, is occasionally accepted as a formality, and thus not always considered as part of the design process.

Since the software described in this thesis is principally concerned with aiding a developer produce an electrical specification for the switchgear in an embedded generation scheme, it is appropriate that the process of design should be examined to crystallise the paradigms or methods utilised by human design practitioners. By examining such design paradigms and evaluating if such processes may be captured within a computer software environment, can any judgment be reached concerning the validity and usefulness of the resulting software tool? Any software package should exhibit one or more of the design paradigms presented before it may be said to exhibit any recognisable form of design expertise.

This section will explore the features that characterise design rationale in an attempt to reach a satisfactory description of the activity. The discussion will then continue to describe the five paradigms of design that currently exist, with several examples or descriptions of methods based upon these concepts. Finally, a limited review of computer aided design (CAD)[11] systems that attempt to emulate design reasoning will be presented.

[11] This acronym originally stood for *computer aided drafting* but the later development of the field resulted in the re-definition of the acronym to *computer aided design*, as detailed in Section 2.2.3, page 47.

2.2.1 What is Design?

Since the dawn of mankind, humanity has designed, either consciously or in more primitive times in the thought processes of manufacture, creating artefacts such as hunting implements, shelters, and clothing. Such tasks have attracted and occupied practitioners for scores of centuries before the development of the natural sciences. Indeed it is a characteristic of human nature to change or sculpt the world to suit our purposes; it could be argued that the act of design is a manifestation of intelligence [261].

Yet the mental thought processes that underpin and orchestrate the act of design are not completely understood. The process of design lacks a distinct, universal definition due to its complex, difficult nature. This has resulted in a multitude of definitions being proffered. Brown [41] suggests two reasons for this to be the case: the *goal state*, or final ultimate state, of a given design process is difficult, if not impossible to define. Secondly, the implications of various actions or decisions during the design process are generally not predictable.

The process of design may defy universal definition but not description. Simon [244, pp.58-59] captures the process eloquently:

> *"The natural sciences are concerned with how things are. ... Design, on the other hand, is concerned with how things ought to be, with devising artefacts to attain goals."*

It is this combination of logical reasoning combined with creative or imaginative thought processes, concepts opposing each other, that defines a complete, universally accepted definition of design rationale.

The discipline of design methodology developed independently of other fields of research in the early 1960's by pioneers such as Asimow [12], Alexander [5] and Jones [135]. However, logical and systematic analysis of design processes had been developing throughout the 1950's especially focusing upon engineering analysis and design based upon graph theory [253]. The interest in design was precipitated and assisted by the development of electronic computers, automated systems combined with a substantial interest in rationalising mechanical engineering design [229] and architectural design [23]. Figure 2.15 illustrates the diverse range of disciplines typically involved in the design of an artefact.

The multitude of disciplines engaged in design methodology initially resulted in two distinct divisions within the field being formed: design as a routine process based upon scientific reasoning and design as an inventive, innovative and creative process. Advocates of the scientific view of design view the process as a problem

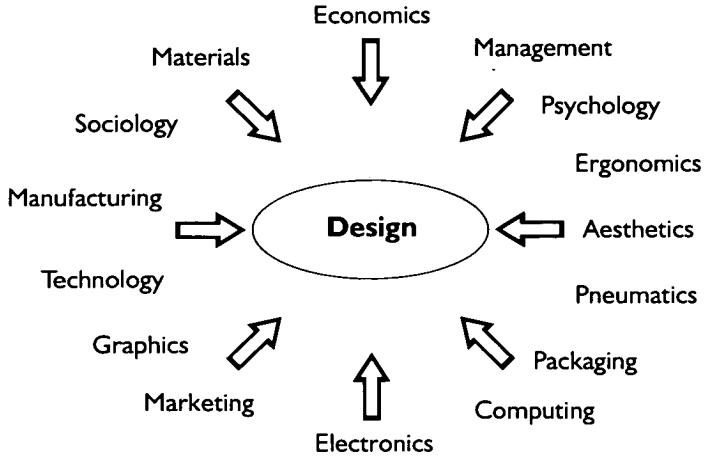


Figure 2.15: Knowledge areas involved in design [223, p.32].

solving or decision making scheme involved with finding, optimising and producing a solution; Figure 2.16 illustrates the parallels between design and scientific problem solving. The scientific approach also allows the process to be rationalised, then recorded or implemented. Supporters of the creative aspects of design view the process as a very human activity which may be understood from observing or testing practitioners as they perform design tasks, from which lessons or methods may be learned to accelerate and/or ensure that optimal solutions may be found.

Attempts have been made to unify these two contrary points of view, the natural sciences verses the engineering science (or as Simon called it - *the science of the artificial* [244]), but a complete theory has proved elusive. However, both approaches have yielded important results in furthering our understanding of design reasoning and rationale. Examination of the creative aspects of design led Rittel [221] to recognize the existence of *wicked problems* within some design domains; subsequently Simon [243] defined the concept of *ill-structured problems* – problems that are inaccessible or insoluble to the problem solving techniques of artificial intelligence.[12]

However, the development of modern technology has driven the need for the study of design methodology and, at least partly, yielded possible solutions. In 1964, Alexander [5, pp.3-4] noted that design information is hard to handle, widespread, diffuse and unorganised but the quantity of information was then well beyond the capabilities of individual designers; this situation has since become more acute with the spiralling complexity of modern artefacts compounded by the limitations of human reasoning [177]. This situation has resulted in the exploration of the scientific approach to design by many researchers [261, 73, 66, 30], encouraged by developments in artificial intelligence and computational hardware since both disciplines allow the study and capture, respectively, of reasoning.

[12]The term 'artificial intelligence' is defined in Section 3.2, page 53.

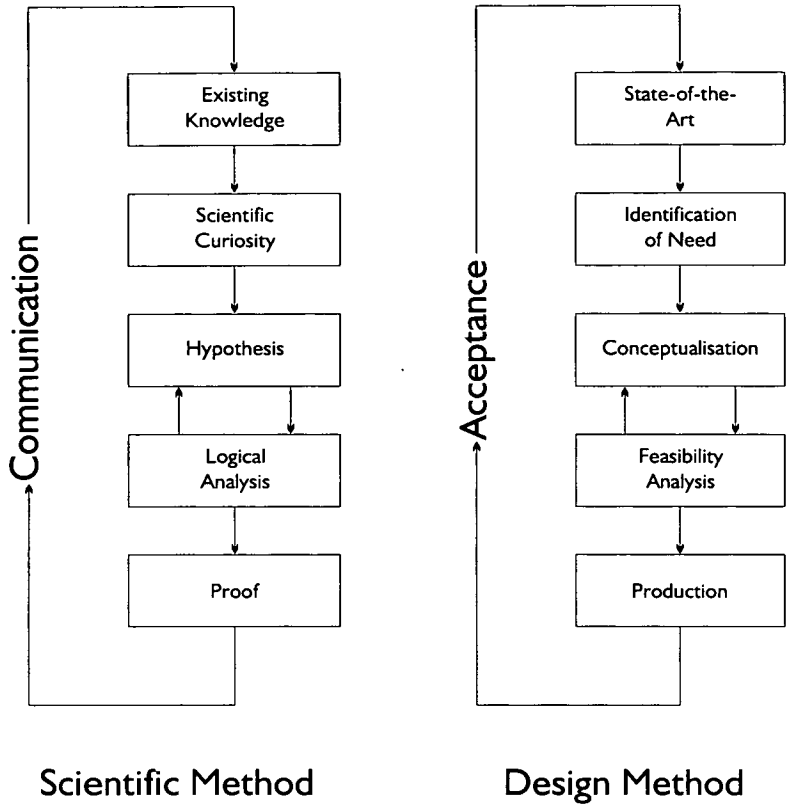


Figure 2.16: Scientific knowledge and product design loops [115, p.36].

The universal feature of design, from either the natural or engineering point of view, is that design practitioners perform a purposeful act; the identification of needs, the isolation of the problem, and the requirement for action to improve the situation.

2.2.2 Design Paradigms

The entire field of design methodology may be categorised into one of five design paradigms, according to Braha and Maimon [34]. Each paradigm provides a framework for the solution of design problems; from which any number of practical or concrete procedures or computational tools may be derived. Each paradigm is briefly outlined in the following sections.

2.2.2.1 The Analysis-Synthesis-Evaluation Design Paradigm

The Analysis-Synthesis-Evaluation (ASE) design paradigm has been widely adopted and utilised throughout the engineering disciplines. As indicated by its name, this paradigm has three key stages which are summarised below:

Analysis: The identification of need(s) and the development of an understanding of the problem domain, resulting in an explicit statement of requirements (goals).

Synthesis: The generation of possible solutions that meet the needs of the problem.

Evaluation: The consideration and assessment of possible solutions to the original needs and requirements resulting in the selection of the most appropriate solution.

There are many advocates of this paradigm who expand the above three steps into between five and twenty five steps [5, 207, 80, 30]. Frequently, proceeding through the paradigm once fails to yield a suitable or agreeable solution, but the process does produce valuable information concerning the design domain which may be used to change the initial requirements, therefore requiring the process to be performed again. Such cases of repeated application of the ASE paradigm are frequently referred to as *iterative design*.

2.2.2.2 Case Based Design

In design domains where there is no obvious or direct correspondence between form and function, or components of an artefact and the overall operation of the artefact, designers will utilise previous solutions and adapt them to new circumstances. Such selection and adoption is performed on a case by case basis, hence the name of this paradigm.

Case based design exemplifies how engineering students are taught design; typically no design algorithms exist so solutions are provided by the incremental modification of previous solutions. This paradigm embodies the intuitive and creative aspects that design practitioners display, typically as an art or skill within a specific domain. This is still an aspect of engineering practice highly valued to the present day, where experience out ranks formal education [261].

2.2.2.3 The Cognitive Design Paradigm

Cognitive design is concerned with the study of how humans perform tasks, both mental and/or physical, and the interaction they have with machines (frequently referred to as the *man-machine interface*), typically computer systems. Research based upon this paradigm explores, by the use of psychological experiments, how designers perform design and thereby allow tools to be constructed that facilitate design.

For example, Thomas and Carroll [267] report a selection of experiments including analysis of professional designer's discourse, analysing individual designers' solutions to an identical problem, and studies involving the observation of peoples' responses when given restricted or open style tasks. The authors provide several methods to assist design practitioners, such as unstructured word lists to assist people to generate ideas. A similar but more comprehensive series of methods is reported by Cross [60, 58].

More recently, Cross and Cross [59] have reported upon design strategies employed by expert designers, concluding that successful designers adopt a systemic view of a design situation, frame the problem in a challenging manner and then utilise the first principles of engineering to generate the solution. Such results deepen our understanding of design, allowing future research to develop more comprehensive theories about design.

2.2.2.4 The Artificial Intelligence Design Paradigm

The field of artificial intelligence attempts to emulate intelligent human thought processes [32]. Within design, artificial intelligence techniques are utilised to capture knowledge of the problem domain and by using this knowledge, provide solutions. By attempting to automate the process, the machine must be able to evaluate the design options available during the process and determine the most appropriate choice. Therefore the artificial intelligence paradigm models a design domain as a series of states; commencing with an initial problem state, through the design process via intermediate states, to a final goal state. In order to proceed from one state to another a series of operators must be defined. Such operators are based upon the functional requirements or goals of the process, design constraints and meta knowledge.

The organisation of intermediate states frequently relies upon the *function-structure-behaviour* of the artefact or its components and their relationship between the outside world and each other. Therefore by modelling the causal reasoning that composes an artefact, combined with the fundamental physical laws applicable to the domain and searching through the remaining limited design options, optimised design solutions can be arrived at.

Complex design domains have to be divided into manageable sizes comprising of modellable artefacts, either individually or in groups. This is achieved by utilising a *design strategy*; Simon [244] proposed the three main strategies which are *bottom-up*, *top-down* and *meet-in-the-middle*. Bottom-up design commences with a series of basic structures which are combined to produce the final artefact. Top-down design, which is also one of the most frequently used human design practices

due to its simplicity and effectiveness, studies the final behaviour required of the artefact, and proceeds by subdividing this behaviour into smaller behaviours, continuing the process until each small behaviour relates to a pre-defined or existing component of the problem domain. Composition of the components results in the required behaviour of the original design task. Meet-in-the-middle design selects either bottom-up or top-down design depending upon the suitability, difficulty and information available at various stages of the process.

Artificial intelligent technologies include expert systems, case based reasoning, neural networks and genetic algorithms. These technologies are discussed, including their operation, in Chapter 3 and their use for modelling design problems for various domains is reviewed in Chapter 4. It should be noted that the artificially intelligent design paradigm is particularly appropriate within tightly coupled, highly integrated and ill-defined design problems.

2.2.2.5 The Algorithmic Design Paradigm

This approach to design hinges upon the premise that design problems are *well-structured*,[13] thus amenable to a domain specific algorithm which if given a set of requirements, typically numerical parameters, can generate a design solution in a finite number of stages. The resulting solution must be accessible so that it may be checked against definite criteria to test if it fails, meets or exceeds the requirements of the design. This design paradigm is frequently referred to as *parametric design*.

There are several other design methods based upon the algorithmic paradigm. For example, Haugen [108] has developed a design method to account for the random phenomena experienced by mechanical engineers based upon probability. This approach has been extended to a general design theory by Siddall [242]. A practical demonstration of such an approach within power engineering is reported by Tse *et al* [270].

This approach to design has been and remains one of considerable interest, especially to engineering disciplines, as it provides methods of design automation utilising modern information technology. There are numerous areas within mechanical and electrical engineering where the automation of complex systems design is invaluable [75, 210, 22, 29]. The majority of design problems are ill-structured and thus have no definite finish point, or are fragmentary in nature, therefore the algorithmic paradigm is not applicable; however, such problems may be divided into segments that are well-structured and thus amenable to this paradigm.

[13] As defined and described by Simon [243].

2.2.2.6 Summary

The previous five sub-sections have indicated the fundamental paradigms which currently exist within design methodology. It should be noted that a certain but very limited degree of commonality exists between all of the paradigms. Some design methods may be classified by more than one paradigm, which is particularly true of certain methods that lie between algorithmic and artificially intelligent design paradigms; genetic algorithms, for example, can be technically classed within either paradigm.

2.2.3 CAD Software

An engineering drawing allows the unambiguous representation of an artefact on a two dimensional space thereby allowing communication of ideas and details between engineers. However, the process of drafting such drawings was a time consuming and repetitive task until the introduction of computer aided drafting (CAD) [73, p.42].

By the early 1960's, computers had been developed that operated *on-line*, allowing users to interact directly with the machine, obtaining feedback instantaneously. This facilitated Ivan Sutherland to develop 'Sketchpad' in 1962, an interactive computer graphics system that allowed engineering drawings to be created and edited directly using a light pen and a keyboard. The details of this pioneering work are given in Sutherland's Ph.D. thesis [263].

By 1964, General Motors extended Sutherland's concept of drafting engineering drawings on a computer to also link this information to a computer numerically controlled (CNC) machine, thereby allowing mechanical components to be directly machined. The system was called Design Augmented by Computer One (DAC1) and was the first computer aided design and computer aided manufacture (CADCAM) system developed [30, p.266].

These early developments have shaped the entire structure of CAD/CAM computer systems. By organising and eliminating the tasks previously required to produce drawings, CAD has allowed designers to concentrate on design problems while also reducing the time required to produce an artefact, thus allowing substantially more thorough analysis of artefacts at the early design stages, reducing prototype and manufacturing costs [73, p.44].

Such advances in computing technology and CAD software were quickly adopted by industry throughout the 1970's and 1980's, accelerated by the declining cost of computing systems. During this period, other aspects of the design process were also being included into CAD software other than the drafting of the two

or three dimensional physical aspects of artefacts. For example, projected costings, estimated manufacturing timescales and the inclusion of functional aspects to the geometrical representations of components were also modelled as part of the CAD systems [57, 171]. Such development continues to the present day based upon advanced design models using algorithmic and artificially intelligent design paradigms [274].

2.2.4 Summary

This section has introduced the field of design methodology and charted, briefly, the development of the field to the present. A discussion on the ubiquitous question – *what is design?* – has been presented. An introduction to the key paradigms of design methodology has been given and finally the basic history and operation of CAD systems has been described.

2.3 Rationalisation

This section is the third and final part to this chapter. Having discussed the main features of embedded generation schemes in the first part of this chapter, followed by a review of design methodology in the second part, this final part discusses how the design process of switchgear for embedded generation may be rationalised. In other words, how design paradigms may be applied to the process of switchgear design. This is not intended to be an exhaustive list but to illustrate that the design of switchgear for embedded generation has scope for automation within a suitable software technology.

2.3.1 Legal and Regulatory Requirements

As indicated in Section 2.1.1, the Electricity Association Recommendation G59, and supporting implementation guide ETR113, detail the minimum safety requirements for embedded generation schemes connected at either low or high voltage connections. To ensure consistency and reduce the complexity of the proposed design tool, design practice for high voltage switchgear will be assumed throughout the project.[14]

[14]It could be argued that by assuming high voltage switchgear design practice for low voltage installations that the specifications produced by the software will be over-designed. This aspect of the software is discussed in Section 7.1.2.

With regard to earthing, high voltage installations will be assumed to have a switched neutral earthing arrangement connected via circuit breaker to the developers independent earth electrode. Other earthing arrangements do exist[15] but are uncommon. However, provision will be required to allow this assumption to be reviewed in the future. Low voltage installations will be assumed to have a TN-S[16] utility connection. Again other low voltage utility connections exist but are not frequently encountered.

The most important aspect of G59, the protection arrangements for embedded generators, must be fully and completely implemented by the software. This may be realised by utilising the algorithmic design paradigm to develop a computer interpretable representation of Table 2.1. In cases where G59 is open to interpretation or dependent upon installation site conditions other factors within the design will be considered; these situations are discussed in the following section and Section 2.3.5.

2.3.2 Connection Topologies

From observation of typical topologies, as illustrated in Figure 2.14, it may be observed that every embedded generation installation requires switchgear with G59 protection fitted. This is the case with both low and high voltage installations. The location of G59 protection equipment outside of the switching section for a generator does occur on sites with either both low and high voltage generators of similar generation capacities or sites with large numbers of machines and a single G59 protection installation. However, such sites are very limited, exceptional and therefore beyond the scope of this project.

Below 33kV, most utility networks are operated in a radial fashion in the interests of simplicity and economy. Due to the lower security offered by such networks regional electricity companies will tend towards the use of automating switching via remote control. Consequently the risk of islanding increases the further the generator is located away from the local distribution transformer (or substation). Such a risk is substantially increased if the generator can supply the loads connected to the distribution network in its vicinity [225]. In such cases it is very prudent to connect a loss of mains protection device; indeed the local utility will assess a proposed embedded generation scheme and normally insist that such a device is fitted. Performing an assessment of the local distribution network is normally impractical as the information required only resides with the local utility which may not make such information available to the public. Analysis of local or site loads also fails to

[15] Refer to ETR113 [17], Figure 5.8, page 37.

[16] As defined in the IEE Wiring Regulations [123], page 15.

reduce this ambiguity, therefore the decision to fit loss of mains protection will be based upon the generator specification, discussed in Section 2.3.5.

2.3.3 General Switchgear Observations

Section 2.1.3 indicated that switchgear sections have a variety of purposes other than the switching, monitoring and protection of generators. However, since switchgear installations rarely consist of just a generator section it was proposed to model all switchgear sections.

The application of protection and instrumentation equipment may be determined from a section's function as indicated by the user. Thus using either the algorithmic or artificially intelligent design paradigm, the appropriate equipment can be selected and designed for that section. Furthermore, since both protection and instrumentation equipment require connection to either, and in some cases both, current or voltage transformers and auxiliary supplies with appropriately rated fuses, such design reasoning lends itself readily to automation. The implementation of such switchgear design reasoning would also allow the designer to monitor the design as it proceeds. For example, the burdens placed on a particular voltage transformer at any point during the design process can be readily known.

2.3.4 Protection Requirements

Traditionally, protection devices have been based upon electromagnetic principles for their operation[17]. As a result individual devices have only one condition for which the device provides protection against. However, advances in protection technology have resulted in electronic multi-function devices becoming available. Although currently not widespread, such devices are likely to become commonplace in the future, providing protection for embedded generation projects. Currently their high cost and unproven track record excludes them at present from being adopted, especially by small scale embedded generation schemes.

Multi-function protection devices are capable of large numbers of sensing and logical operations with which it is easy to create an overly complex protection scheme, that may in fact under-protect or over-protect a generator. Both situations are undesirable, with excessive protection programming frequently resulting in complex commissioning, troubleshooting and testing.

It is proposed that the software should focus upon discrete protection devices by default, as individual electromechanical or electronic devices are still currently

[17] As discussed in Section 2.1.3.2.

preferred by developers. However, in recognition of the fact that multi-faceted digital protection devices will become industry standard over the next decade the model should have a capability to allow such devices to be included in a design scheme. This is accomplished by allowing the model to include a function attribute with each protection device, specifying the protection functions that the device performs. It is proposed that it is the functionality of the protection devices present in a protection scheme that will be checked to ensure that the said scheme is sufficiently protected and complies with G59. Such an arrangement appears to strike a balance between keeping the model relatively simple while adaptable or extendable for the future.

Digital protection devices require lower current and voltage transformer burdens to operate as they only sample and quantize the waveforms; typical burdens are less that 1A at 110V or less. Older electromechanical devices, however, require significant burdens to energize their internal magnetic and mechanical circuitry with burdens of 5A standard at either 110 or 400V [191]. Both of these conditions must be reflected in the software model.

2.3.5 Generator Requirements

In general, electrical generators are classified into one of two categories; either induction (or asynchronous) machines and synchronous machines. In terms of switchgear design induction machines are easier to switch and protect due to their simplicity; they do not require synchronising, automatic voltage regulation or exciting equipment. Synchronous machines, however, require the aforementioned equipment. In terms of installation, synchronous machines may be installed further away from distribution substations, require limited or even export reactive power and may operate through a wide range of power factors. Such advantages are not available from induction generators as they require reactive power to operate and do so at a fixed power factor [48].

From a regulatory point of view, G59 is not concerned with the classification of a proposed embedded generator, but whether the generator is mains dependent or not. *Mains independent* generators are generally associated with synchronous machines since they may use an external electrical power source or a permanent magnet/tertiary winding supply for excitation allowing such machines to operate without a mains supply. Such machines must be fitted with a loss of mains protection device.

Induction machines are generally thought to be *mains dependent* as they can not generate without a mains supply since they rely on it for excitation. However, induction machines have been shown to remain excited for sustained periods following a loss of mains supply; obtaining their excitation from other generators, power

factor correction capacitors or utility system cables or overhead lines [130, 132, 225]. Such sustained generation is deemed to be islanding and therefore requires loss of mains protection to be installed.

It is therefore proposed that the design software will take into account the ability of a machine to be mains dependent or not. If a synchronous machine is selected, then loss of mains protection will be selected automatically. If a single induction machine is selected with power factor correction capacitors of a capacity below 10% of the machine's rating then loss of mains protection will not be selected. However, if two or more induction generators are selected, or a single machine has power factor correction capacitors of a capacity above 10% of the machine's rating then a loss of mains protection device will be included.

2.3.6 Summary

This section has highlighted the main features of the switchgear design process for embedded generation schemes and how aspects of the design process may be rationalised and captured using one or more design paradigms.

2.4 Chapter Summary

This chapter has laid the foundations of this project. The first part of the chapter introduced embedded generation, with the discussion covering legal and regulatory issues, the equipment required for constructing and operating an embedded generator, typical connection topologies and current design practice of embedded generation schemes. The second part of the chapter reviewed design methodology with an attempt to describe what design is and the five design paradigms of design. The third and final part of this chapter discussed aspects of the switchgear design process, indicating some of the features of the process that allow it to be rationalised and encoded within a software design tool by utilising one or more design paradigms.

Having established that the automation of aspects of the design of embedded generation switchgear is possible, Chapter 3 reviews the current software technologies available, with which a software design tool may be realised.

Chapter 3

Design Software Technologies

Since the advent of modern electronic computing a large number of programming languages and artificially intelligent technologies have been created. It is the purpose of this chapter to review the currently available technologies and their use within the Electricity Supply Industry (ESI), thus forming a basis upon which a decision can be reached on their suitability for coding an embedded generation design tool.

3.1 Chapter Overview

Available software technologies for developing the design tool fall into one of two categories: either an Artificially Intelligent (AI) technique, or the use of a programming language based upon a programming paradigm. The structure of this chapter reflects these two categories, both of which are discussed and assessed individually, with a brief indication of their impact upon the electricity supply industry.

3.2 Artificially Intelligent Techniques

The term *artificial intelligence* was coined by John McCarthy and officially adopted by the research community during the summer of 1956 at the first artificial intelligence conference at Dartmouth College. Artificial intelligence was then defined as [32]:

> *'computer processes that attempt to emulate the human thought processes that are associated with activities that require the use of intelligence.'*

In 1958 McCarthy developed the first complete AI system, 'Advice Taker', a hypothetical program able to use knowledge to search for solutions to problems. Research continued, exploring artificial neural networks and genetic algorithms, with rapid developments and high expectations [233].

It was not until the mid 1960's that the mainly algorithmic search mechanisms used in artificial intelligence began to show their limitations, stagnating research in the area. The fortunes of artificial intelligence did not improve until a decade later with the development of knowledge based systems, leading to expert systems. By the beginning of the 1980's the first commercial expert systems appeared.

The maturing of artificial intelligence, coupled with the development of powerful, cheap, personal computers (PCs) has led to the application of artificial intelligence, especially expert systems, in a large number of fields including power systems. In the late 1990's artificial intelligence continues to develop with new technologies such as intelligent agents.

The following sections briefly explain the fundamental principles behind the aforementioned artificial intelligent technologies, their use within the electricity supply industry and for artificial design tasks.

3.2.1 Knowledge Based Systems

The first generation of artificial intelligence researchers focussed upon solving problems by utilising search algorithms combined with elementary reasoning; a process called *weak methods* [233, p.22]. This approach quickly showed deficiencies as moderately complex problems presented an enormous search space, frequently too large to be completely explored. It became apparent from the study of human experts that they are able to achieve high levels of performance because they utilise domain specific knowledge. Thus the focus of artificial intelligence research changed to developing systems which could capture knowledge of a specific domain in a machine interpretable form, thereby developing a branch of artificial intelligence termed *knowledge based systems*.

Within the field of knowledge based systems, three fields of activity exist, the most utilised being expert systems, followed by case based reasoning and model based reasoning [159]. Only expert systems and case based reasoning will be explored here, as both these technologies have been used in prototype and production applications with reasonable success within the Electricity Supply Industry and in the domain of artificial design.

3.2.1.1 Expert Systems

The basic features of an expert system is illustrated in Figure 3.1. The essence of an expert system is the *knowledge base* which contains the problem solving knowledge of the application domain. For example, the most popular form of knowledge representation is *rule-based* knowledge, which is recorded in a series of statements called *rules* of the form:

if condition(s) **then** *action(s).*

A rule is said to operate if its conditions have been satisfied and the resulting action may perform calculations and / or activate other rules. Knowledge representation may also be of the form of *object-attribute-value triplets, semantic networks, frames* or *logical expressions* [278].

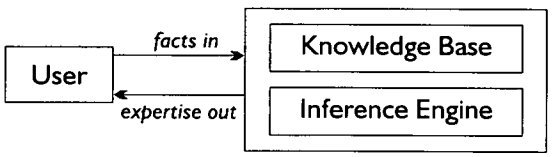


Figure 3.1: Conceptual diagram of an expert system [118, p.216].

During operation of an expert system, the user will supply facts relevant to the system's domain of expertise, normally via a series of dialogs within a graphical user interface, resulting in the *inference* engine consulting the knowledge base. The inference engine in effect acts as an interpreter between the user and the knowledge base. The conclusions drawn from the knowledge base as a result are normally in the form of a series of linguistic statements, directly interpretable by the user and which constitute expert advice. In some instances it may be of interest to view how a particular conclusion, or series of conclusions, was reached; this may be performed by the expert system recording the path of decisions taken through the knowledge base [156, 157].

Due to their extensive use and commercial development for over two decades the key advantages and limitations of expert systems have been widely explored and documented; they are summarised below [118, 127]:

Advantages:

1. The possibility of providing and sharing expert knowledge with anyone having access to the required computer equipment.

2. The capacity to permanently encapsulate knowledge acquired from one or more human experts.

3. The ability to provide advice and knowledge quickly, with more flexibility and improved consistency than if human experts were used.
4. Expert systems provide excellent performance but in limited domains.
5. If coded appropriately, expert systems have the ability to justify advice to the user.

Limitations:

1. Expertise is difficult to extract from experts and frequently is highly heuristic in nature.
2. Heuristic rules tend to fail when given insufficient or unexpected data.
3. There is a lack of scientific methodology in knowledge engineering resulting in fragmented, solitary systems.
4. Rule based searches become inefficient with very large rule bases (> 500 rules) and are unable to access or manage large volumes of information.
5. Maintenance, verification and validation of rule bases can become exceedingly time consuming.
6. The explanations of reasoning are descriptive – only indicating conclusions already implicit in the knowledge base.
7. The knowledge contained within all expert systems is very task specific.

Initially expert systems were coded entirely using a programming language such as LISP or implemented using specialised hardware, but after a decade of commercial development coupled with the advent of powerful, general purpose workstations or desktop personal computers, several expert system shells became available incorporating integrated tools within a graphical user interface; Figure 3.2 illustrates this arrangement. Such expert system shells came with empty knowledge bases in which the user could enter domain specific knowledge. A review of such systems was performed by Mettrey [173] at the height of their popularity.

The first commercial expert system was the 'R1' developed for the Digital Equipment Corporation in the early 1980's and was judged an enormous commercial success [233, p.24]. Expert systems were then developed for many industries, early adopters using the technology for financial, medical and civil construction applications. The electricity supply industry was no exception, with the first prototype expert systems being developed by the mid 1980's. Indeed, within the electricity supply industry, expert systems have been the most frequently applied artificially intelligent technology to date, with numerous application areas; the main areas of

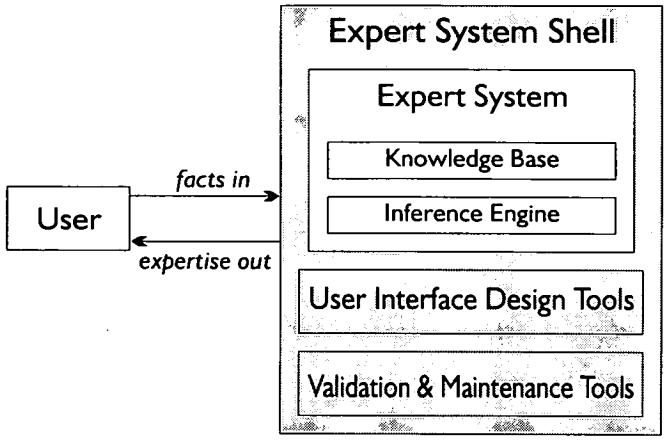


Figure 3.2: Conceptual diagram of an expert system shell.

interest and development being alarm processing [290], planning [91] and control [97]. Other areas of research within the electricity supply industry include fault diagnosis, system restoration, scheduling, security and protection [86, 162, 62].

3.2.1.2 Case Based Reasoning

The key paradigm underlying Case Based Reasoning (CBR) is analogy; the importance of previous experiences to the solution of new problems by exploitation of their partial similarity. Developed from cognitive theories on the operation of human memory, case based reasoning was introduced by Schank and Abelson [239] in 1977. They proposed that general knowledge of certain situations is recorded in human memory in the form of scripts, thereby allowing expectations of similar situations to be perceived and thus inferences performed. A period of research and development followed as case based reasoning matured, with the first commercial systems being deployed by Lockheed Aeronautical Systems Company in 1990 [279].

The operation of case based reasoning systems is based upon the case based reasoning cycle, illustrated in Figure 3.3 and described as follows:

1. A problem specification is entered into the system, resulting in the case library being searched for case(s) similar in nature which are then retrieved.
2. The case(s) selected are evaluated and a case is selected (as the basis of a possible solution).
3. The proposed solution is evaluated against the problem criteria, revised and modified if required.

4. The confirmed solution is outputted and the new solution recorded (retained) in the case library.

The key component of any case based reasoning system is the *case library*; each case represents contextual knowledge representing an experience within the application domain in a structured and indexed manner. The basic representation of a *case* is an ordered pair:

$$(problem, solution).$$

The problem element of a case describes the problem domain, when the case occurred, and its circumstances. The corresponding solution element records the total or partial resolving of the problem, including principles utilised, explanations or descriptions of how the solution may be obtained and/or decisions taken. In certain instances, the solution may require an action with effects that can not be determined from the data supplied. Therefore the concept of a case has been extended to a triple:

$$(problem, solution, effects),$$

where the effects detail the consequences of the solution if implemented [219].

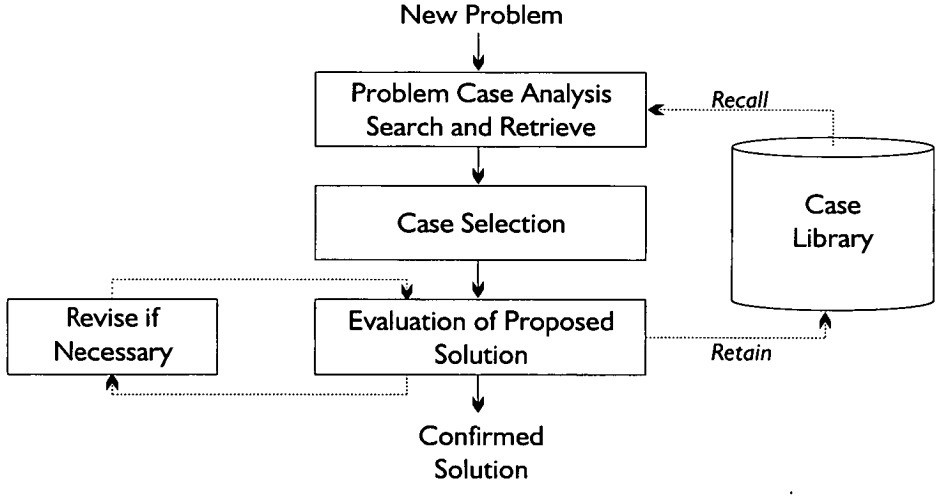


Figure 3.3: The case based reasoning cycle [164, (Adapted) p.35].

The representation of cases includes all standard knowledge based representations including semantic networks, rules and logical expressions, with the majority of case based reasoning systems utilising a combination of frames and objects. Bergmann [27] presents a detailed discussion with practical, industrially applied, examples of case representation with CDROM software.

The main advantages and limitations of case based reasoning systems are detailed below [187]:

Advantages:

1. The process of acquiring cases is normally significantly shorter in duration than the development of rules for a similarly specified expert system.
2. Case based reasoning systems are easier to construct, modify and maintain compared to other knowledge based systems.
3. Cased based reasoning systems learn new cases as they arise without reprogramming.
4. Case based reasoning systems are better understood by their users as they capture knowledge using familiar concepts and terms.

Limitations:

1. To attain reasonable performance from case based reasoning systems a sufficient number of cases have to be acquired.
2. Discovering a suitable case representation for a given domain is a difficult task.

The amount of interest in case based reasoning within the electricity supply industry has been limited mainly because case based reasoning, as well as model based reasoning, are recently matured knowledge based technologies, and, although independent technologies, they are generally applied within an expert system shell, forming only a minor proportion of the overall solution to an application domain. Example applications within power systems include the fault diagnosis of steam turbines [94, 95], generator fault diagnosis [187], electrical drive diagnosis [178] and design of electromagnetic devices [227].

3.2.2 Artificial Neural Networks

The natural world provides many examples of creatures which are able to perform computational tasks beyond the capabilities of present artificial systems. The study of such biological entities reveals that these systems are constructed from highly parallel and interconnected networks of simple processing elements called *neurons*. The computational paradigm which mirrors the processing architecture of biological systems was originally proposed by McCulloch and Pitts [167] in the early 1940's, based upon the concept of the artificial neuron.

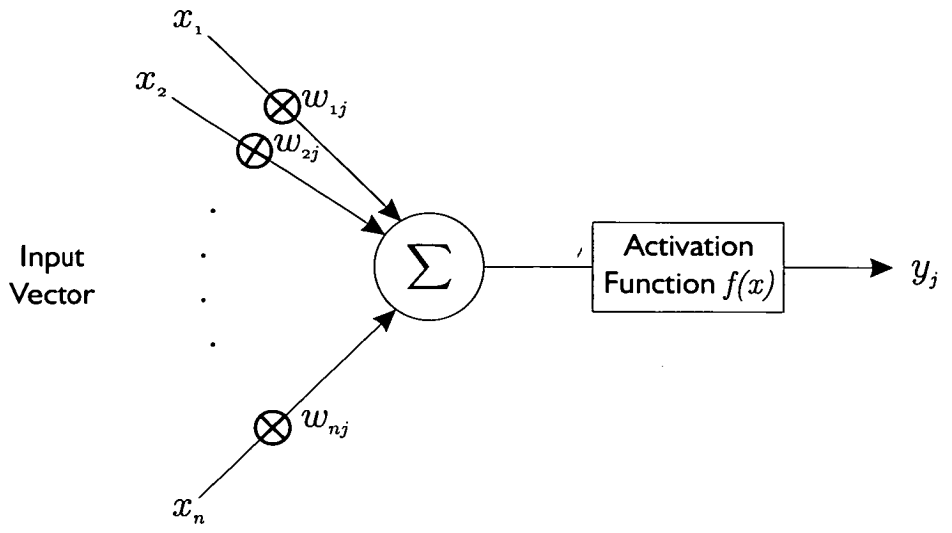


Figure 3.4: A model of an artificial neuron, as proposed by McCulloch and Pitts [109, p.9].

The artificial neuron operates by taking the inputs x_i, applying a *weight*, w_{ij}, summing all the weighted inputs together, and applying an *activation function* $f(x)$ to produce an output y_i, as illustrated in Figure 3.4. The activation or *threshold function* selected depends upon the neural model and learning algorithm being utilised; some examples of such functions are given in Figure 3.5.

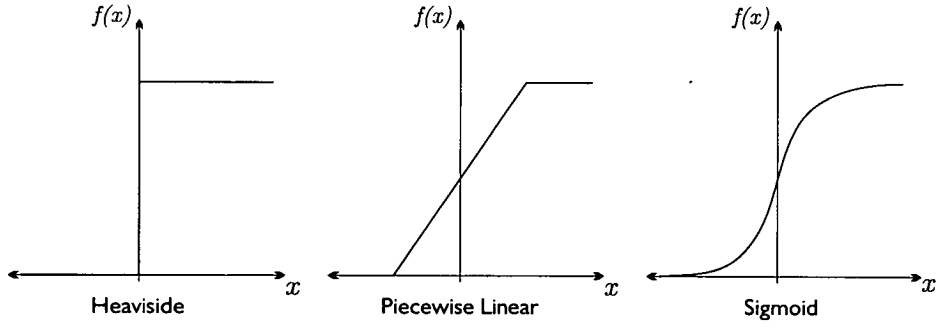


Figure 3.5: Examples of threshold functions [109, p.11].

Networks of neurons may be constructed, termed Artificial Neural Networks (ANN's). Many such network architectures exist *e.g.* the Multi Layer Perception (MLP), Kohonen's self organising feature map, Adaptive Resonance Theory (ART) and recurrent networks [188]. The architecture selected and number of neurons required depends upon the task the ANN is required to perform.

Generally ANN's are applied to problems involving massive amounts of data from which a limited number of decisions may be deduced, normally via an intricate, non-linear mapping [137, p43]. From such large data sets, standardised training data may be derived of a similar nature as that which the network will

encounter in service. During training the ANN learns the non-linear mappings of its application space by setting and storing the weights, w_{ij}, at the inputs of each neuron. When in use, the weights of an ANN may be either static or dynamic; if static the ANN performs the same mappings as it learned during its training but, if dynamic, the ANN can learn new mappings by adapting weights in proportion to an error term, therefore allowing the network to adapt to new data. They are hence commonly known as *learning* ANN's.

The key advantages and limitations of ANN's are summarised below [1]:

Advantages:

1. ANN's are adaptive, able to digest data and subtle relationships, and learn from them accordingly.
2. ANN's can generalise by being able to process incomplete or noisy data.
3. ANN's are non-linear, thus complex interactions present within the data may be captured.
4. ANN's are highly parallel in organisation hence allowing independent, simultaneous operations to be performed.

Limitations:

1. ANN's are unable to easily account for their results, thus results have to be accepted at face value or externally verified,
2. Training methods for ANN's are not completely understood therefore selecting an architecture and size of an ANN for a given problem is currently ill-defined,
3. ANN's require large amounts of computational resource especially during the training period.

ANN's have been extensively demonstrated and utilised within the electricity supply industry. The tasks at which they excel are listed below with examples:

Classification: Detection and classification of faults [137], fault discrimination [281],

Recognition: Fault recognition for transmission lines [87, 262], identification of voltage stability in power systems [250],

Optimisation: Dynamic analysis of voltage stability [240], simulation of power plant transients [83], reduction of external power system networks for modelling and analysis [154].

Aggarwal [2] presents a more detailed discussion of example applications of ANN's in power systems.

3.2.3 Fuzzy Systems

The concept of fuzzy set theory was created by Lotfi A. Zadeh in his seminal paper published in 1965 [291]. Zadeh continued the maturing of fuzzy set theory by being a key contributor to its development which led to the creation of fuzzy logic [141].

The motivation for Zadeh's work was that he realised the unsuitability of traditional analytical techniques for the solution of future engineering and information processing problems in several diverse areas, including natural language understanding, pattern recognition and system control. The use of such analytical techniques resulted in complex models, encouraged by the availability of increasing computational power, but the models created became extremely difficult to analyze and understand. Zadeh realised that this complexity and resulting precision was largely unnecessary and could be elevated by introducing various *degrees* of circumstances into models [215]. In other words, Zadeh created variables to indicate the strength, or degree, of relationships between the components of a system.

Instead of creating a model of a problem with specific logical elements producing crisp, or *discrete* solutions, the application of fuzzy logic to the model allowed the conception of a series of functions with continuous, or *fuzzy* solutions; *e.g.* discrete solutions are characteristic of boolean logic, either zero or one, as indicated in Figure 3.6 by the left hand side graphs, whereas fuzzy solutions are vague, stating a value between zero and one representing a degree of certainty, as shown on the right hand side of Figure 3.6.

In Figure 3.6 the graphs $C(x)$ and $D(x)$ both represent single fuzzy membership functions. In practical applications, however, fuzzy membership functions frequently overlap in a complementary manner with each function given a contextual meaning representing an aspect of the model. The degree of certainty of such functions always sums to unity; a simple example of such a model is illustrated in Figure 3.7 where fuzzy sets are used to represent busbar voltage levels. Instead of presenting an absolute value to the user, the information may be simplified by allowing voltage levels to be classified as either low, normal or high, therefore making the information more readily interpretable. For example if the voltage is 0.92pu, it is unity or 100% certain that the system voltage is low, or if the voltage is 1.02pu, then the system voltage is 80% certain to be normal but 20% certain to be high. Note that $f(x)$ is continuous, even between voltage classifications.

The application of fuzzy sets or logic in isolation to the electricity supply industry is relatively uncommon; those examples that exist include power flow control

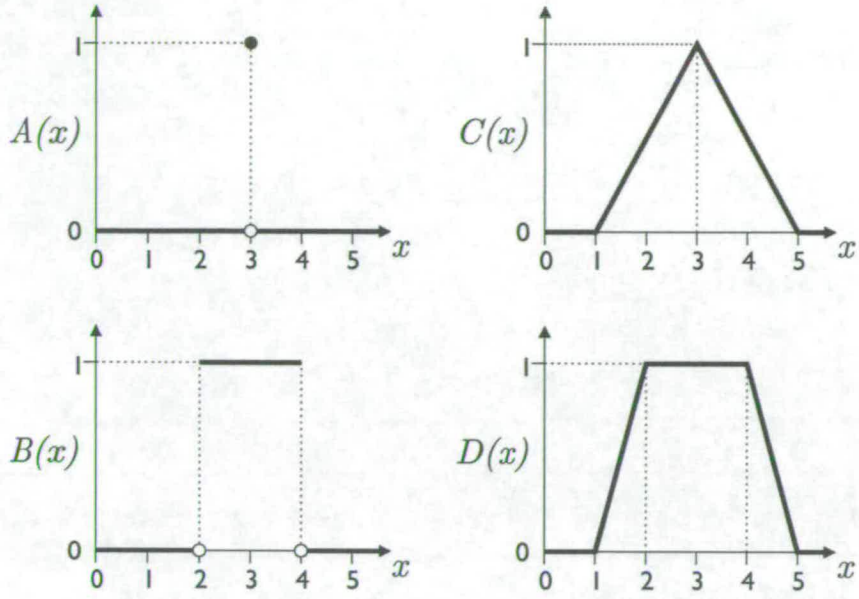


Figure 3.6: Examples of membership functions [141, p. 1.2].

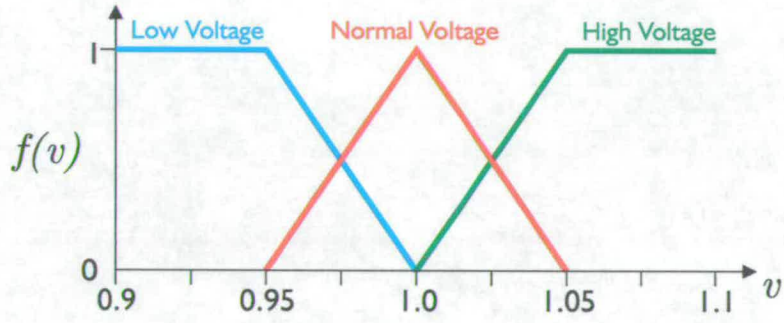


Figure 3.7: Example of fuzzy sets modelling busbar voltage levels [146, p.5].

[153], power system stabilizer control [269, 116], load flow calculations [158], and short-term load forecasting [287].

Fuzzy systems, however, are being increasingly utilised in combination with other artificially intelligent technologies; in particular expert systems, artificial neural networks and genetic algorithms. The development areas of these hybrid systems are within power system planning, operation and analysis, *e.g.* power dispatch [252], distribution load shedding [271], network reconfiguration [251], and fault diagnosis [184].

3.2.4 Genetic Algorithms

Genetic algorithms are heuristic search algorithms developed from the biological study of natural selection and genetics. Both empirically and theoretically proven,

genetic algorithms produce robust searches in complex, often large, spaces by application of a fundamental natural principle – survival of the fittest [88]. Evolution strategies, evolutionary programming, genetic programming and simulated annealing, with genetic algorithms, comprise a field of research referred to as *evolutionary computing*. Within this arena genetic algorithms have received the most attention in terms of both theoretical basis and application.

The study of genetic algorithms originated in the late 1950's with the investigation of genetic systems using digital computer simulations [99, p.89] in order to gain understanding of natural phenomena. It was not until 1962 when John Holland performed his ground breaking work on utilising genetic type operators to artificial systems that the field of evolutionary computing was created [119]. Holland was not just interested in optimisation or heuristic search problems; his research goal was to create general programs able to adapt to arbitrary environments. In pursuing this goal, Holland created the foundation theory of genetic algorithms.

The term *genetic algorithm* was first defined and given its first practical application in 1967 with J.D. Bagley's pioneering doctoral thesis at the University of Michigan. Bagley used genetic algorithms to explore game theory developing a structure and method of operation not unlike genetic algorithms of the present day [99, p.93].

A basic genetic algorithm consists of four elements:

1. a *population* of *chromosomes*,
2. a *fitness function*,
3. a *crossover operator*, and
4. a *mutation operator*.

A chromosome is a binary string which represents a possible valid solution within an application domain or search space. Each binary element of the chromosome is a gene. A group of chromosomes form a population. Thus if a genetic algorithm represents a problem domain, a population of chromosomes represents a series of possible solutions or outcomes.

The fitness or *reproduction function* describes the problem domain algorithmically in terms of some measure to be maximised. By assessment, the fitness function determines the *fitness value* of each chromosome, the fitness value reflecting the performance or ability of the chromosome within the application domain. The larger the fitness value, the greater is the probability of the chromosome contributing to one or more offspring in the next generation when mating occurs.

Since different chromosomes represent different solutions of varying levels of fitness, improved solutions may only be attained if information can be passed from one chromosome to another of the next generation. To allow this information exchange to occur within a population, the crossover operator mimics the natural process of mating by allowing new chromosomes to be created from existing ones; the fundamental form being single point crossover, illustrated in Figure 3.8. By allowing chromosomes to exchange genes (or sections of their binary strings), new solutions to the problem domain are created and will then be explored in the next generation.

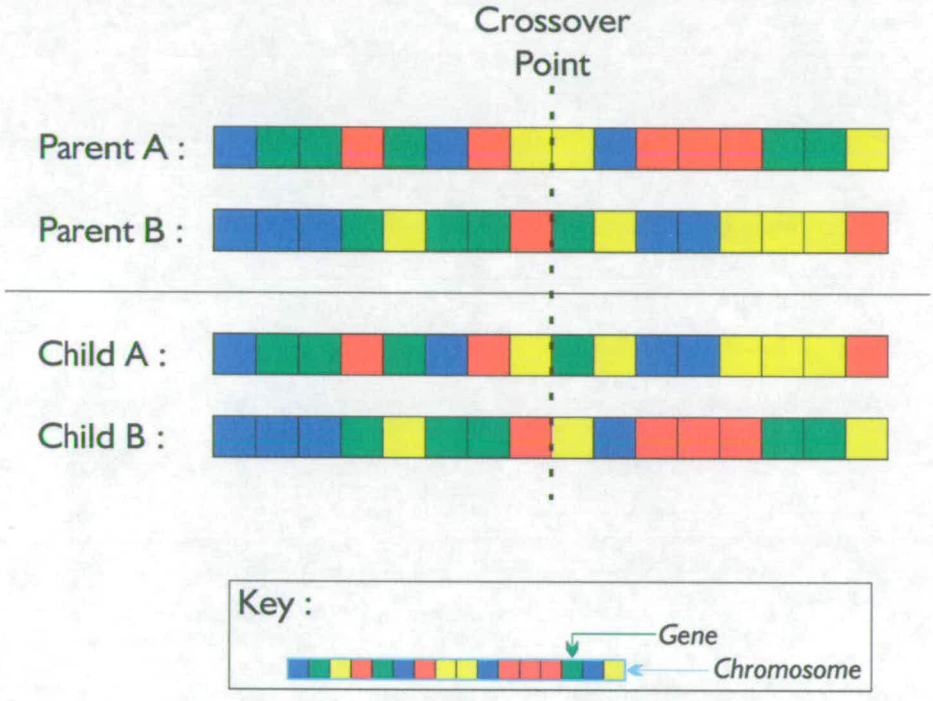


Figure 3.8: The visualisation of the crossover operator.

Finally, the process of mutation ensures that a population of chromosomes remains diverse by randomly changing genes of a few chromosomes in the new generation. If this process is not performed the population may be dominated by overzealous chromosomes causing the potential loss of useful genetic information.

In operation, an initial population of randomly generated chromosomes is created and evaluated by the fitness function with the generation of corresponding fitness values. An intermediate population is created by randomly selecting chromosomes from the current population but weighted relative to their fitness value, the fitter chromosomes having a greater chance of being selected. Finally the next generation is created by applying the crossover and mutation operators to this intermediate population. The previous generation is then discarded. This process of fitness assessment, selection, mating and mutation continues until a satisfactory solution or group of solutions has been found [175].

The main advantages and limitations of genetic algorithms are listed below [99, 146]:

Advantages:

1. Genetic algorithms are robust and effective in many different environments.
2. Genetic algorithms can search multi-dimensional spaces too large for conventional optimisation techniques to be applied.
3. Genetic algorithms consider parameter sets, not individual parameters, thus they search multi-dimensional spaces simultaneously.
4. Genetic algorithms search from a population of points, as opposed to individual points, thus operating on several solutions simultaneously.
5. Genetic algorithms use objective function information rather than derivatives or auxiliary knowledge, thus requiring virtually no problem-specific information.

Limitations:

1. Genetic algorithms require extensive computing resource and long execution times (frequently of the order of days on powerful workstations).
2. As the complexity of application domains increases, the lengths of the chromosomes also increase resulting in greater computing resource requirements.
3. Search domains must be expressible in terms of fitness functions and chromosomes.

Within the power industry, genetic algorithms have been extensively utilised; Figure 3.9 gives an historical indication of the number of publications relating to power engineering. More specifically, genetic algorithms have been used for coal purchase [125], economic dispatch [249], distribution system planning [288], load flow [286], and flexible AC transmission systems (FACTS) control design [218, 265] to name but a few. A common property all of the aforementioned applications share is a high numerical content within which genetic algorithms offer either the ability to compute such complex domains or significantly reduce computational intensity associated with traditional analysis methods.

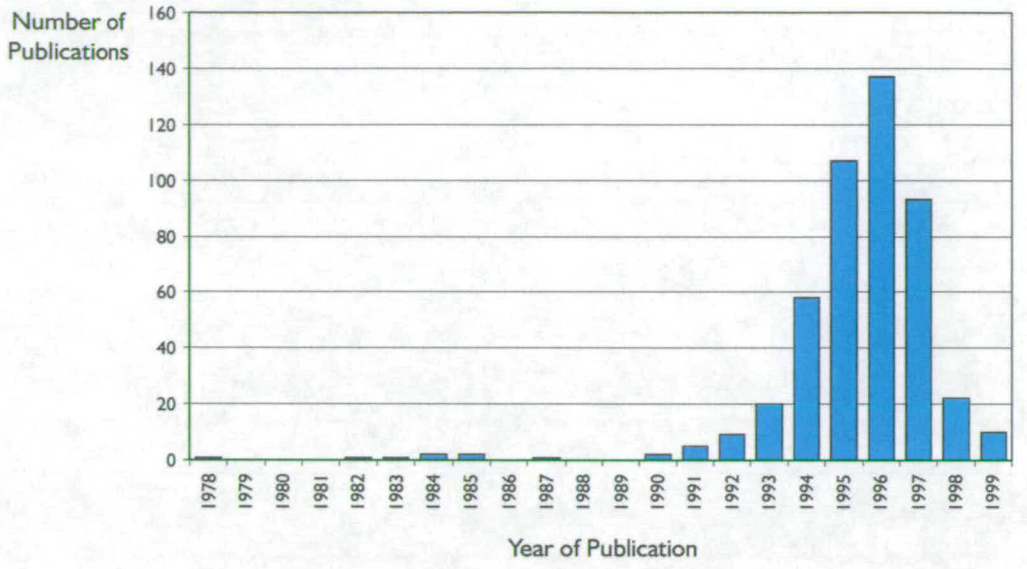


Figure 3.9: The number of genetic algorithm papers published annually in power engineering [4, p.7].

3.2.5 Intelligent Agents

One of the latest and most promising fields of research within artificial intelligence, *intelligent agents* emerged as a recognised field of research in the early 1990's. Within the artificial intelligence community there appears to be no consensus upon the precise definition of an intelligent agent, but Nwana and Ndumn cautiously venture the following [193, p.5]:

> *"... an agent is defined as referring to a component of software and/or hardware which is capable of acting exactingly in order to accomplish tasks on behalf of its user."*

Since such definitions are not definitive it is informative to describe the key attributes of agenthood where there is more agreement amongst practitioners. These attributes are eloquently summarised by Ndumu and Nwana [190] as:

Autonomy: the ability to function largely independent of human interaction,

Social ability: the ability to interact 'intelligently' and constructively with other agents and/or humans,

Responsiveness: the ability to perceive the 'environment' and respond in a timely fashion to events occurring in it,

Pro-activeness: the ability to take the initiative whenever the situation demands.

The fundamental basis of intelligent agents can be traced back to control theory of the eighteenth century, but it was not until the advent of electronic computing in the 1950's that the first steps into agent theory were taken. Research within the then newly formed field of cybernetics was attempting to unify models of control and communication phenomena, similar to those observed in animals, into a single mathematical model [183]. Also the emerging field of robotics allowed the physical construction of agents to be explored[1] [114]. The work in these fields formed the basis of a new field of research referred to as Distributed Artificial Intelligence (DAI) based upon the paradigm that collective interactions of large numbers of simple, interacting, semi-autonomous individuals (or agents) produce intelligence [159, pp.13-17]. This is a key departure from traditional artificial intelligence rationale, based upon the logical representation and manipulation of knowledge, utilising inference as a primary mechanism for intelligent reasoning.

Since intelligent agents are in their infancy in terms of theoretical and practical development, there are presently few mature industrial applications with most practical examples of agents remaining at the development stage. The main application areas are in information management relating to the internet, focussing on assisting browsing, but other tasks include scheduling of meetings, news filtering and answering questions [208, 180].

Several notable exceptions of more mature applications do exist; for example the use of agents to generate customised integrated circuit layouts [67]. One of the largest projects undertaken utilising intelligent agents has been the ARCHON (an ARchitecture for Cooperative Heterogenous ON-line systems) project which was developed between 1989 and 1994 [284]. The main application of the ARCHON project was to develop a multi-agent system for electricity management. Several systems were implemented and demonstrated, one of which was developed to assist control engineers of the English and Welsh distribution systems in fault diagnosis, security analysis, creation of safe switching schedules and collection with cross referencing of system information.

One particular aspect of intelligent agents which becomes apparent from the aforementioned applications is that, since agents themselves do not contain significant amounts of knowledge concerning the application domain, they require access to underlying knowledge systems (such as the internet) or knowledge bases (as with the ARCHON project) to operate. This is an extremely limiting factor if the application domain has yet to have knowledge encapsulated in a computer interpretable form. Furthermore the principle feature of intelligent agents is autonomy which results in a transfer of control from human to artificial reasoning. For a review of the significant implications of this transfer of control see the article by

[1]There are authors, such as Brooks [40], who suggest that the physical study of agents, *i.e.* in robot form as opposed to soft-agents, is extremely valuable if not more so.

Kitano [140].

3.2.6 Summary

The maturing of the artificial intelligence field has found an abundance of applications within the electricity supply industry. This is evident in power systems research by a broadening of the traditional analytical techniques into the fields of artificial intelligence and information technology.

The previous sections have presented the key artificially intelligent technologies and indicated some of their typical applications within power systems. This is not an exhaustive list of artificially intelligent technologies as there are other technologies, such as model based reasoning, which have not been widely adopted or utilised, or more recent developments, such as ontologies [47], which are only in their infancy. For these reasons such technologies are not included in this discussion.

The next section will explore the key programming paradigms currently available and the facilities that they offer.

3.3 Programming Paradigms

In 1835 Charles Babbage conceived, but never completed, the world's first programmable computer. The field of computing remained dormant until the 1940's when Konrad Zuse constructed the first programmable computer, and with it developed the world's first high-level programming language, called Plankalkul [143]. The first popular programming language, FORTRAN (FORmular TRANslator), was developed in 1954 and since then there has been a profusion of languages, all based upon several distinct programming paradigms.

Before the development of FORTRAN, or other so-called 'automatic programming systems', almost all programming was performed using machine or assembly language. Use of such low-level languages severely limited the complexity of programs which could be written in a machine interpretable form, and in turn limited the problems they described. High-level programming languages, such as FORTRAN, Algol, LISP and later Pascal, C and Smalltalk were developed to allow computer code to become more readily understandable, computing platform (or hardware) independent, and bug free [165].

A programming paradigm is a general purpose technique for solving problems effectively; a half way house between the real world problem being considered and

machine interpretable code. Once a problem has been resolved in a programming paradigm it may then be translated into computer code, supported in a convenient manner by a programming language. It is important to note that for the purposes of this research, a programming language does not support a programming paradigm if it takes exceptional effort to write code in the given language; the language simply enables the paradigm to be used.

The key motivation for the development of programming paradigms has been the requirement to manage complexity. In 1956 Miller discovered that the maximum number of pieces of information an individual can simultaneously comprehend is approximately seven, plus or minus two [177]. This limitation is related to short-term memory capacity, which is used extensively when programming but is not a feature of long-term memory. As the complexity of software tasks has increased, new programming paradigms have been required and subsequently developed[2]. Further discussion of this and other human cognitive factors that affect software engineering are presented by Sommerville [247].

The following sections describe the programming paradigms[3] considered of importance and relevance to this research, including a brief discussion of their foreseen advantages and limitations.

3.3.1 Procedural Programming

This is the original programming paradigm; eloquently summarised by Stroustrup [259, p. 23] as:

Decide which procedures you want;
use the best algorithms you can find.

Programming languages based upon this paradigm concentrate upon data processing and the construction of the most efficient algorithms to perform a required computation. Features typical of such languages are the creation and assignment of variables of basic types, global data, performing arithmetic, tests, loops, arrays, and functions. The structure typical of a procedural program is illustrated in Figure 3.10. Computer languages which focus upon the procedural programming paradigm are FORTRAN I, ALGOL 58 and IPLV and are referred to as the first generation of programming languages (1954 – 1958) [31, 65]. The focus on an algorithmic approach is reflected in the literature concerning these languages [241, 211].

[2]Note that management of complexity is not the only motivation for the development of programming languages; for example Sun Microsystems developed JAVA with the main motivation to create a language which was completely computing platform independent without the need of re-compiling the code.

[3]This is not intended to be an exhaustive list as there are many other significant programming paradigms. Section 3.3.4 extends this discussion.

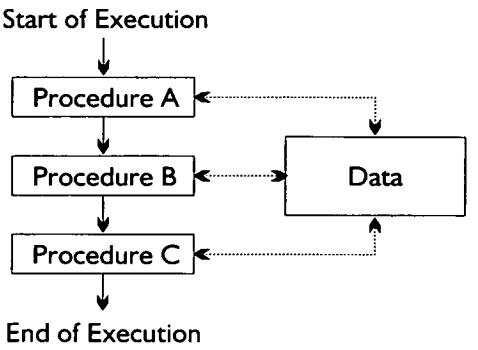


Figure 3.10: Typical structure of a procedural program [282, (Adapted)p.57].

The fundamental limitation of the procedural programming paradigm is that all functions have the same level of access to the data on which they operate. The resulting errors from incorrect code are then difficult to locate and correct. Furthermore, the inability to group or organise the program structure, combined with our limited capacity short-term memory and the limited features of such languages, crippled their capacity to encode abstract or complex concepts. These factors resulted in the main applications of such languages to be numerical processing or systems programming domains. Indeed, this is also reflected in the applications of such languages within the power systems domain, typical applications being load flow and circuit analysis.

3.3.2 Modular Programming

Initially computer programming focused upon algorithms, since the quantities of data being processed were relatively small and the algorithms themselves, compared to their modern day equivalent, uncomplicated. As data and algorithm size increased to cope with more advanced tasks and computing hardware, so too did program size and more importantly, the complexity. The emphasis in programming design hence moved from the design of algorithms or procedures towards the organisation of data. Hence the development of a *module* – a self contained unit of procedures and the data they manipulate [259, p.26]. The key features to be coded in order to create a module are:

1. The provision of a user interface for the module,
2. The data stored within the module must only be accessible via the interface,
3. The module must be initialised before being used.

The important feature of the modular programming paradigm is that the user code is separate from the code which implements the module. Therefore the user of a module does not need to know how a module is implemented, indeed the implementation may change (as a result of bug fixes or adoption of more efficient code) but the use of the module, and more importantly the code which interfaces with it, will remain unaltered. The modular programming paradigm was first introduced to languages such as FORTRAN II and ALGOL 60, referred to as second generation programming languages (1959 – 1961) [31, 65], and is now supported by nearly all modern computer programming languages.

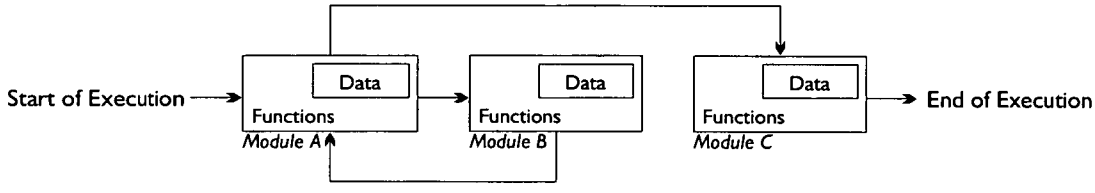


Figure 3.11: Typical structure of a modular program.

Modular programming presented a significant advancement over procedural programming. By concealing data within a module, otherwise known as the *data-hiding principle*, larger, more complex problems may be redefined into individual, ideally independent, segments of code, *i.e.* black boxes. Once created and executed together, the modules would interact to resolve a particular computational problem.

The ability to encode application domains in greater complexity utilising modular programming was rapidly exploited throughout the 1960's, complemented by an explosion of computer processing power. As the complexity of problem domains escalated it was mirrored by the scope for, and frequency of, problem redefinition resulting in the editing of modules. This situation was an improvement over having to substantially or entirely recode a procedural program although, it did frequently result in the editing of modules which had previously been de-bugged and tested. Additional editing then normally resulted in new bugs being introduced into the modules in question. This limitation was also compounded by the organisation (or classification) of modules as there are no language features to directly create, organise or manipulate such structures. Thus the modular programming paradigm created inflexible, linearly organised units of code.

The application of modular programming to the electricity supply industry reflected the changes modular programming precipitated but the applications studied remained highly numerical. An example of such an application is load flow.

3.3.3 Object Oriented Programming

The first object oriented computer language was developed by Ole-Johan Dahl and Kirsten Nygarrd between 1962 and 1967 at the Norwegian Computing Center. The language they developed was called SIMULA (SIMUlation LAnguage), originally developed for discrete event simulation but was later revised and released as a general purpose programming language. SIMULA was never widely accepted but it has significantly influenced the development of modern programming methodology, introducing concepts such as classes, objects, inheritance and dynamic binding[4] [120].

Dahl and Nygarrd realised that instead of modelling a domain as a series of abstract operations, they could represent the actual components of a system as a number of *objects*, with each object created from a specification. The specification, or *class*, of an object is defined such that multiple, identical objects may be created from any given class. In turn, another class may be derived from an existing one to represent similar types of objects. In Dahl and Nygarrd's language, each object is similar to the concept of a module in that it is self-contained with its own data and functions; in this case, however, the data and functions mirror their real world counterpart's attributes and actions respectively. Objects may also contain references to other objects, along with other properties, allowing several objects to cooperate together.

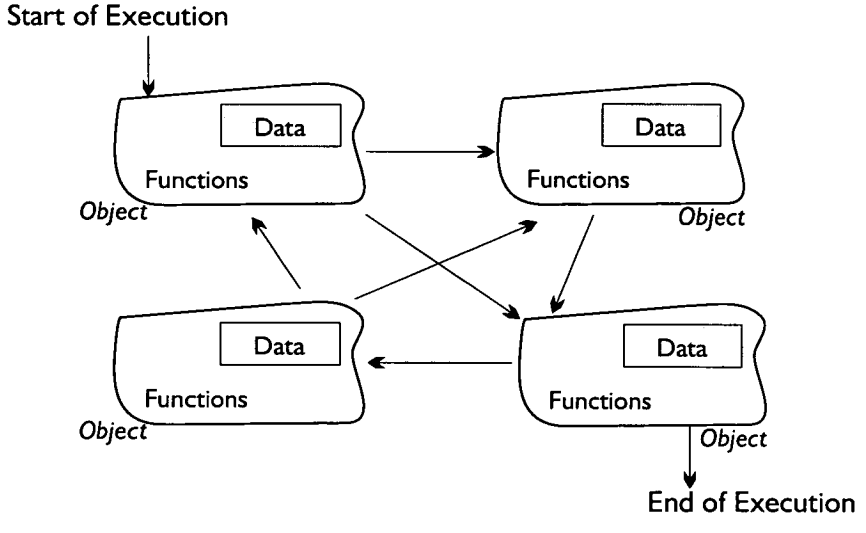


Figure 3.12: Typical structure of an object-oriented program [282, (Adapted)p.57].

Another powerful feature of SIMULA is the process of *inheritance* in which the classes are organised into a hierarchy of classes and sub-classes. Greater flexibil-

[4]These and other terms associated with object oriented programming are explained in depth in Chapter 5.

ity and re-use of code may be achieved as classes may be adapted and/or augmented without affecting the original implementation of the class. SIMULA also introduced *dynamic binding* allowing many different types of objects to respond to a common set of operations, but resulting in different operations being performed depending on which object was selected; previous to this development all function calls within a program had to be determined at compile time, a process termed *static binding*.

Throughout the 1970's research and development of programming languages was extensive and vigorous resulting in the production of many languages, examples include *Fred*, *Chaos* and *Tranquil*, but few endured [31, p.29]. However, many of the developments made during this period evolved to be incorporated into the next generation of programming languages. The fundamental concepts behind SIMULA were developed and resulted, in 1980, with the release of *Smalltalk*, the programming language which revolutionised object oriented programming.

During the same year, Bjarne Stroustrup released *C with Classes* which would, five years later, mature and develop into C++; currently one of the most popular object oriented programming languages available [258]. By 1986 object oriented programming was becoming academically and commercially accepted resulting in the first conference devoted to its discussion, OOPSLA (Object Oriented Programming Systems, Languages and Applications) [174], which marked the beginning of the hype surrounding the technology, focusing upon Smalltalk before becoming industry wide.

The paradigm of object oriented programming has been eloquently captured by Stroustrup [259, p. 39] thus:

Decide which classes you want;
provide a full set of operations for each class;
make commonality explicit by using inheritance.

The key advantages and limitations of object oriented programming are described below [31]:

Advantages:

1. Object oriented systems are more resilient to change as they are constructed from independently constructed and proven components (objects).
2. Object oriented programming allows complex software systems to evolve incrementally from smaller, previously tested systems.

3. As object oriented systems are based upon the modelling of real world objects, developers are able to capture their interaction through direct modelling of concepts they are familiar with.

4. The use of dynamic binding allows function calls from objects to have a contextual meaning allowing intelligent reasoning to be captured.

5. The interaction of groups of objects during execution allows intelligence to be exhibited.

Limitations:

1. Not all problem domains lend themselves to the object oriented paradigm; if commonality can not be found in the problem domain, object oriented programming will be of limited use.

2. The exposing, exploiting and expression of a domain in an object oriented paradigm is a non-trivial process, requiring great skill and experience.

3. Application domains expressed in the object oriented paradigm require greater computational resource than similar procedural or modular implementations (but this increase is normally regarded as marginal).

Direct application of object oriented programming to areas within the electricity supply industry has been limited, with the majority of work focusing in the field of power system load flow. Neyer *et al* [192] performed early research in this field in 1990. This research was continued by Foley and Bose [89] in 1995 with the addition of switched components allowing the dynamic load flow to be determined during simulation. While both these papers presented systems which were natively coded in a object oriented language, Phillips *et al* [209] utilised the SIMIAN (SIMulation Image and ANimation) framework, which includes object oriented databases and tools, to perform power system analysis. This hybrid approach is also mirrored by Deb [68] who utilises an object oriented expert system to the similar problem of line capacity estimation.

3.3.4 Summary

The proceeding sections have explored, in only basic detail, the key aspects of programming paradigms which are of interest as possible implementation technologies. There is a significant body of literature based upon the study of programming languages; see for example the work of Marcothy [165].

Not all programming paradigms have been discussed. For example the *goal-oriented paradigm* of PROLOG [36] or the *symbolic paradigm* of LISP [111, 283] would

have been interesting languages to explore artificial design within. Upon initial consideration, however, although PROLOG and LISP appear sufficiently expressive to warrant further investigation, the lack of Microsoft Windows binaries and integrated graphical user interface tools make them inapplicable to this work, and are hence not explored further here. The *generic programming paradigm*, which is supported by several modern languages including C++ is also not explored since there is no obvious application of such a language feature to artificial design.

3.4 Chapter Summary

This chapter has explored available artificial intelligence technologies and programming paradigms and their use with the electricity supply industry. For each technology a discussion of the advantages and limitations has been presented with a view to assessing its suitability for this project. In the ensuing chapters these technologies will be reviewed with respect to design, a selection made and implemented in a design tool for embedded generation switchgear.

Chapter 4

Selection of Software Development Environment

This chapter identifies the important issues raised from both embedded generation switchgear design and design methodology. By considering these issues, a system specification for a software development environment is presented. A review and comparison of possible implementation technologies, indicating their respective prevalence and effectiveness in performing design tasks, is given. The chapter concludes with a discussion indicating the reasoning behind the selection of the implementation technology and the software tools required to realise the embedded generation switchgear design software.

4.1 Factors Raised from Embedded Generation Experience

The implications of relatively minor mistakes committed at the design stage of an artefact may result in increased costs during the succeeding stages of development or construction.[1] The economics of embedded generation are marginal, hence the need for government intervention through the renewable obligations. Any possible reduction of such costs, especially if they can be made during the initial stages of an embedded generation project proposal, is therefore important. By capturing the key aspects of G59 combined with the design knowledge of switchgear in a computer interpretable form safe, accurate and complete specification of electrical switchgear for embedded generation schemes may be produced quickly, easily and effectively. For such a tool to be realised the following criteria must be met by the selected implementation technology:

[1] As discussed in Chapter 1.

- The ability to encode the reasoning required to design the various switchgear sections, including the internal components of each section, thereby allowing a correct and complete specification of the proposed embedded generation scheme to be produced.
- The software tool must be easy to operate and must simplify the process of designing electrical switchgear for embedded generation without limiting the user to any artificial design restrictions.

4.2 Factors Raised from Design Methodology

Design paradigms allow the characterisation and analysis of a domain to be performed thus allowing the process of design to be made more effective. However, not all the paradigms presented in Chapter 2 are suitable as a basis for developing computer interpretable design reasoning. The analysis-synthesis-evaluation (ASE) paradigm is only performed by human design practitioners. The cognitive paradigm is the study of human design reasoning and thought processes which is not directly applicable to automating design reasoning. Case based design would be difficult to capture within a computer model as domains which are suitable have no discernible design methods or rules. However, both the artificial intelligence and algorithmic paradigms are readily and directly applicable to serve as a basis upon which to develop automated design methods.

The study of design methodology yields a significant and universal characteristic common to all design methods – *evolution*. Whether it is the adoption of existing solutions to new circumstances, the direct application of iterative design or the selection of a final solution from a multitude of others, evolution manifests itself as an important aspect of design. The inherent inflexibility of certain software technologies to adapt, either during construction or execution, has already proved to be a significant limitation for modelling design processes in any domain.[2] Since design reasoning, even within narrow domains, may change over time, models of these processes must also evolve. If design domains are static, it is probable that the scope of the design tool is not. It is therefore important that the software technology selected must be sufficiently flexible not only in use but also in construction, modification and maintenance.

[2] All software technologies have limitations in their adaptability but as the discussion in Section 4.4 indicates, some technologies which initially appear particularly suitable for design tasks are inflexible when required to adapt.

4.3 System Specification

Having considered the factors raised from current embedded generation design practice and design methodology, in essence, any implementation technology should meet the following three specification criteria:

1. The software should be able to capture design methods, with reasonable coding effort, sufficient to allow the switchgear components of an embedded generation plant to be modelled.
2. The software should be sufficiently flexible to allow the modification or adaptation of the software project to aspects of the design not originally envisaged without affecting existing functionality.
3. The software must operate on a personal computer[3] in a robust and stable manner with a reasonable speed of execution; preferably utilising a standard interactive graphical user interface (GUI), such as the Microsoft Windows© operating system.

4.4 Comparison of Implementation Technologies for Design

Computional tools have been used to assist human designers since the mid 1960's, resulting in the development of computer aided design technology [263, 30]. However, the majority of the research and development in this field has focused solely upon the visual and parametric aspects of design. This section will review the software technologies described in Chapter 3 with respect to their suitability and performance for capture and reproduction of design reasoning.

4.4.1 Expert Systems

The development of design tools based upon expert system technology has been extensively investigated with the first generation of tools appearing towards the end of the 1980's. Examples of such applications include Steinberg's exploration of the design of digital integrated circuits [256] and Soh and Soh's development of a tool for producing the preliminary design details for offshore structures such as oil riggs [245], both developed in 1990. In the same year, Gupta and Siewiorek's creation of the M1, a tool for prototyping small computer systems [106], was also

[3]The specification of the personal computer being a Pentium (x586) class processor or better, operating at 120 MHz or faster with a minimum of 16 MB of RAM.

developed. Research in this area has continued throughout the last decade, allowing the development of commercial expert system design tools, such as Hammond and Davenport's *'RaPiD'* system for dental prosthesis [107], and Allen's system for enhancing the quality of printed circuit board assembly design [6].

Applications within the electricity supply industry continue to be developed and include the design of auxiliary power supply systems for thermal generation plant [97], the design of distribution networks [155, 213], integrated renewable energy systems design [212] and switched mode power supply design [161].

The predominant reasons for the adoption of expert system technology for design applications are as follows:

- Expert systems were the first commercial and widely accepted artificially intelligent technology, resulting in the extensive development of specialist expert system shells or development environments, thereby allowing rapid and robust development of applications.
- Expert systems are particularly adept at the capturing of heuristic reasoning and/or knowledge which is a common feature of many design domains.
- Expert systems allow automated decisions to be traced, thus allowing additional validation of the design reasoning by either a human operator or specialist tools.
- Specific design domains may be easily and accurately captured within an expert system knowledge base resulting in a tool that produces high quality results.

Interest in expert systems for design tasks peaked in the early 1990's with design domains being explored and reported in compendium books, indicating the large volume of research into computional design tools based upon this technology. A general review of the application of expert systems to engineering oriented design tasks was compiled by Rychener [234] who presents eleven expert systems, focusing mainly upon architectural/structural design; other areas explored include computer construction and mechanical design. Coyne *et al* [55] presents a comprehensive text on the construction, knowledge representation and operation of expert systems specifically for design tasks. Issues associated with the modelling of creative design utilising knowledge based systems was reported in detail by Gero and Maher [96].

The common element which may be traced throughout expert system based design shells is that the design domain must be heavily parametric. Within such models, numerical calculations may be performed either in the knowledge base(s)

or by external simulators, or a combination of both. Although many design domains, especially those within the engineering disciplines, are particularly suitable for modelling using expert system technology with excellent results, this technology does have limitations.

In particular, Reddy [217] indicated the brittleness and limitations of expert systems for electronic power supply design. Assuming that the complex and time consuming task of entering all the knowledge for a domain has been encoded within a knowledge base, Reddy indicated that the editing of such a knowledge base to allow adaptation or extension of the expert system required considerable effort. Reddy also indicated the necessity, and significant effort required, to maintain an expert system. This limitation has also been acknowledged by other developers in the field, for example Waterman [278, pp.196–199]. There has been research into reducing this workload of maintenance, for example by inductive learning and rule generation strategies [238] to allow such systems to adapt to a changing environment, or the use of distributed internet based tools for sourcing, modifying and storing design knowledge [223]. However, such expert systems have become excessively complicated to construct and have yet to be generally proven.

To reduce the brittleness of expert systems, Ohki *et al* [199] have extended this technology to perform electronic circuit design incorporating qualitative reasoning to present several possible circuits to the user, who then selects the preferred circuit solution. By manipulating and reasoning using parameter ranges as opposed to specific numerical values, and allowing the system to continue reasoning unevaluated conditions, Ohki *et al* have created a more flexible, robust expert system in its operation than traditional approaches. The system remains, however, application domain specific with the underlying structure of the software requiring significant maintenance.

4.4.2 Case Based Reasoning

The capturing, recall and reuse of specific experiences offered by case based reasoning is particularly relevant to design, as it is an element instantly recognisable in the human practitioner. It is therefore not surprising that a significant number of artificially intelligent design tools have been developed utilising case based reasoning, though the majority of these systems have remained at the research or prototype stage. Maher and Silva Garza [163, 164] have reviewed the use of case based reasoning specifically for design purposes, indicating the main application areas are within architectural and structural design. The authors also indicate that software development, mechanical device representation and planning tasks are other design application domains utilising this technology. Case based reasoning has also

served as a basis for exploring several models of creative design, as reported by Goel [98].

Other disciplines in which case based reasoning has been applied include electrical and mechanical engineering. For example, the design of aircraft subsystems such as flight controls, fuel systems, landing gear and electrical systems has been reported by Domeshek *et al* [76]. Rong and Lowther [228, 227] have explored the use of case based reasoning for the design of electromagnetic devices and Lees [148] has investigated the provision of quality assessment advice within the domains of mechanical component design and software development. Within the electricity supply industry the adoption of case based reasoning for design oriented tasks appears non-existent.

A review of case based reasoning applied to design tasks indicates that this area of research is still a developing discipline that has yet to mature. The fundamental issue of representing design cases has still to be formalised into a practical theory or even a set of guidelines, as indicated by Maher and Silva Garza [164, p.38]:

> *"The approaches are as numerous as there are systems, indicating that this aspect of case-based design has not matured enough to lead to general principles of how to represent a design case."*

The other two key features that comprise a case based reasoning tool, namely *case indexing* and *case adaptation*, have a variety of possible implementations. However, both these features are significantly further developed and pose no limitations to the implementation of design systems as they are algorithmic in nature. Case based reasoning design tools will produce excellent results in comparison with other artificial design tools if the following two criteria are met:

- The arrangement of system components or design factors of an artefact may be recorded, with reasonable effort, into a series of computationally interpretable cases.
- The design reasoning relating (or optimising) the arrangement of components or factors to the final artefact's operation (or performance) is not definable.

When considering an application domain where the design process is definable algorithmically, then other artificially intelligent technologies, such as expert systems, should be selected instead of case based reasoning. This is due to the fact that the collection and representation of cases is non-trivial thus making other development technologies more attractive.

4.4.3 Artificial Neural Networks

The use of artificial neural networks for artificial design has been limited. Of the few examples utilising this approach, it was Mohammed *et al* [182] who initially proposed the parametric design of magnetic cores for electromagnetic devices using neural networks. This concept of design optimisation utilising neural networks was extended by Idir *et al* [122] where neural networks are used to optimise a set of specification parameters for the design of induction motors. Idir *et al* produced the training data for the neural network to operate upon jointly from an expert system and a finite element simulation. The neural network then learned the relationship between motor performance and design specifications (or constraints).

The software tool produced by Idir *et al* performed design reasoning as follows. The principal motor design parameters are entered into the input layer of the artificial neural network, with its output layer producing the current motor performance and design constraints. The trained neural network is given an initial specification and a set of desired constraints. With the use of a feedback loop the neural network interactively executes, comparing the current constraints to the desired constraints; the resulting difference is combined with an evaluation of the performance of the current design solution and returned as an error term. The error term adjusts the design parameters on the input layer of the neural network resulting in the evaluation of a new design solution. This cycle of design, evaluate, adjust continues until an optimum solution has been found.

The fundamental limitation of utilising artificial neural networks for design tasks, as exemplified by Mohammed *et al* and Idir *et al*, is that extremely few design domains have large, parameterised, computer interpretable design data upon which the networks may be trained. Both authors relied upon synthetically generated data and the use of evaluation strategies (either by the optimisation of training data, the direct expression in empirical formulas or use of simulation tools.) The issues concerning the validity of parametric domain data generation and evaluation strategies, combined with the high computional intensity of this technique has resulted in the limited adoption of artificial neural networks as a basis for artificial design tools.[4]

4.4.4 Fuzzy Systems

Fuzzy set theory has not been applied directly and solely for the implementation of a design tool since this technology only allows the *fuzzification* of decisions, with no greater reasoning structure. It is therefore typical to combine fuzzy logic with

[4]Both the software systems reported by Mohammed *et al* and Idir *et al* have remained at the prototype stage.

other artificially intelligent technologies to produce *hybrid* systems; examples include adaptive neuro-fuzzy systems which combine artificial neural networks with fuzzy logic [186] and fuzzy expert systems [292].[5]

The use of fuzzy hybrid systems for design applications has not been extensively developed. The only example of such a hybrid system was reported by Pang who implemented an adaptive neuro-fuzzy system for control design [202]; the example presented is a control system for balancing an inverted pendulum. As previously indicated in Section 3.2.3, although hybrid fuzzy systems have been applied in the electricity supply industry, there is no evidence of their application within this sector specifically for design applications.

4.4.5 Genetic Algorithms

The use of genetic algorithms within the electricity supply industry has been intensive in recent years [4]. This trend has also been reflected in other application areas, with the field of artificial or automated design methods being no exception. The first design tools were developed in the early 1990's. Examples of such tools include Bramlette and Bouchard's research [35] into utilising genetic algorithms for supersonic aircraft design; work which was later extended by Rasheed and Hirsh [214] and complemented by their exploration of the related design domain of supersonic missile technology. Parmee and Vekeria [204] utilise genetic algorithms for the optimisation of material distribution for building construction. Deb and Goyal [69] present four mechanical engineering design domains, such as gear chain and hydrostatic thrust bearing design, resulting in components that outperform their respective counterparts produced by previously known design methods.

More recently, Carlson-Skalak *et al* [45] have developed a genetic algorithm to design a cooling system for manufacturing machinery by configuring standard components to a desired specification. This is an advancement over the aforementioned design tools as it involves the extension of design reasoning from domain specifics, or component level, to an overall integrated system level.

Specifically within the electricity supply industry, genetic algorithm based design software has been developed for several diverse application areas. Power system stabiliser design and optimum street light placement has been reported by Yeh *et al* [289]; the former has also been investigated by Baaleh and Sakk [19]. Concrete arch dam optimisation and two areas of gas turbine design, namely blade cooling geometries and turbine annulus design, has been explored by Parmee and Denham [205]. The design of three phase, surface mounted, permanent magnet motors was

[5]Other hybrid design tools based upon combinations of other software technologies are discussed in Section 4.4.9.

considered by Bianchi and Bolognani [28] which allows the optimisation of one or more objective functions, such as motor torque or efficiency, to be determined by the user before the simulation proceeds. A final example of this approach was reported by Taranto and Falcão [265] who present a tool for optimising the location of static Var compensators (SVC's) and thyristor controlled series compensator (TCSC) damping controllers to control power flow within a small power system network.

The features common to all the above genetic algorithm design tools are:

1. The design domain must be expressible as a string of numerical parameters, with algorithms that allow design solutions to be evaluated, compared and an optimal solution selected.[6]

2. The design domain must be extremely large; *e.g.* Carlson-Skalak *et al* [45, p.65] indicate that for a basic problem within their design domain, the number of possible solutions would be approximately 10^{187}.

3. The highly numerical nature of the design domain frequently overwhelms human designers.

It is also interesting to note that some authors in this field of research argue that the application of genetic algorithms to design tasks mimics creative design, as displayed by the human practitioner. The counter argument to this debate is that this technique is simply performing a parametric optimisation that has little in common with the abstract thought process behind the human act.

Even though this is a relatively new area of research, design tools based upon genetic algorithms have already been proven in many diverse application areas. As this field matures, the complexity of applications tackled using this approach and the quality of the artefacts produced will certainly increase, as already indicated by Carlson-Skalak *et al* [45]. Given a large design domain that may be described numerically, genetic algorithms will produce excellent results quickly and efficiently.

Although this section is primarily concerned with the use of genetic algorithms for design, it should be noted that the other technologies indicated in Section 3.2.4, under the general term *evolutionary computing*, have not been used specifically for design oriented tasks. The only exception to this trend has been Chen *et al* [49] who presented a novel approach to the design of feedback controllers for thyristor controlled series compensators (TCSCs) for power systems utilising simulated annealing.

[6] Design domains which meet this condition tend to be highly numerical (or parametric) in nature.

4.4.6 Intelligent Agents

The use of intelligent agents to implement design systems has been a recent development in the field of automated design methodology, thus there are currently extremely limited numbers of example applications. One of the first agent based design tools to appear was developed by Paredis and Khosla, in 1997, to produce fault tolerant manipulators for satellite docking onboard the space shuttle [203].

More recently, a design tool based upon similar but more elaborate principles than those used by Paredis and Khosla is the 'A-Design' environment developed by Campbell *et al* [44], in which the authors explore electro-mechanical design of weighing machines. The A-Design model operates based upon four subsystems which are:

1. An agent architecture that allows the creation and progressive improvement of design alternatives,
2. A conceptual representation of the design domain that may be queried by agents to obtain design goals,
3. A system of decision making that allows multiple objectives and solution retention (thus allowing for design changes), and
4. An iterative algorithm for the evaluation and improvement of basic design premises.

The A-Design system gains its robustness and flexibility from the collaboration of various agents within the framework, allowing the centralisation of data and objectives but still allowing individual specialist agents to be used for specific design goals or issues. This fundamental organisation of the design tool is analogous to how a company or consultancy operates, allowing the dynamic evolution of design tasks to be explored. It is therefore not surprising that the underlying technology of the A-Design system is based upon a whole wealth of technologies developed from the artificial intelligence community: primarily intelligent agent theory but also artificial life, genetic algorithms and simulated annealing.

The advantages of utilising intelligent agents as a technology for artificial design is discussed by Campbell *et al*, however, Lander [145] explores this arena in more detail. The central advantages are three fold and are discussed as follows:

- Design domains are characterised by the constantly evolving development of technology and ideas, resulting in software tools that model such environments becoming quickly dated. Since intelligent agents operate through well defined, specified and separate interfaces, frequently via a common overall

architectural framework, interfaces may be constructed to incorporate existing tools into an agent based design application. As the design domain continues to change, individual agents may be modified or replaced without propagating modifications through the entire design tool.

- The ability of agents to share data and process information, termed *interoperability*, allows the complex and varied flow of information within an agent based design tool to be managed, either by the architectural framework or by specialised system components. This allows the design process to be remodelled as and when required. By exploiting the relationships and information within a given design domain whenever possible, high quality design solutions may, in theory, be arrived at with minimal effort.

- Individual agents may also be coded to account for and manage conflicts which will undoubtedly arise during any design process. Typical sources of conflicts are incorrect information or knowledge, conflicting assumptions or goals between agents and differing evaluations of alternate designs. Such conflicts would normally result in a system halting, but agents may be coded to resolve such issues by classification and negotiation.

In summary, intelligent agents allow the myriad of diverse design information and methods to be modelled in an extendable and flexible environment which, if implemented using similar protocols to the internet, does not even have to exist on a common computing platform. As a tool for artificial design, intelligent agents present a wealth of possibilities, therefore it is highly probable that this technology will become a common base for artificial design development in the future.

4.4.7 Procedural & Modular Programming

The procedural programming paradigm was utilised to construct the first generation of computer aided design (CAD) systems that began to appear during the mid 1960's [30]. As these systems matured throughout the 1970's, their functionality increased substantially, resulting in a corresponding increase in the complexity of their construction. To allow developers of computional design tools to manage the code of such tools, the procedural programming paradigm was superseded by the modular programming paradigm [293, p.85].[7]

The advent of personal computers (PC's) in the 1980's resulted in a rapid expansion of computer aided design systems aimed at small businesses for computerised drafting and basic parametric design purposes [220, 43]. However, the complexity

[7]Examples of modular based programming languages utilised for the construction of computer aided design tools included FORTRAN, Pascal and C [293, p.85].

of these tools continued unabated. Therefore by the early 1990's the modular programming paradigm could not effectively manage the complexity of future computional design tools; necessitating the adoption of the object oriented programming paradigm [169].

Both procedural and modular programming paradigms have been mainly used to construct visualisation and drafting tools or to perform parametric or algorithmic design within narrow, well defined domains. Their use within the electricity supply industry, and indeed beyond to industry at large, has been extensive. However, due to their limited capability in comparison with modern artificial intelligence and programming techniques, both these paradigms are no longer directly and solely utilised to build such systems.

4.4.8 Object Oriented Programming

The use of object oriented programming for the development of design tools has mirrored the popularity and, more importantly, the availability of object oriented programming languages. One of the first systems proposed to exploit object oriented programming for design problems was 'IIICAD' (Intelligent Integrated Interactive Computer Aided Design) reported by Akman *et al* [3]. The IIICAD system was particularly advanced when it was proposed in 1987, utilising a graphical user interface and specialist software components to assist mechanical artefact design. An extremely powerful feature of the IIICAD software is its ability to communicate between the various design components via an underlying system core. This is facilitated by the integration of data base and expert system technologies. It should be noted that the entire IIICAD system was encoded using the object oriented programming language Smalltalk.

It was not until the early 1990's that other developers explored similar areas of research to Akman *et al*. Initially developers focused upon the management of design information using object oriented programming. As these systems advanced, design processes and concepts became increasingly important, resulting in their modelling and inclusion into design tools.

It should be noted that the object oriented paradigm, which is concerned with all aspects of software development, was also simultaneously developing and evolving during this period. As object oriented programming matured, a series of methodologies to facilitate the understanding of this paradigm and to assist in the realistic, practical construction of such systems developed. Examples of these methodologies, of which there are many, include the CRC (Class, Responsibilities and Collaborators) method [24], OMT[8] (Object Modelling Technique) [231], and

[8]This notation is used in the ensuing chapters of this thesis and a notational summary is presented in Appendix A.

more recently the UML (Unified Modelling Language) method [90, 232].[9] Henderson-Sellers [113] presents a complete listing of methodologies and a brief discussion of their eventual unification.[10]

The concepts proposed by Akman *et al* were developed by Barbian and Schlageter [22] who proposed a system called CODA (COoperation model for Design Applications) to support groups of human designers that would be involved in large scale tasks, as exemplified by aircraft, integrated circuit or software design. Their contribution to this area of research was made by integrating the storage and retrieval of design data to a central location combined with communication, synchronisation and coordination of information between users.

In the same year, 1993, Popieul and Anguè [210] developed what they described as a 'design helping tool', implemented in Smalltalk, to assist in the design of artefacts for manufacture utilising computer numerical controlled (CNC) machines. A similar approach, again using Smalltalk, was applied by Billington and Cox [29] for the design and analysis of control systems utilised in the water supply industry.

The development of aforementioned software assistants focused upon the management of design information and mainly parametric design processes. A more ambitious project of creating a high level design assistant (HILDA) was originally proposed by Wnuk [285] for the creation of computer systems, but it was not until four years later, in 1994, that the system was realised through a language developed by Carmichael and Mangum [46]. The language they developed was supported by the Institute of Electrical and Electronic Engineers (IEEE) in an effort to establish a common language for the communication, documentation and design of electronic systems. The methods employed in the language were a top-down design strategy with constraint propagation. Although by no means a complete set of design methods, it represented a substantial attempt to model the design of electronic systems using this technique.

The fundamental concepts described by Wnuk of attempting to relate parametric, functional and hierarchical knowledge within objects and applying constraints within the context of a design process has become an area of significant research. The extension of Wnuk's concepts has resulted in the development of design models not previously explored due to their complexity and the continued advancement of modelling design processes.

[9] A brief review of various object oriented design methodologies is presented by Daniels [65]. The review is now dated (as it was published in 1994 - before UML existed) it, however, remains a useful article.

[10] Note that object oriented methodologies frequently use the term *design* which, in this context, refers to the act of creating an objected oriented system. This should not be confused with the context presented in this section where the software systems presented attempt to emulate human design reasoning utilising the object oriented programming paradigm.

The research performed by Mitchell *et al* [179] is a prime example of the extension of the object oriented modelling. The authors have developed a three dimensional structural and visualisation model that also allows the relationships between a structure's physical form and actual function to be captured. Mitchell *et al* applied their software model to the domain of design and manufacture of gold club head prototypes.[11]

Examples of design tools for use within the electricity supply industry based solely upon object oriented technology have yet to be reported, despite the fact that this technology has been utilised within the sector, as indicated in Section 3.3.3. However, there have been several design tools which rely upon the object oriented programming paradigm but are combined with an artificially intelligent technology to capture and perform design reasoning; such systems are described in Section 4.4.9.

4.4.9 Hybrid Systems

The majority of artificial design tools, in general, utilise a key software technology in order to re-create design reasoning, normally within a narrow domain. Within this thesis such technologies have been classified as either an artificially intelligent technique or programming language paradigm. It is therefore not surprising that some researchers have selected a combination of such technologies, based upon their cooperative merits, to implement design tools. An example of this hybrid approach has already been presented in Section 4.4.4.

The combination of evolutionary computation, normally in the form of genetic algorithms, and artificial neural networks has proved constructive in the field of artificial design. An example of the operation of such a design tool would be that the neural network learns design attributes that have a high probability of producing successful solutions. From a set of design parameters, or a specification, the neural

[11] *Postscript:* During the course of this project, developments within the field of object oriented design have continued apace. The following brief discussion indicates research that was not published when the decision to use the object oriented paradigm for construction of the design tool, described hitherto in this thesis, was initially made. The following authors indicate the viability of the object oriented paradigm for computional design tool construction.

Bento *et al* [26] have extensively explored the use of logic and object oriented programming to represent the engineering design knowledge of structural engineering, focusing upon the modelling of framed building construction. The modelling of product and design processes, including the relationships between form and function have been reported by Gorti *et al* [101]. By developing their model, Gorti *et al* have also constructed a general architecture for expanding traditional object oriented design techniques; the demonstration domain described product models for television remote controls. Computional design tools for use within the electricity supply industry are limited in number. However, a notable example of such a tool implemented utilising object oriented programming is described by Sachdev *et al* [237] who presents a software application to design substation interlock schemes. A general formalism for object oriented design processes including evaluation of alternative designs, exploration of alternatives and the management of design history is presented in detail by Liang and O'Grady [149].

network will produce a design solution which is then perturbed to produce a population of possible design solutions. An evolutionary algorithm is then utilised to optimise this population of solutions.

Mohammed *et al* [181] were some of the first authors to adopt this hybrid approach for the optimal design geometry of electromagnetic devices. More recently, Grierson's research [104] on the conceptual design of bridge structures and Fan's work [82] on inverse design methods for turbo-machinery diffuser blade profiling have both exploited the combination of genetic algorithms and artificial neural networks.

Another expanding research area for artificial design is the use of object oriented programming and a variety of artificial intelligence techniques. For example Gorti and Sriram [101] have used the object oriented paradigm as the fundamental technology for their CONGEN software for exploring bridge design. However, the authors also employ object oriented versions of expert systems, case based reasoning and intelligent agent technology, all of which communicate with an object oriented database management system. Ormerod *et al* [201] utilise an object oriented database technology for storage of previous design solutions combined with intelligent agents to retrieve previous solutions and capture new solutions.

Finally, Atanackovic *et al* [18] present a software tool for the complete planning and design of power systems including sub-systems or components, such as generation, interconnections, transmission (AC and DC), distribution networks, towers and substations. The design software was implemented utilising a series of individual, specialist expert systems and numerical simulation tools, each one created to resolve particular aspects of the application domain, however, these individual tools are integrated such that the overall design goals may be achieved and optimised.

The advanced application of the artificially intelligent techniques discussed in this thesis for machine learning within various design environments was reported by Duffy [77].

4.4.10 Summary

This section has briefly indicated the use of various software technologies for the implementation of computerised design tools. From this review it is clear that a large majority of computional design tools have been produced based upon expert system technology. However the limitations of this technology have resulted in genetic algorithms and object oriented systems being developed, and more recently, intelligent agent systems. The vast majority of the design systems cited in this section have remained, unfortunately, at the prototype stage of development.

4.5 Implementation Selection

The system specification detailed in Section 4.3 lists the criteria which must be satisfied to ensure the success of the project. The resolution of the first two criteria depend upon the implementation selection; the final criterion is dependent upon the selection of the development environment which is the subject of the next section. The discussion which now follows described the reasoning behind the final and crucially important selection of an implementation technology for the project.

Expert systems initially appear to be particularly suitable for constructing a computationally based design tool, as decisions may be traced and many design tasks may be described in heuristic rule sets. However, this approach has been extensively explored, proving to be inflexible and problematic. Furthermore, the experience reported by Atanackovic *et al* [18] indicates that within the domain of electric power system planning and design, the sequence of reasoning steps performed by the object oriented expert system when solving a design problem followed an established order of execution.[12] Since the order of design reasoning underpinning Atanackovic *et al* software could be determined, the inference engine of the expert system was rendered redundant. As a result, the latterly developed stages of the project were coded using a purely object oriented approach. For these reasons this technology was rejected despite the suitability of the switchgear domain to this approach and the availability of expert system shells, *e.g.* CLIPS [50] and KappaPC [136], within which graphical user interface features could be constructed.

The lack of extensive training data for switchgear design, either in the form of a series of previous cases or in raw numerical form, excluded both case based reasoning and artificial neural networks from being considered any further. Indeed, the reliance upon synthetic or parametrically generated design data would be problematic and error prone. Also when considered with the high computational intensity of neural networks and the operation of a graphical user interface, it was unlikely that such a design tool could operate on the specified personal computer without extreme delays between design stages. Furthermore, the limited number of design tools based upon either case based reasoning or artificial neural networks indicated that both these approaches are unsuitable for the majority of design tasks, unless combined with other artificially intelligent technologies. This too was also the case for purely fuzzy based systems.

The use of genetic algorithms for exploring design domains has proved successful, although only a recent development. Indeed, genetic algorithm technology could be applied to switchgear design, with an initial population of randomly generated chromosomes representing various switchgear configurations. The fitness

[12] This is also the case for switchgear design.

function could be determined by the user which would be a symbolic representation of the required switchgear configuration. However, some cost or function minimising would also have to be incorporated into the fitness function to ensure grossly over specified designs were not produced. A hierarchy of components, or chromosome sections, would also have to be included so that switchgear sections could have precedence over, say, sets of fuses.

Such limitations of a genetic based switchgear design tool could be resolved, but at the expense of contorting the genetic algorithm paradigm. Furthermore, the switchgear design domain does not require the vast search capability that genetic algorithms offer since a final specification for a particular site may be arrived at through logical deduction. For these reasons, genetic algorithms were not considered a suitable implementation technology.

The very recent developments in intelligent agent based systems for implementing design tools have provided very encouraging results. Indeed such systems will provide the basis for many future design systems. The justification for this statement is two fold:

1. The ability of agent based technology to unify and integrate many smaller systems into a coherent, powerful design tool.
2. Design information may be stored or extracted from many diverse sources via specially constructed database interfaces or the internet.

However, agent based systems are such a recent development that their theory of construction and operation was only emerging when this project was considering the choice of implementation technology. This lack of maturity within the field was compounded by the large size of many agent based systems, requiring at least several people for their development. It was decided that an agent based approach was not appropriate for this project. Even if agent systems where applied in a 'kit form', the domain knowledge upon which the agents operate must also be encoded – a non-trivial task.

The functionality available within any purely procedural programming language will certainly allow the fundamental reasoning of any design domain to be crudely captured. Indeed most software systems could be re-coded only using procedural programming techniques, but such a task is seldom performed as it defeats the motivation of more recent programming paradigms – the management of complexity. Only the most basic design domains could be implemented using procedural programming techniques without the complexity and strategy of the design task becoming too great for the programmer to code. Therefore procedural programming was not considered as a possible implementation technology for modelling switchgear design.

The key feature underpinning the concepts of modular programming, the concept of *data hiding*, resembles strongly the practice of top-down design. Both practices require the division of a domain into resolvable, separate modules with the construction of definite interfaces, thereby allowing the individual components, once constructed, to operate in unison to produce a solution.

Modular programming presents a significant advancement over and above procedural programming techniques. Unfortunately, a major limitation of this approach is that the interaction between modules is fixed at compile time. Although initially this is not an obvious limitation, it becomes apparent as the complexity of design rationale modelled in software systems increases. Design processes adapt and evolve, however, the adaptation of module interaction during execution is not natively expressed in solely modular based programming languages. As a result, this severely limits its applicability to all but the most basic design situations.

The organisation or classification of artefacts or their constituent components is an essential process utilised by human practitioners of design. This process is also not directly reflected in modular programming languages since module organisation is basically linear. Finally, the key to design rationale is evolution; this is reflected, to a limited extent, in software languages as flexibility, but the modification of existing coded and tested modules frequently poses significant problems. Thus, due to the limitations and inflexibility of modular programming, the use of this technology to implement the project was rejected.

The design of switchgear for embedded generation may be characterised by the use of components with specific features which when combined in an appropriate logical order constitute an operational switchgear installation. On a more general level, the relationship between objects or artefacts and design processes is a central issue in design methodology.[13] Object oriented programming allows the creation of interacting objects, each with its own data and functions (or behaviour), to be captured in software directly. This ability permits the modelling of design concepts or artefacts equivalent to that which the designer would manipulate in real life. The object oriented paradigm additionally enables design reasoning to be captured in a flexible manner, thereby allowing the model to evolve with the domain. As long as the object interfaces are adhered to (they must remain unchanged, but may be further expanded to model additional behaviours), the rest of the design system will operate as previously coded.

The division of design domains into objects, either directly representing part (or a component) of the final artefact or as a conceptual aid, enables designers to efficiently manage the complexity of the design domain and this is likewise facilitated in the object oriented paradigm. Furthermore, the object oriented paradigm allows

[13] An example of such discussion is presented by Daley [64].

algorithmic design and artificial intelligent design strategies as well as the rationalisation procedures outlined in Section 2.3 to be captured in code efficiently and elegantly. If any artificially intelligent technique were required, it is very likely that an object oriented algorithm would be available, already coded via the internet, that could be directly compiled or if needs be, easily translated. Indeed, the operation of object oriented programming languages, especially the ability to define object interfaces, resembles the behaviour offered by intelligent agents.

However, it should be noted that not all design domains lend themselves to the object oriented paradigm. If no such classification or commonality can be found within the problem domain, the object oriented programming paradigm will be of limited use.[14] Indeed exposing, exploiting and expressing such commonality is a non-trivial process, even for experienced programmers, but if achieved leads to flexible, maintainable, robust software.

Given the advantages discussed above, combined with the ability of the object oriented programming paradigm to capture multi-component design rationale and general design strategies (subject to the development of a suitable framework) led to the decision that this was the most appropriate software technology currently available with which to develop an artificial design application. Within the electricity supply industry there are many possible multi-component design problems within which the development of an artificial design software application would be of use. The benefits of constructing such an application for embedded generation switchgear design are considerable. Hence its selection as the target domain.

Having selected an implementation technology, a suitable operating system and implementation language had to be selected. This discussion is the subject of the following section.

4.6 Development Platform and Language Selection

The third criterion in Section 4.3 stipulates that the development platform must be a personal computer; however, there are many operating systems available for this platform, the most popular being Microsoft© Windows. Within the electricity supply industry, and indeed throughout many other industries and universities, the Windows operating system is utilised. Therefore Windows was selected as the development platform for this project.

Microsoft Windows may be supplied in several versions, the most popular being version 95, which has been superseded by version 98SE and will shortly be

[14] This is not the case for embedded generation switchgear design.

superseded again by the Millennium Edition (ME).[15] Therefore it was decided that any compiled code should operate on the Windows 9x operating system. However, the Windows 9x series of operating systems is not ideally suited for programming development due to its instability.[16] The Microsoft Corporation fortunately produces a stable and enhanced operating system that also includes the functionality of Windows 9x series, called Windows NT [246].[17] For this reason Windows NT, version 4.00, was selected as the development platform for the project.

Having selected an operating system the final decision to be made was the choice of compiler. This was dictated by the choice of programming language. There are many object oriented programming languages available for Windows NT, the most widely used languages being C++ [259], Delphi (object oriented Pascal) [70] and Java [128]. The widespread use of C++ and the existence of compilers for a large variety of platforms (should the code ever need to be exported), combined with the maturity, speed and power of the language made C++ the logical choice of implementation language.

It should be noted that the choice of C++ as the implementation language does not, by default, include libraries or tools for the construction of graphical user interface features used within the Windows operating system, such as menus, dialogues and icons. To construct such features a specialist Windows based compiler is required. At the time of compiler selection for this project, there existed two C++ compilers that operate with a *visual* environment; Microsoft's Visual Studio [260] and Borland/Inprise's C++ Builder [42].[18] The visual functionality of both compilers refers to their ability to construct all the visual items required within the Windows environment, *e.g.* menus, icons and dialogues, using similar graphical tools as opposed to having to code these items in a text based language, then compiling this code to render the final graphical user interface for the application in question. Since the design tool must be easy to use, this visual aspect of both compilers is a very important feature.

Both Visual Studio and C++ Builder compilers were technically and functionally suitable for the project but it was decided that Visual Studio, with its intuitive manner of operation, extensive libraries, on-line help and enormous user base, was the most appropriate solution.

[15]This family of Microsoft operating systems will hence forth be collectively referred to as 'Windows 9x'.

[16]When programming it is easy to produce code that is syntactically correct but logically flawed. Code of this nature will therefore compile without error but when executed will result in an the operating system attempting to perform an illegal operation. Such illegal operations will frequently result in Windows 9x *crashing*; that is, the computer becoming completely inoperable. The computer must then be switched off and re-started. Other operating systems for personal computers, such as Sun Microsystems© UNIX, any distribution of Linux or Microsoft's Windows NT, are constructed so that illegal instructions will result in the user being informed of an error but the system remains usable, hence these operating systems are described as 'stable'.

[17]NT stands for New Technology.

[18]A review of both compilers is reported by Anderson [10].

4.7 Chapter Summary

This chapter highlighted the main requirements any implementation technology had to fulfil from embedded generation switchgear and design methodology perspectives. These requirements were characterised into a system specification before a comprehensive review of computional based design tools was presented, categorised by implementation technology. Based upon this review a discussion of which implementation technology should be selected was presented; the final decision being the object oriented paradigm. Finally, a suitable development platform, Microsoft Windows NT, and an object oriented programming language, C++, were selected to implement the project which led to the Microsoft Visual Studio compiler being selected.

Having decided upon the software apparatus with which to construct the embedded generation switchgear design tool, the next chapter describes how the rationalised design process may be encoded and captured using the object oriented programming language C++.

Chapter 5

Software Methodology

Having selected the object oriented paradigm as the most appropriate implementation technology, it is the purpose of this chapter to discuss how the switchgear design software is implemented utilising this programming technique. This chapter begins by outlining the object oriented programming approach, its implementation and the underlying language mechanisms that allow this programming technique to function. The discussion then proceeds to the application of the object oriented paradigm to the problem domain, namely switchgear design. This section indicates how the software technique was applied in order to create the design tool, as well as outlining some of the difficulties associated with the applications development. The chapter concludes by summarising the overall structure of the software.

5.1 Object Oriented Programming

The purpose of this section is to explain the essential aspects of the C++ programming language, in particular, the mechanisms that allow object oriented systems to function. The C++ language features presented in this section have been discussed in detail, from a variety of perspectives, by numerous authors [282, 74, 144, 206, 121]. The definitive guide to the C++ language is, however, given by Stroustrup [259], who also created the language. In November 1997, the International Standards Organisation (ISO) ratified the third edition of the language to form the C++ *Standard*. It should be noted that the mechanisms discussed in this section are not specific to the C++ language but are common to all object oriented programming languages. However, the language commands detailed in the text to invoke these mechanisms are specific to C++. To enhance the clarity of the text, all language commands or references to class names, member functions or member variables are highlighted in the text by the use of a monospaced font; *e.g.* `virtual` or `CTransformer`.

5.1.1 The Object Oriented Approach

The object oriented paradigm is fundamentally concerned with the representation or modelling, in software, of real world artefacts within a particular problem domain.[1] In the real world, people are surrounded by objects, both actual and imaginary, that represent the various aspects of the world in which they exist. Such objects may be recognised by their identity and behaviour; typical examples include:

- Physical objects, *e.g.* a car, a tree or a telephone,
- Human entities, *e.g.* an employee, a customer or an organisation,
- Incidents (relating to time), *e.g.* a transaction, a journey or a meeting,
- Interactions (links between objects), *e.g.* an electrical connection, a legal implication or a geographical location,
- Inclusion (definition of sets of objects), *e.g.* an inventory, a directory or a specification.

Therefore the fundamental philosophy underlying the object oriented paradigm is that the world consists of interacting, classifiable and identifiable objects. These objects should be modelled in an object oriented software system to reflect their real world counterparts as closely as possible, in both design and operation [248].

5.1.2 Object Definition and Creation

The key to object oriented programming is the hierarchical organisation of *classes*, their use to create objects and the resulting behaviour of these interacting objects. To simplify and assist their understanding and management of the world around them, people tend to group artefacts with similar attributes together under a category name. For example, Pelton, Francis and Kaplan are all types of hydraulic turbine. Within an object oriented program, a `class` provides a description for creating objects. Indeed, a single class may be used to produce any number of identical objects.[2] Individual objects are defined based upon the construction details of the class through a process termed *instantiation*.[3]

Classes are organised into hierarchies where abstract, common features of a group of similar objects are modelled centrally at the top of a hierarchy by *abstract*

[1] As discussed in Section 3.3.3.

[2] This is the case as long as there is sufficient, unallocated memory within the computer.

[3] The term *instantiation* arises due to the fact that an object is a real example or an *instance* of the class. It should also be noted that the terms *object* and *instance* are interchangeable.

classes. Progressing down through a class hierarchy, the general features of abstract classes are reused, through a process called *inheritance*, and refined; with additional new features included to yield detailed, specific class specifications – called *concrete classes*. Concrete classes differ from abstract classes as they have sufficient definition to instantiate objects that can be utilised to perform practical tasks. Conversely, abstract classes are generalised and lack such detailed definition; they are utilised only to create and organise class hierarchies, hence their name.

To illustrate the use of abstract and concrete classes, consider an object oriented simulation tool for hydraulic turbines. An abstract class called `HydraulicTurbine` would define all the common features associated with such machines. For example all turbines have a power rating, a maximum flow rate and physical dimensions. However, such a class can not define an object called, say, `a_HydraulicTurbine` as it would be meaningless; it would be a generalised, poorly featured object. If the class `HydraulicTurbine` formed the basic model of such machines, through the process of *inheritance*, the concrete classes `Pelton`, `Francis` and `Kaplan` could be developed which would include the features common to all turbines with additional details specific to each type of turbine, sufficient to make the objects created from these classes realisable and purposeful.

5.1.3 Object Implementation

Objects developed based upon the object oriented programming paradigm are composed as follows:

$$\text{Object} = (\text{private})\text{Data} + (\text{public})\text{Processes}$$

where the data segment stores the particular attributes and current state (and therefore the unique identity) of the object, referred to as *member variables*, and the processes define how the object behaves to external actions, termed *member functions*. The terms in brackets indicate the access other instances have to the object's implementation details; they are referred to as *access operators*.

Ideally, all member variables within an object should be declared as `private`, therefore denying direct access to the object's data by other objects. The hiding of data in this manner is crucial to the object oriented paradigm. By employing this technique, referred to as *encapsulation*, improved security and reliability of the resulting software is achieved. However, in exceptional circumstances such private data may have to be declared `public`, thereby allowing other instances direct access to an object's data; thus breaking the object's encapsulation.

Objects have the ability to respond to messages (implemented as function calls) from other objects within an application. Such interaction is provided by the `public` member functions that define the interface between the object and the rest of the application instances; this arrangement is illustrated in Figure 5.1. Note that any member function may additionally be defined as either `protected` or `private`.[4] A summary of access operators available in C++ and their effect upon access to member functions and variables is depicted in Figure 5.2.

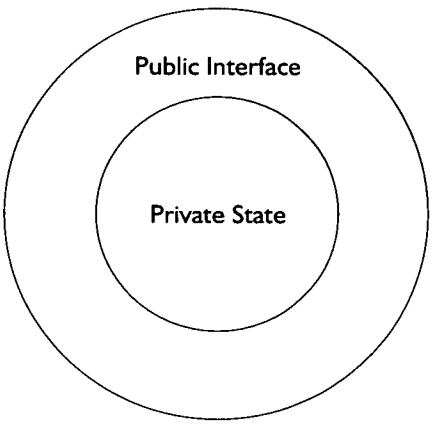


Figure 5.1: A schematic representation of an object [206, p.54].

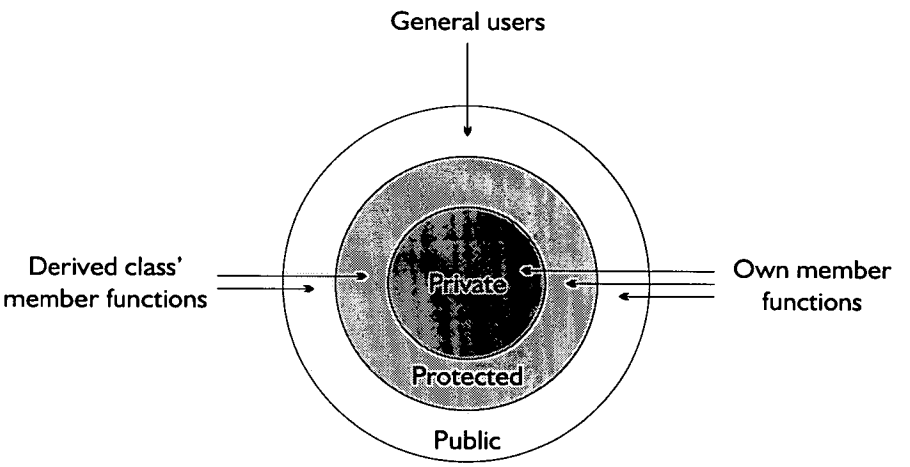


Figure 5.2: A illustration of access operators within C++ [259, p.402 (Adapted)].

An object oriented application performs its purpose by creating sets of interacting objects from classes. During execution, a series of three important events take place that enable the interaction of objects to occur thereby allowing the software application to function; they are as follows:

[4]It is important to note that due to space limitations a complete discussion concerning access operators within C++ has not be given, especially with respect to the `protected` and `friend` declarations. For a more complete discussion on their use refer to Stroustrup [259, pp.402-407 & pp.278-282].

1. As an object oriented program begins execution, several fundamental objects are created that comprise the application's basic functionality; additional objects are created, as required, during the programs operation.

2. As the program proceeds, it is the interaction between objects through messages (or member function calls) that allows the program to fulfil its task.

3. During execution, if objects are no longer required, they are deleted and the memory they occupied is reclaimed. Once the task is complete, the program ceases to execute and all application objects are deleted and the memory reclaimed.

5.1.4 Object Oriented Terminology

The object oriented paradigm is founded upon three powerful concepts that allow the technology to function. These concepts are *abstraction/encapsulation, inheritance* and *polymorphism;* all of which are concerned with the formation of, and relationships between, classes. What these concepts are and how they manifest themselves, with respect to the C++ language syntax, is briefly explained and discussed below.

5.1.4.1 Abstraction/Encapsulation

Abstraction is the term given to the process of creating a self-contained model, an object, that includes all the essential features, states and behaviours required for its operation. Objects are constructed from a combination of basic data types,[5] called *member variables,* to create what is termed an *abstract data type* that defines how the object will be represented in the computer's memory. Such abstract data structures are also encapsulated with a set of operations, referred to as *member functions* that define how the object interacts, or behaves, within the software application.

The term *encapsulation* is strongly associated with the process of abstraction. Encapsulation refers to the practice of ensuring that an object is independent, self-contained and that the implementation details of the object are completely hidden from all other instances in the system. Therefore the internal state of an object can not be directly modified from outside the objects interface. If any modification is required, for example to store a number, it should be instigated by calling the appropriate public member function from the object's interface. Although this initially appears to be a convoluted approach, as opposed to storing the said number directly in a global variable, such encapsulation ensures that any changes to member

[5]Examples of data types include integers, strings (of characters or numbers) and floating point numbers.

variables are appropriately type checked, validated and, if required, within predetermined value boundaries. Furthermore, modifications to the type representation of such variables may be made without affecting the external interface of the object. Therefore the coding or operation of other objects that constitute the system will remain unaffected.

5.1.4.2 Inheritance

The process of *inheritance* allows one class to inherit the structure and behaviour of another class as if it were part of it's own implementation. The process of inheritance is not limited to a single relationship between *superclass* (or parent) and *subclass* (or child); a subclass may inherit from one or more superclasses through a mechanism termed *multiple inheritance*. The inheritance process fulfills two important, complementary roles:

1. Inheritance permits the inclusion of generality to be captured in software.[6]
2. Inheritance allows 'families' of similar objects to share their common data types and behaviours.[7]

Both the above roles are graphically illustrated in Figure 5.3.

5.1.4.3 Polymorphism

The term *polymorphism* means *'having many forms'*; this is reflected directly in how this mechanism operates within the object oriented paradigm. Polymorphism allows different objects to respond to the same member function call (or message) in different ways, thereby allowing specific behaviour to be associated with each type of object. The implementation details of how each object deals with a request is left to the receiving object.

In practice, polymorphism is performed within the C++ language through two mechanisms, namely *virtual functions* and *function overloading*. Virtual functions are distinguished from normal member functions by the placement of the keyword `virtual` in front of the function definition. Apart from this keyword addition, the function is defined and implemented in a similar manner to any other standard

[6]Traditional software techniques, such as procedural or modular based programming languages, utilise general models and transform them into specific solutions. In doing so, such specific solutions lose the generality of the problem domain resulting in software that is difficult to modify or extend. Object oriented software, on the other hand, includes the burden of generality by expressing it through inheritance, resulting in software that may be easily restructured or adapted [65].

[7]By using the inheritance mechanism, programmers have only to code the differences that exist between subclass and superclass to extend the functionality of existing software.

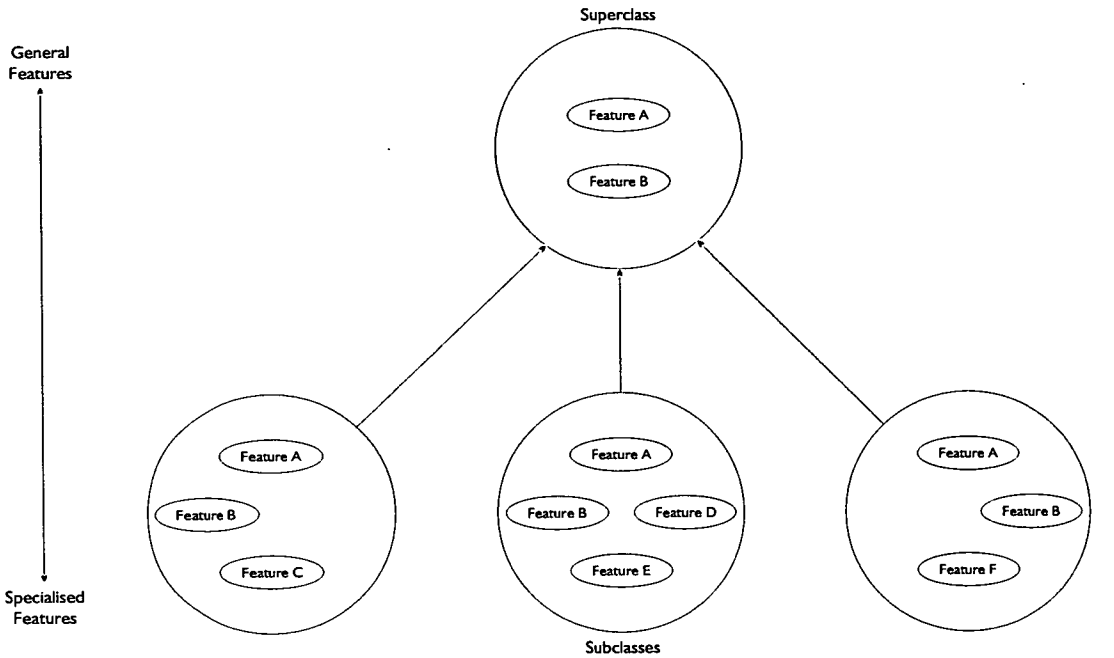


Figure 5.3: Diagram of inheritance [144, p.13(Adapted)].

function. Subclasses which inherit from superclasses that contain virtual functions may re-define the implementation of any virtual function within their own class definition resulting in a modification of their behaviour to that function call.

Function overloading allows the symbolic names of functions to be used repeatedly throughout a series of class definitions, or even within a single class as long as the argument list for each function definition is unique. Therefore when an overloaded member function is called, it is the arguments provided by the calling object that will dictate which implementation of the said function will be executed. By allowing several functions to bear an identical symbolic name, an operation common to a hierarchy of similar classes may bear the same symbolic name, allowing such general operations to be expressed elegantly.[8]

Both the virtual function and function overloading mechanisms permit the decision of which function implementation will be executed to be deferred until runtime through a compiler technology referred to as *dynamic binding*.

5.1.4.4 Aggregation

The three concepts of abstraction/encapsulation, inheritance and polymorphism apply to the construction and relationships between classes. However, *aggregation*

[8]This is in contrast to procedural or modular languages within which each function definition must be given a unique symbolic name so that all function calls may be pre-determined at compile time. Such programming languages are based upon a compiler technology known as *static binding*.

relationships apply only to objects but constitute another important aspect of the object oriented paradigm. The aggregation mechanism allows objects of one class to be composed of objects of another class. Through aggregation, composition relationships may be represented, *e.g.* x is part of y. The aggregation mechanism is a powerful feature of object oriented software as it allows structures of objects to be easily reorganised. Furthermore, if during the construction of an object oriented program, the developer has a choice between utilising either the inheritance or aggregation mechanisms, it is recognised that aggregation relationships should be selected due to their inherent flexibility [92].

5.1.4.5 Summary

Through the aforementioned mechanisms of the object oriented programming paradigm, allows greater levels of complexity and flexibility to be captured in software. However, it is the reuse of existing code that is one of the most powerful features of object oriented software. By the use of encapsulation, abstract data types may be created as and when required. The redefinition, and thus reuse, of existing classes into new classes is achieved through inheritance. Symbolic names may be reused by utilising polymorphism and in doing so different object behaviours may also be included. Finally, aggregation allows existing classes to be reused as components to construct larger, increasingly complex objects. It is this ability to reuse code that permits object oriented software to be extremely flexible. This flexibility allows the code's shelf life to be extended and code maintenance to be easily managed.

5.2 The Development Environment

Before proceeding to explain the development and construction associated with the project software, the important role played by, and the implications associated with, utilising the Microsoft development environment should be noted. As detailed in Section 4.6, the development environment selected with which to implement the project design software was Developer Studio, a C++ compiler that comes supplied with extensive code libraries and application development tools that enable complete, functional Windows applications to be created using C++ code.

Microsoft Windows was originally coded in a procedural programming language in which an interface was developed between the Windows operating system and Windows applications. This interface was called the Windows Application Programming Interface (API), through which applications could access a large set of functions[9] that allow the application to execute within the Windows operat-

[9]There are over a thousand functions that constitute the Windows API.

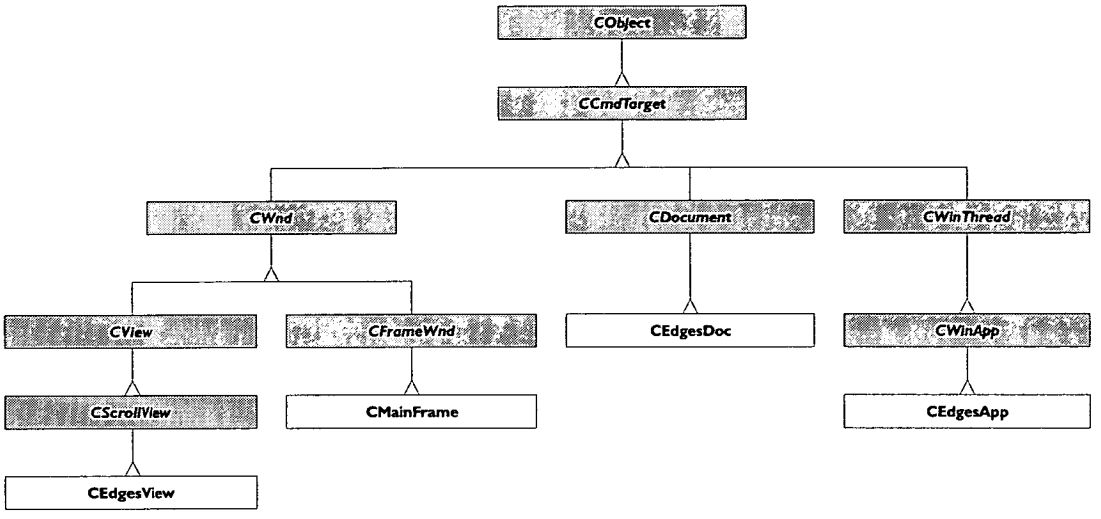


Figure 5.4: The Microsoft Foundation Class hierarchy for Edges.

ing system. In order to facilitate the object oriented programming paradigm, the API was encapsulated into a class library called the Microsoft Foundation Classes (MFC) [176].

In addition to allowing object oriented Windows programs to be written, the MFC provides an *application framework* that defines the structure of a program, based upon the *Document/View concept*. The key feature of the Document/View program architecture is to separate the document, a convenient term for the application's data, from the user's view of the document. This arrangement has several advantages, the most obvious being that a single data set, such as a list of numerical figures, may be viewed as a table or as a chart [142]. By adopting the MFC framework and the Document/View architecture for an application, four fundamental objects are required by the MFC framework to create the basic application.[10] The software developed in this project utilises the MFC framework and the Document/View architecture; named *Edges*, an acronym for Embedded Design Generation Emulation Software, the class diagram for the application is illustrated in Figure 5.4. The diagrammatic notation utilised throughout this chapter is explained in Appendix A. Each of these four classes and their function are explained below [84, 121]:[11]

`CEdgesApp` The purpose of this class is to represent the overall application. Therefore the responsibilities of the class include dispatching messages (generated mouse or keyboard activity) to the various windows and dialogs that con-

[10]Technically the application referred to here is a single document interface (SDI) application framework.

[11]It should be noted that within MFC all class names are prefixed with a capital 'C' to denote that the symbolic name represents a class definition; this naming convention has been adopted throughout this project.

stitute the application and the management of memory and application subprocesses (in so doing, keeping the application running).

CMainFrame It is the responsibility of this class to represent the main frame of the application window; more specifically, the title bar, menu bar and border of the application window (refer to Figure 6.1).

CEdgesView This class defines how the data contained within `CEdgesDoc` will be displayed to the user in the client area of the application window (refer to Figure 6.1) created by `CMainFrame`.

CEdgesDoc The purpose of this class is to provide a location and archiving functions for the application data. All objects created during the course of the execution of the application will be stored within this class.

Of these four classes, it is the `CEdgesDoc` and `CEdgesView` classes that will be of most interest as it is through these classes that the objects which perform switchgear design are stored and viewed within the application.

5.3 Development of the Model

The previous sections presented the features and mechanisms that allow object oriented software to operate and the features of Microsoft Foundation Classes that ease the programming burden of developing a Windows application. The discussion in this section focuses upon the construction of the switchgear design software that aims to meet or exceed the criteria described in Section 4.3.

The development of any object oriented software application requires the resolution of non-trivial architectural software design problems; such issues include:

- Which artefacts, either real or symbolic, constitute, are appropriate or are necessary to be included within the model?
- Which particular aspects, or features, of the artefacts selected should be included in the model?
- How should the artefacts selected be organised within a class hierarchy?
- How should the objects created from class definitions be arranged and stored in the computer's memory; how should they interact?

Such questions have posed significant problems for software developers. As a result, object oriented *design methodologies* have been developed to assist developers

resolve these issues. If applied intelligently, design methodologies will yield solutions that will not only resolve the initial problem but will also provide extendable, flexible software solutions.

This section will consider how such object oriented design methodologies were applied to develop the switchgear design tool, the problems that arose during its construction and the architecture of the various classes that together constitute the software. It should be noted that not every feature of every class has been included in the following discussion as this would extend the discussion immeasurably and bury the important aspects of the description under a wealth of relatively minor points of detail.

5.3.1 Commencing the Modelling Process

As a key architect of object oriented design methodology, Booch [31] recommends considering the problem domain; the nouns that constitute the domain should serve as the basis of class definitions and the verbs within the domain should constitute the actions of these classes. The organisation and responsibilities of each artefact modelled should be reflected in the software as closely as possible through the hierarchy and interaction of objects. Adhering to this recommendation for embedded generation switchgear design, the initial noun that appears is *switchboard*. Each switchboard contains a series of two or more *sections*, with each section fulfilling a particular function, as discussed in Section 2.1.2, by the inclusion and arrangement of specific constituent components within each section. Therefore each of these artefacts, specifically, the switchboard, the various section types and constituent components, will be individually modelled by a class definition. The arrangement of objects created by these class definitions will mirror the arrangement of these components in an operational switchgear installation.

A switchgear installation consists of one or more switchboards, connected to a source of electrical energy; either to the distribution network or to one or more embedded generators. However, since the function of each switchboard is to distribute electrical energy to one or more on-site loads or, in the case for a single or group of embedded generators with no local load, to the local distribution network, it was decided that individual switchboards need only be modelled during a design scenario. This practice is also in accordance with current switchgear design practice.

The responsibilities of the class `CSwitchboard` must reflect the attributes and features of a standard switchboard installation, which are:

- To store all the attributes common to the entire switchboard and provide access to these attributes via an interface.[12]
- To provide storage and a manipulation interface for an indefinite number of switchgear sections.

The attributes common to the entire switchboard include climatic conditions (altitude, humidity and temperature), electrical definitions (service voltage and frequency, busbar current rating and prospective fault ratings) and general information (switchgear manufacturer, horizontal or vertical isolation, inclusion of labelling, location details and applicable safety standards). Each of these attributes is stored in a private member variable with public member functions to provide and regulate access to these attributes. This arrangement ensures that the `CSwitchboard` object is completely encapsulated by its interface. This approach of encapsulating member variables has been adopted throughout the project software. The storage of individual switchboard sections is also the responsibility of the `CSwitchboard` class, and is achieved by creating a dynamic array that stores `CSection` objects as they are created with the provision of a set of manipulation functions that allow the instantiation, editing, drawing, and printing of all section objects.

To enable the `CSwitchboard` class access to the facilities offered by the Microsoft Foundation Classes framework, it must include within its class definition a series of appropriate member functions and variables. This is achieved by allowing the `CSwitchboard` class to inherit the required interface from the MFC base object, a class specifically created for this task, called `CObject`. Indeed, this is the case for all objects designed within Edges. Since a `CSwitchboard` object will be required for every design scenario performed using the software, a public `m_switchboard`[13] object is included as part of the document class of the application. The relationships between `CSwitchboard` and the MFC framework are illustrated in Figure 5.5.

Every switchboard contains two or more sections, with each section fulfilling a distinct purpose.[14] As there are six functional types of section, an abstract class called `CSection` was defined to allow the features common to all sections to be described within its definition. These common properties may then be shared and included with functionally specific classes that model the operation of the various section types, through the mechanism of inheritance. The resulting class hierarchy is depicted in Figure 5.6.

[12]This arrangement ensures that all sections are consistently specified and that the duplicate storage of such common design data is avoided.

[13]The notation of placing 'm_' in front of a class data member is standard throughout the MFC framework. It has been adopted in this project to assist in identifying class member variables.

[14]Refer to Section 2.1.3 for details.

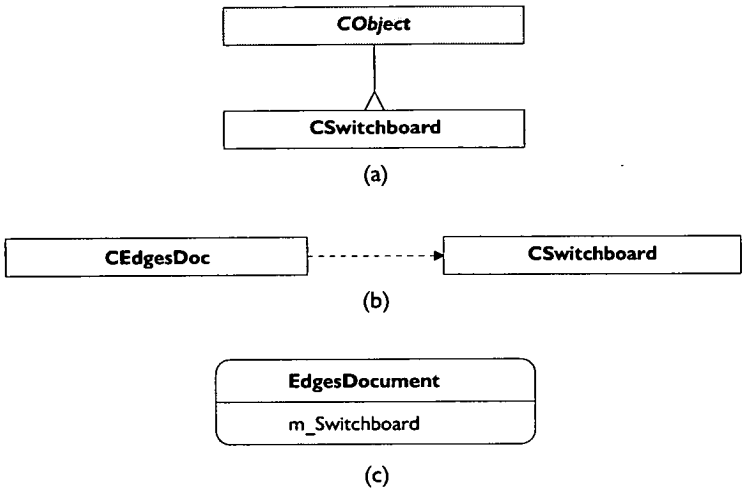


Figure 5.5: The relationships between `CSwitchboard` and the MFC framework; (a,b) class diagrams, (c) object diagram.

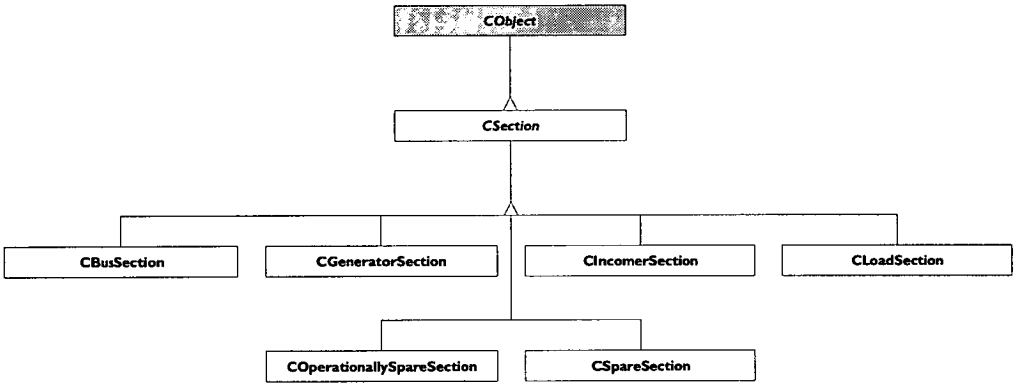


Figure 5.6: The initial class hierarchy for switchgear sections.

All sections share several common properties with the switchboard in which they are contained; as previously alluded to, these properties are shared with other `CSection` objects through the use of the aggregation mechanism as opposed to the inheritance mechanism. The reasoning for electing to utilise aggregation in favour of inheritance for the modelling of sections is three fold; it allows:

1. The realistic representation of the relationships between switchboards and sections,[15]
2. That the responsibilities of a switchboard and a section are completely separate and independent, and,
3. The resulting software architecture allows greater flexibility to be achieved.

[15] A switchboard is comprised of a number of sections; however, if inheritance was utilised it would imply that each switchboard is a generalisation of a section, a conclusion which is not rational in this context.

Therefore the class relationship between `CSwitchboard` and `CSection` classes is illustrated in Figure 5.7.

Figure 5.7: The aggregation relationship between `CSwitchboard` and `CSection` classes.

Further consideration of the responsibilities of the various section types yielded that both `CGeneratorSection` and `CLoadSection` classes may also be defined as abstract classes since both are generalised section definitions. There are several different types of generator and load sections which have a direct impact upon the electrical specification of these sections and to a lesser extent upon the overall switchboard specification. For example, synchronous generators require synchronisation protection relays, as opposed to induction generators where such equipment is unnecessary. Furthermore, both generators and loads have substantially differing fault contributions which must be considered to ensure the total fault level of connected equipment does not exceed the maximum fault rating of the switchboard or utility connection.

Therefore by modelling generator and load sections in greater detail, programming only the differences between section types utilising inheritance and polymorphism mechanisms, the accurate inclusion of the various section types and their respective behaviours can be captured within the software model. The features common to all sections conform to an identical interface thereby allowing these different objects to be easily and uniformly managed. This revised arrangement is illustrated in Figure 5.8. Furthermore, through this hierarchy of section types the first important stages of design reasoning are captured within the software, as by selecting to create a particular section type the appropriate protection and instrumentation equipment may be automatically allocated.

The operation of a switchboard relies upon the existence and correct operation of both external connections and constituent components. The following two sections considers both these aspects of switchgear installations respectively and how they may be incorporated into the object oriented design model.

5.3.2 External Connections

A switchboard requires several important electrical connections to be made to allow it to operate. These electrical connections consist of one or more connections to a source of energy (either via the distribution network and/or to an on-site embedded generator) and connections to auxiliary power supplies. Load sections are optional except when no utility connection is present, in which case, one or more

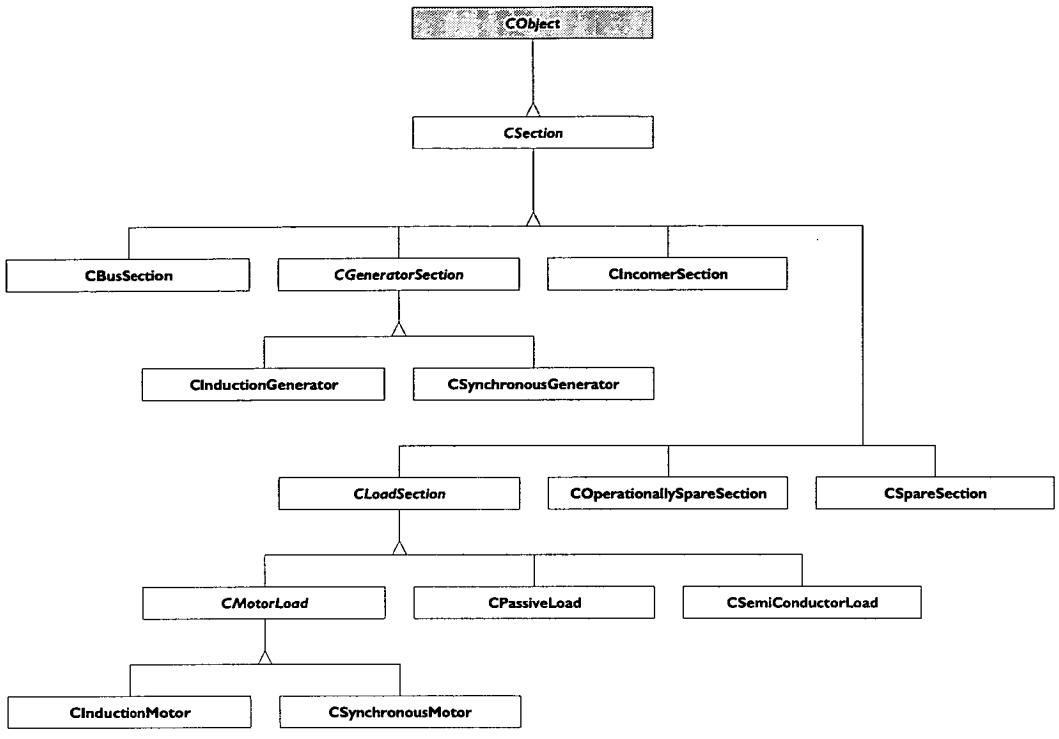


Figure 5.8: The expanded class hierarchy for switchboard sections.

load sections must be included within the switchboard design. The connections to the distribution network, generators and loads are already represented within the model by the hierarchy of various section types, depicted in Figure 5.8. However, auxiliary power supplies are a completely separate entity, sharing no general attributes or responsibilities with any of the various section types or the switchboard itself. Therefore auxiliary supplies are modelled as a completely separate class.

As indicated in Section 2.1.3.6, it is the responsibility of the auxiliary supplies to provide electrical power to the switchboard. This responsibility will be included within the class definition of any auxiliary power objects, along side the monitoring of the number and wattage of the loads connected. Therefore, it would appear logical to perform Booch analysis as previously applied to switchboard sections to yield the class hierarchy illustrated in Figure 5.9.

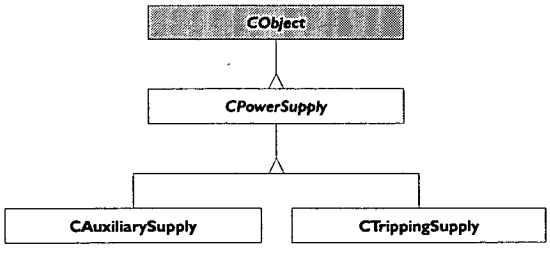


Figure 5.9: An initial class hierarchy to represent switchgear auxiliary supplies.

However, both supplies differ in only minor respects; their names, the type of electrical power produced and whether a secure supply is required; otherwise their responsibilities and functionality are identical. Thus the class hierarchy represented in Figure 5.9 appears unnecessarily complex, as such differences can be accommodated by providing arguments, *e.g.* supply name, voltage, current type and backup required, to the `constructor` member function of the class.[16] By utilising this technique as opposed to the inheritance mechanism, the class hierarchy illustrated in Figure 5.9 may be simplified to one concrete object, `CPowerSupply`, as shown in Figure 5.10.

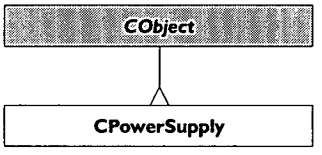


Figure 5.10: The class hierarchy for auxiliary power supplies.

Having decided upon the responsibilities and class definition sufficient to represent auxiliary power supplies, the relationships between `CPowerSupply` and the other classes in the model must be considered. The majority of new switchboard installations will require both an auxiliary supply and a tripping supply. However, in certain design cases one or both supplies may be omitted from the final switchboard specification.[17] Thus, there is a choice of including auxiliary supplies either by default (therefore instantiate two `CPowerSupply` objects and include member functions to either include or exclude each supply from the final specification dependant upon the users requirements), or instantiate them as required.[18]

Experimentation with both arrangements, default instantiation and on demand instantiation of `CPowerSupply` objects, yielded that including two initialised `CPowerSupply` objects by default was the most elegant solution. This approach had the advantage of allowing the user to continuously monitor the predicted loads placed on both auxiliary supplies throughout the design process, a factor often omitted or guessed during switchgear design. The class and object diagrams illustrating the relationship between `CSwitchboard`, `CPowerSupply` and the MFC

[16]Once objects are instantiated within a computer's memory, they invariably require some form of initialisation, *e.g.* the setting of values for each member variable or checking the existence of other necessary or cooperating objects. Within C++, every class has an initialisation member function, referred to as the `constructor` which is called automatically after the object has been created in the computer's memory. Constructor functions may be overloaded to accept arguments that initialise the object in a specific manner. Similarly, when an object is destroyed, the `destructor` member function is called which is utilised to assist in the correct destruction of the object.

[17]Users may have existing switchboards installed on a site, with existing auxiliary and/or tripping supplies with sufficient spare capacity to utilise these supplies as opposed to having to purchase and install new supplies.

[18]The relationship between `CPowerSupply` objects and their respectively connected loads presented several issues which are discussed in Section 5.3.3.2.

framework are given in Figure 5.11.[19]

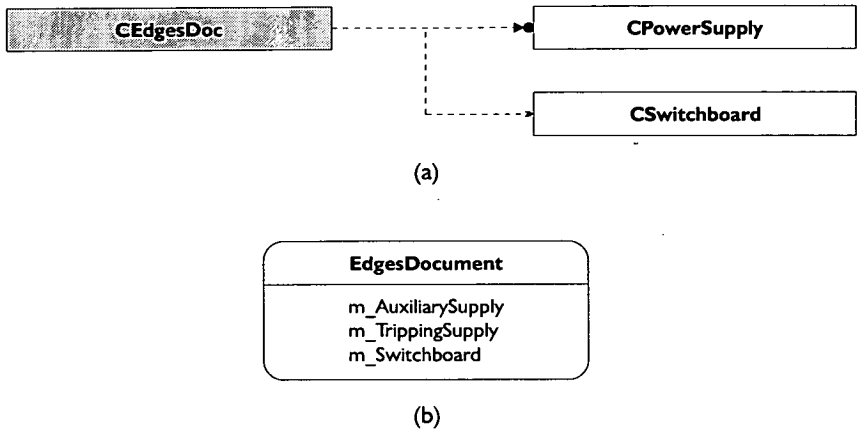


Figure 5.11: The relationships between `CSwitchboard`, `CPowerSupply` and the MFC framework; (a) class diagram, (b) object diagram depicting the inclusion of switchboard and auxiliary objects within the `EdgesDocument` object.

5.3.3 Constituent Components

The purpose of this section is to consider the constituent components necessary for the operation of each individual switchboard section, with a description of their interrelationships and how these components are modelled and incorporated within the existing class hierarchy.

5.3.3.1 Organisation of Components

The constituent components of a switchboard section is discussed in Section 2.1.3; the items of equipment described include:

- Circuit breakers,
- Current and voltage transformers,
- Fuses,
- Instrumentation equipment, and
- Protection equipment.

[19]Note that through the MFC framework all objects derived from `CObject` may obtain a pointer to `EdgesDocument` at any point during execution and hence gain access to the interfaces of all publicly declared objects included within `EdgesDocument` object.

However, additional components are also required for the control and operation of the above items of equipment which are also part of any switchgear installation. Such items are of relatively minor importance and therefore are frequently overlooked or omitted in a switchboard design specification. Such items include:

- Anti-condensation heaters,
- Control switches, and
- Indicator lamps.

All the above items of equipment are important to the successful installation and operation of a switchgear installation and therefore are included within Edges. In addition to these items of equipment, labelling for both sections and instruments is also included within the software for completeness. Other minor components not included within the model but which are provided by the switchgear manufacturer include cable boxes, foundation bolts, operational interlocks, and terminal blocks.

Accepting that each of the above components will be represented by individual classes, the next important issue is how these classes should be organised or arranged into a structure within the existing application framework, such that they fulfil similar responsibilities to their real life counterparts. In addition to this requirement, the classes developed should also interact with each other to simulate design reasoning and perform logical decision making based upon the design information contained within them.

After several attempts to structure a series of component classes it became apparent that during the initial stages of the design process of any artefact the designer will manipulate generalised concepts or components of the problem domain. As the design process proceeds, the designer will make decisions based upon trading the limitations between the various components of the problem domain against the final overall features and performance of the artefact under construction. During this process, concepts and components develop from abstract, poorly defined items into becoming definite, accurate and completely specified artefacts. Careful study of various design domains shows that this aspect of the design process, the movement from abstract to definite, has a strong parallel with the object oriented programming paradigm if the inheritance and polymorphism mechanisms are utilised appropriately.

For example, within a switchgear design scenario, the developer will have to make some initial decisions concerning all the components contained within the switchboard. Typical decisions would include deciding upon an interrupting medium, or the type of instrumentation equipment required. If such information was

entered into the design software before the commencement of defining an individual switchboard section, then this information can be used to automate the design reasoning of each section. Furthermore, each attribute that affects other aspects or components in the design domain may be checked, adapted if necessary or a query presented to the user, with the result that the overall design of the switchboard is complete, accurate and consistent.

Encoding such generalised design criteria or information may be performed within an object oriented class hierarchy by the creation of *specification classes*. The responsibility of each specification class will be to define all the common attributes associated with a constituent component of a switchboard section. Once the design domain has been populated with design data for all the components, this information may be shared with other objects. By constructing objects to share such basic but important design information (otherwise referred to as *leading dimensions*) the software can reach realistic and consistent design conclusions.

By utilising inheritance, polymorphism and the automatic memory allocation features of C++, these specification classes may be extended to create complete and realistic objects, called *design classes*, that mirror the functionality of their real life counterparts. By proceeding in this fashion, not only is the switchgear design process modelled but the reuse of existing code is also achieved. As a design scenario proceeds design objects may be automatically generated in an organised, structured fashion, resulting in correctly specified artefacts. At any juncture, these objects may be individually edited by the user or additional objects included within the design as required.

To incorporate the component specification and design classes into the existing application structure, each class will be instantiated by the `CEdgesDoc` class definition, as illustrated in Figure 5.12 through the use of the aggregation mechanism.

Upon construction, each specification object will be initialised with default design parameters, thereby allowing the software to commence the definition of switchboard sections within a design scenario immediately. To obtain the maximum benefit from the software, the user should decide upon the components to be utilised in the proposed design and enter the attributes associated with each component at the beginning of a design scenario.[20] However, if specific component information concerning one or more components is not available at the commencement of a design task, the default data may be used. Once accurate or complete information becomes available it may then be entered into the software, resulting in this new information being propagated throughout previously defined components. New components will utilise the revised component specification details. By including this mechanism within the software the consistency of switchgear specifications developed using Edges can be ensured.

[20]The details of each component are entered via a specification dialog, illustrated in Figure 6.3.

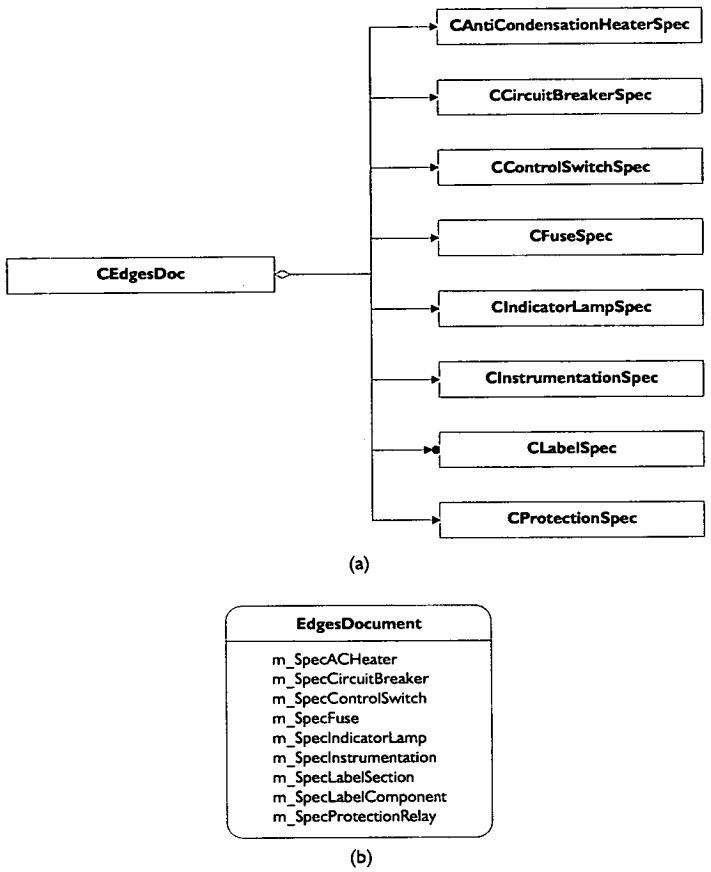


Figure 5.12: The relationships between the existing design classes and the MFC framework; (a) the class diagram, (b) the object diagram.

As the design progresses, components will be selected and included in the specification automatically to ensure that the overall design confirms to legal and regulatory standards, especially G59. Additional components may be included as required.

The following sections discuss the modelling and inclusion of component classes into the existing class hierarchy, indicating the operation of the specification classes, their extension into complete design classes and how the interaction of these objects allows design reasoning to be automated and performed. Initially the minor components will be considered before the discussion extends into the larger, more complex components.

5.3.3.2 Anti-Condensation Heaters

To exemplify several programming issues that required resolution in a clear manner, the discussion of the constituent switchboard components will commence with a very minor component before proceeding to the more complex and important

items. Anti-condensation heaters play a minor role in switchgear design. Discussions with Wallace [277] highlighted that switchgear situated in cold or damp environments should, as a matter of course, include an anti-condensation heater within each section. However, these heaters are often omitted from switchgear specifications. Such heaters are low wattage units, powered from the switchboard's auxiliary supply. By ensuring that the internal temperature of the section remains above the *dew point*, water vapour will not condense internally within any switchboard section. The inclusion of anti-condensation heaters within each section improves the reliability of the switchgear installation by reducing occurrences of electrical contact corrosion.

At some points during their operational life, switchboards may experience climatic conditions which could result in condensation forming inside a section cabinet. It is part of the designer's remit to decide if anti-condensation heaters are necessary or not. Therefore to assist the user, the software checks the climatic conditions in which the switchboard is located.[21] If the likelihood of condensation is of moderate or high probability,[22] then anti-condensation heaters will be included within each section. An experienced switchgear designer may wish to include these devices as a matter of course to ensure the reliability of the switchgear installation; this feature is also included within the software.

An anti-condensation heater specification class, called `CAntiCondensationHeaterSpec`, stores the specification attributes common to all anti-condensation heaters included within a design scenario. Such attributes include power rating, manufacturer and part number. Due to the flexibility of object oriented code, additional parameters (such as the device's physical dimensions) may be included at a later stage to augment the design detail of the device's specification. This applies not only to anti-condensation heaters but to all other components involved in the design domain.

Specification classes act as a basic data container within the class hierarchy; they contain no other features. Design classes then expand upon this basic data container by representing, in software, an actual switchboard component. Design classes therefore manipulate and store additional information that allows the precise and unique specification of the component that they represent. The development of the `CAntiCondensationHeater` design class raised several difficulties associated with including it into the existing class hierarchy, with the appropriate functionality. These difficulties are briefly discussed.

[21] The climatic conditions affect the entire switchgear installation. Therefore this data is available from the public member functions of `CSwitchboard`; users may edit this information via the specification dialog.

[22] The boundary conditions for the inclusion of anti-condensation heater, by the software, are if the average humidity level is over 80% or the average ambient temperature that the installation will experience is 5° Celsius or below [168].

Anti-condensation heaters are only one of several components that require connection to the switchboard's auxiliary supply. Therefore a common interface is required to allow communication between the power supply object and the load objects connected to it. Such an interface is symbolic of the electrical connection that would exist between loads and a supply in the final switchboard installation. Through this interface, the number and total wattage of loads connected to the power supply object may be determined as the design scenario proceeds. In exchange, each load will obtain the supply voltage and current type from the power supply object. By facilitating such information exchange, changes to either load or supply objects may be automatically communicated to the necessary object instantaneously thereby ensuring consistency in the design. In order to achieve this arrangement, a class called `CPowerSupplyLoad` was created to implement the communication required between auxiliary supply objects and their load objects. This interface must be shared between all load objects; it was the incorporation of this interface into load objects that posed several architectural issues.

First instincts would be to use the inheritance mechanism, as this feature allows the incorporation of existing class interfaces into new class definitions to be easily performed. This arrangement is depicted in Figure 5.13. However, this class arrangement illustrates a natural limitation of object oriented programming languages. Although completely legal and within the definition of the C++ language, the class hierarchy shown in Figure 5.13 fails to compile. This compile failure is due to the class `CAntiCondensationHeater` inheriting from both `CPowerSupply-Load` and `CAntiCondensationHeaterSpec`; both of these classes are derived from `CObject`. If an anti-condensation heater object were instantiated, it would contain two versions of every member function and variable from `CObject` since both `CPowerSupplyLoad` and `CAntiCondensationHeaterSpec` inherit from `CObject`. This is an unacceptable situation, hence the compile errors.[23]

A second approach would be to utilise a single inheritance approach, as depicted in Figure 5.14. This class structure resolves the multiple inheritance issues associated with the previous attempt and delivers the required functionality, but at the expense of the model's clarity. The class hierarchy illustrated in Figure 5.14 implies that the specification of anti-condensation heaters is a generalisation of power supply loads and that all specification classes inherit the ability to connect to a switchboard's power supply. This is an impossible situation within the design domain; an ambiguous arrangement. Therefore it was rejected.

Returning to the first class hierarchy, an attempt was made to resolve the issue of multiple inheritance. Any functionality from the MFC framework could be excluded from the `CPowerSupplyLoad` class definition, therefore this class would

[23]Multiple inheritance is legal within the C++ programming language and the compile errors indicated in this example can be resolved by additional programming. To resolve this issue is non-trivial with most C++ authors and experts recommending the avoidance of such class hierarchies.

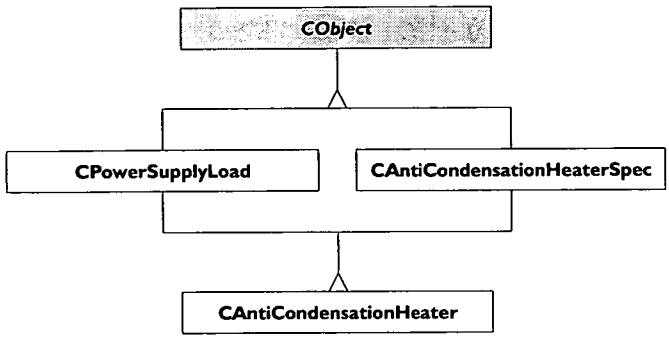


Figure 5.13: A class diagram representing the initial attempt to include the `CPowerSupplyLoad` interface within the `CAntiCondensationHeater` design class.

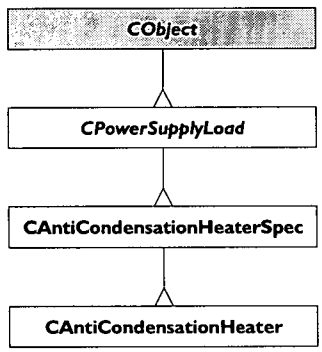


Figure 5.14: The second approach to incorporating the design class for anti-condensation heaters into the existing application class hierarchy.

not need to be a subclass of `CObject`. This class arrangement is illustrated in Figure 5.15. This approach circumvents the multiple inheritance issues present within the first class arrangement. It was rejected since only one auxiliary supply may be connected to each load object. Although this is not a limitation for anti-condensation heaters, other power supply load devices, such as modern circuit breakers and protection devices, require connection to both the auxiliary supply and the tripping supply of the switchboard. On this basis, this arrangement was rejected.

The use of inheritance, as demonstrated by the previous examples, may lead to inflexible or conceptually difficult class hierarchies. The required design functionality was achieved by the application of the aggregation technique. By creating a concrete power supply load class definition derived from `CObject` and incorporating the required number of instances of such load objects into objects that require connection to auxiliary power supplies, the symbolic link between auxiliary power supplies and their loads can be achieved. The class diagram illustrating this arrangement is given in Figure 5.16.

Having explored the practical application of utilising the aggregation mecha-

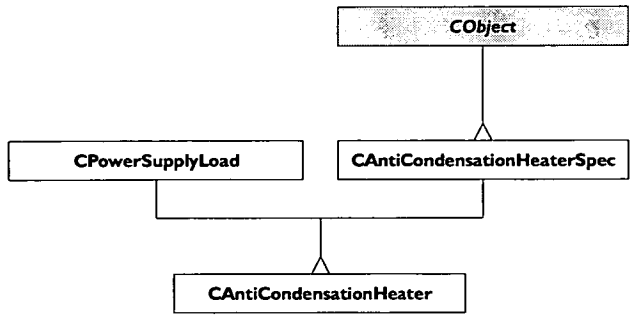


Figure 5.15: The third approach to incorporating the design class for anti-condensation heaters into the existing application class hierarchy.

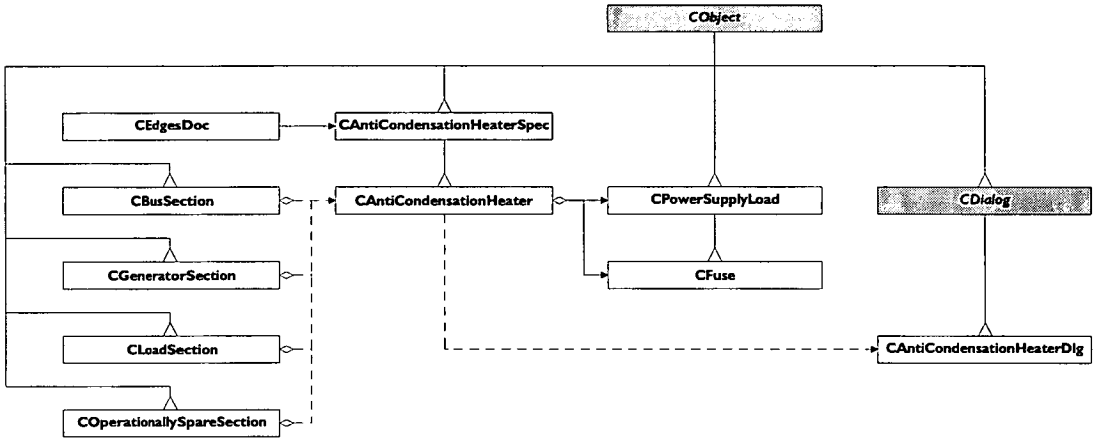


Figure 5.16: The final approach utilised for implementing anti-condensation heaters. Note the difference between aggregation and creation relationships; creation is non-permanent or optional.

nism in favour of the use of the inheritance mechanism in switchgear component design, this approach was adopted for the development of all section components within the design domain. Upon the creation of a section object, it checks to determine if an anti-condensation heater is required. If so, an `CAntiCondensationHeater` object is instantiated. During instantiation, the new anti-condensation heater object obtains its specification from the `m_ACHeaterSpec` object stored in Edges' document object. Based upon this specification information, `CPowerSupplyLoad` and `CFuse` objects are instantiated to meet the requirements of the anti-condensation heater object thereby ensuring that the anti-condensation heater is connected to an auxiliary supply and protected by a set of appropriately rated fuses. The device has now been correctly specified and included within the switchboard specification.

As previously stated, the user should be able to edit all components (assuming that they do not violate any fundamental design rules); to meet this end, an edit function has been included within the anti-condensation heater class definition

which, when called, instantiates and initialises a dialog object `CAntiCondensationHeaterDlg`[24] allowing the user to directly edit the device via a specialised dialog box. Upon the closing of this dialog box, the data is checked and transferred back to the anti-condensation heater object. Through the same dialog box, the edit function of the `CFuse` object may be called and its specifications edited via another specialised dialog box. Once again, any edits to the specification of the fuse are checked and, if appropriate, updated.

This discussion has indicated some of the programming issues associated with the creation of an object oriented design architecture. The development of specification classes for the storage of fundamental design data has been, through the use of the inheritance mechanism, extended to allow design classes capable of performing basic design reasoning to be easily developed. A suitable technique of modelling auxiliary power supplies and their loads has been developed which will be adopted throughout the rest of the design framework. Finally, the aggregation mechanism has been applied to allow different section types to include anti-condensation heaters within their specifications, if required.

5.3.3.3 Control Switches

Control switches are the general term for low voltage electric switches mounted on the front panel of a switchboard section that enable the control and operation of that section. Typical functions for such devices include tripping/resetting the circuit breaker or assisting the use of instrumentation devices. The class hierarchy for control switches and their relationship to the other design classes is depicted in Figure 5.17.

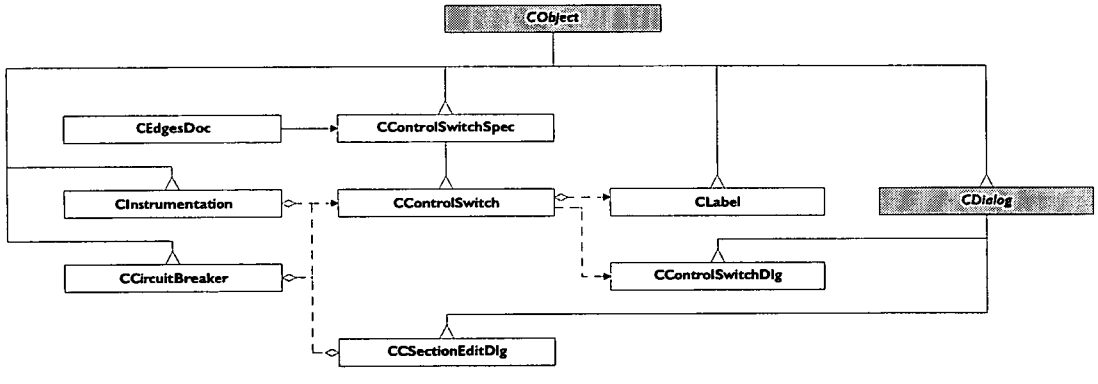


Figure 5.17: The class diagram for control switches.

Following an architecture similar to the one adopted for anti-condensation heaters, the default specification for control switches is stored in the Edges document

[24] All dialog classes are indicated by having the extension `Dlg` at the end of their class name.

object and is used to initialise all CControlSwitch objects. The electrical attributes defined within the specification class for control switches, CControlSwitchSpec, includes the device's contact rating, current type, contact operation and the number of contacts. The control switch design class, CControlSwitch, augments and enhances the basic features of the control switch specification class. Supplementary features include the addition of a function description and part number member variables, a CLabel object and several member functions to assist with the management of control switch design objects. The function description variable stores a text string that describes the purpose that the control switch performs and forms a unique identifier for the device. It may not be edited by the user once the control switch is defined. This text string also forms the default facia label for the control switch, but the user may edit this to suit his/her preferences. As with all constituent components an edit function allows the user to view and adjust the properties for the control switch through a dialog box instantiated from the CContolSwitchDlg class.

The main purpose of control switches is to enable the operation of either circuit breakers or instrumentation devices. Additional control switches may be manually defined by the user. These three circumstances require that control switches are included within a section specification. They are indicated by Figure 5.17 and are explained as follows:

1. The instantiation of a CCircuitBreaker object will, by default, include a trip/neutral/close switch. If remote operation of the section is required an additional control switch will be added, labelled local/remote.

2. The instantiation of a single phase panel meter will (from the CInstrumentation class), by default, include a control switch to allow measurement from any phase to be selected by the switchboard operator, labelled R/Y/B/Off.

3. The direct instantiation of a control switch by the user; when editing a section via the section edit dialog box, the user may include additional control switches as required.

Due to the flexibility of object oriented software, control switches may be included within other design class instances by utilising the aggregation mechanism. Indeed the above three situations may be easily edited or extended to satisfy other design situations.

The additional design data that is not available from the m_SpecControlSwitch object but is required to instantiate the control switch design objects is obtained from the objects that instigate their creation. This data is passed from the

instigating object to the newly formed control switch object via a multi-argument constructor call. As both circuit breaker and instrumentation objects already know their requirements, this is a straightforward process.[25]

All control switch objects, in common with all design objects, are created dynamically. In practical terms this means that control switch design objects are allocated in some random memory location determined by the operating system. Therefore, once an object has been created, a reference to its location must be stored within the application framework to ensure that the object in question can be located and utilised until it is no longer required or the application is terminated. Failure to keep track of dynamically created objects results in these objects being effectively lost in the computer's memory, a situation referred to as a *memory leak*. However, several different objects may contain a reference to a single object thus allowing many to one and one to many relationships to be represented.

Both circuit breaker and instrumentation objects retain a reference to any control switch object that they instantiate, as they form an important part of these devices' specification. However, any changes in the default specification of all control switches, instigated by the user through the component specification dialog, would necessitate a search through all circuit breaker and instrumentation objects to locate all the control switches and then modify their data. In order to avoid searching through other design objects and to facilitate the storage of user defined control switches, a dynamic array was included within each section object, named *m_ControlSwitches*, that allows an identical reference to every control switch object required within a section to be stored. This arrangement is illustrated in Figure 5.18. A similar arrangement is utilised for all other components, such as fuses and labels, that the more complex design objects create.

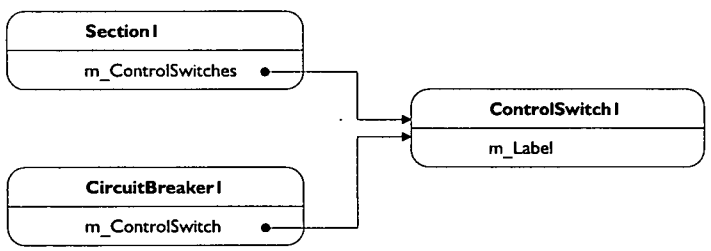


Figure 5.18: The object diagram for a control switch object defined as part of a circuit breaker specification.

[25]To illustrate this point consider the contact operation of a control switch which depends upon its intended purpose. If defined to control a circuit breaker, a non-latching set of contacts is required, but if defined to assist the operation of a single phase current meter via a set of current transformers, a set of make before break contacts are defined instead.

5.3.3.4 Labelling

The inclusion of labelling within a switchboard specification is a very minor issue but they are included within the model as labels are frequently omitted from specifications. Labels are mounted on the front facia of switchgear sections to identify the function of a section, *e.g.* "Mill House Feeder", or the function of the devices, *e.g.* "ROCOF Relay" or, in the case of a control switch "Generator Shutdown". Typically such labels are produced on plastic (trafolyte) or embossed aluminum. The incorporation of labels into the existing design class hierarchy is depicted in Figure 5.19.

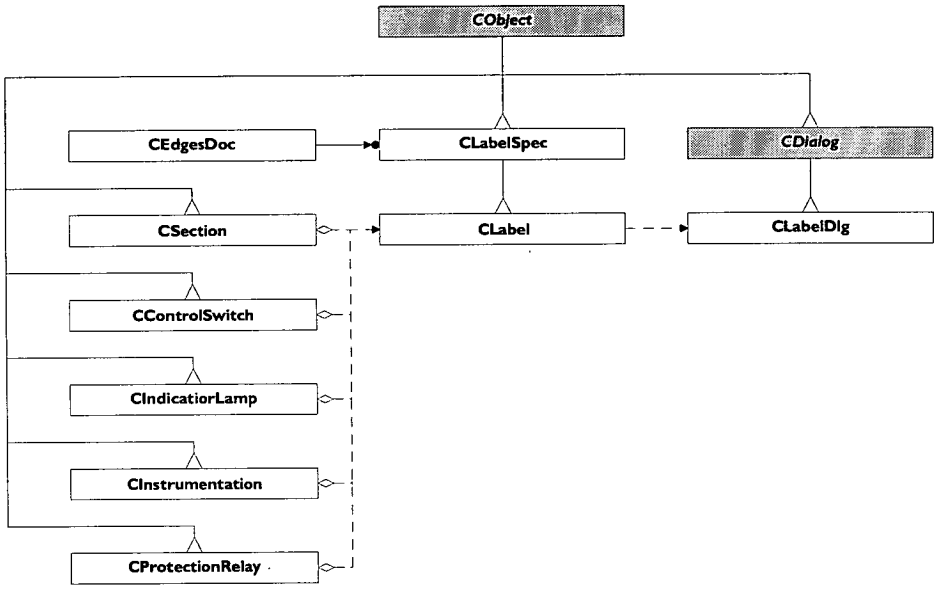


Figure 5.19: The class diagram for labels.

The inclusion of labels follows the same pattern of previous constituent components. The `CLabelSpec` class contains the attributes common to all labels within the design domain; namely the material from which they are constructed, the colour of the text appearing on the label and the text height (in millimeters). Two *CLabelSpec* objects are instantiated and stored within the Edges document object, one of which defines the section labels, the other, all other component labels. Once again, it is through a multi-argument constructor that the instantiating objects determine which label type is required.

The `CLabel` design class reuses and expands the basic class definition of `CLabelSpec` to allow a text string to be stored which will be the text that appears on the label. Several additional member functions are also included to assist in the management of `CLabel` objects, most notably an edit function that allows the attributes of individual labels to be edited; items that can be changed include the label's text, colour, size and material. By default, the labelling for devices is re-

duced in size in comparison with section labelling. Items that require labels as part of their specification are indicated on the left hand side of Figure 5.19.

5.3.3.5 Indicator Lamps

Indicator lamps allow the annunciation of important information concerning the switchboard's status to be visually indicated on the front facia of a section. Typically, such events would include busbar live indication, circuit breaker status and device malfunction, although they are mainly dependent upon the designer's preferences and the function of the section. Indicator lamps are now fast disappearing from the front of modern switchgear installations, replaced by electronic or computerised notification of events; they are favoured by some designers and are included within the design software for completeness. The lamps installed within a section obtain their power from one of the auxiliary supplies of the switchboard, via the device which controls their operation, or directly from the bus bars via a voltage transformer; which supply is utilised depends upon their function. The inclusion of indicator lamps within the software is illustrated in the simplified class diagram of Figure 5.20.

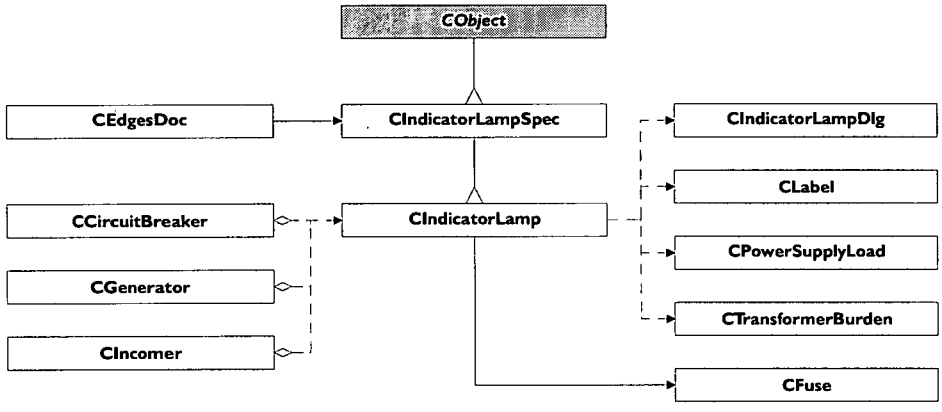


Figure 5.20: The class diagram for indicator lamps.

The attributes common to all indicator lamps are stored within the `CIndicatorLampSpec` class which includes data concerning the lamp's wattage, manufacturer and model. Based upon these common attributes, lamp objects are created using the `CIndicatorLamp` design class which augments the specification class by defining several member functions to assist in the management and editing of indicator lamp objects and several member variables. Each lamp requires a label, a hood colour (*e.g.* red for busbars live), a fuse and connection to a power source. This additional information is provided via a several argument constructor. From the arguments provided by the constructor, the indicator lamp is connected

to a source of energy; either a voltage transformer secondary,[26] by instantiating a `CTransformerBurden` object, or the switchboard's auxiliary supply, by instantiating a `CPowerSupplyLoad` object. From either energy source, the indicator lamp object dynamically obtains its voltage rating and based upon this rating, a correctly sized fuse is included in the lamp's specification.

5.3.3.6 Fuses

The main function of fuses within a modern switchboard installation is to provide protection for items of equipment contained within a section connected to either an auxiliary supply or a voltage transformer. Devices that require protection are indicated on the left hand side of Figure 5.21, which depicts the class relationships between the fuse specification and design classes and the other classes that constitute the switchboard design software.

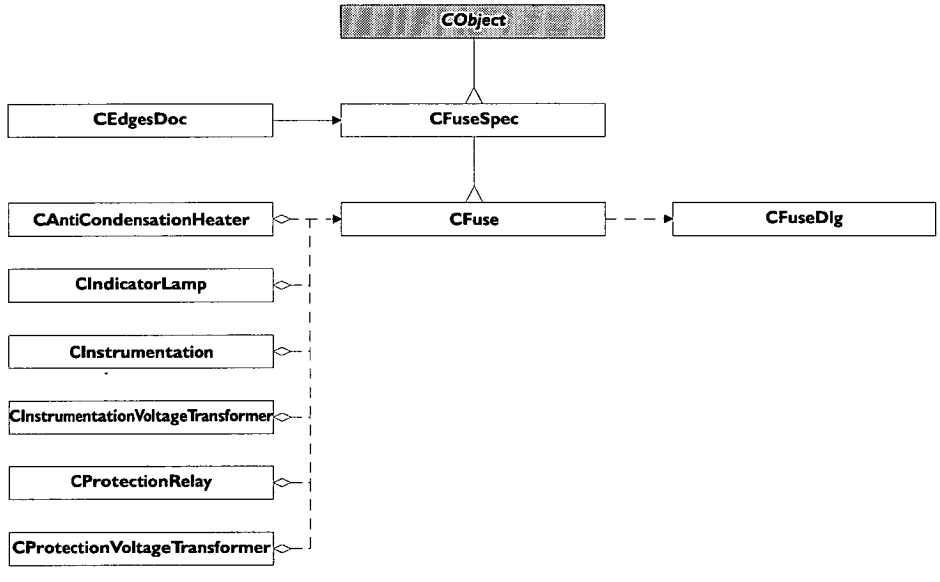


Figure 5.21: The class diagram for fuses.

Following a similar architecture to other constituent component classes, the attributes common to all fuses are captured within the specification class `CFuseSpec`, which includes applicable construction standards, the manufacturer, and the mounting type. Fuse objects are instantiated as required, dynamically, by the objects that require them from the `CFuse` design class. Additional features included in the fuse design class were automatic voltage and current rating, number of fuses required and storage for individual order or part numbers. Extra member functions are included to assist in the management of `CFuse` objects; most notably `Edit()`, and `UpdateRating()`. The `Edit()` function allows the user to directly

[26] Indicator lamps are only directly connected to a voltage transformer when their function is to indicate the energisation status of the busbars.

edit any fuse object via a specialised dialog box. The `UpdateRating()` function allows a fuse object to find the voltage and power rating of the device to which it is connected via a reference; this arrangement is shown in Figure 5.22. Once the voltage and power rating of the protected device is known then then the `UpdateRating()` function proceeds to determine the nearest standard value fuse rating appropriate to protect the device. Once obtained the fuse rating is stored as a member variable within the object. In the event that the rating of the device protected by the fuse changes, the device then calls the `UpdateRating()` function and the revised rating for the fuse is then determined and stored.

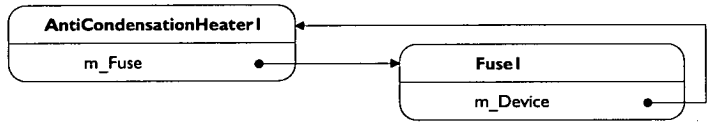


Figure 5.22: An object diagram illustrating the relationship between a fuse protected device (in this case an anti-condensation heater) and a `CFuse` object.

5.3.3.7 Circuit Breakers

The devices that perform the actual switching of electrical circuits connected to the switchboard are the circuit breakers. Every section within a switchgear installation contains a circuit breaker with the exception of spare sections; this relationship between the circuit breaker design class and the sections in which they are contained is illustrated on the left hand side of Figure 5.23.

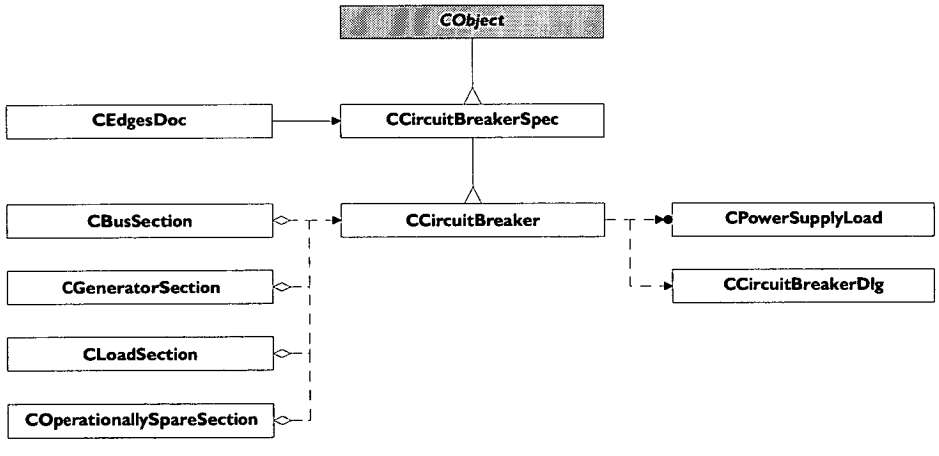


Figure 5.23: The class diagram for circuit breakers.

The specification class `CCircuitBreakerSpec` contains all the attributes that are required to specify a circuit breaker, such as the interrupting medium, the charging type and wattage (if applicable) of the spring charging mechanism and

its required auxiliary voltage, the wattage of the spring release and shunt trip coils and their required voltage, applicable standards relating to the device's construction, and part number. The manufacturers' details are omitted as these are obtained from the `CSwitchboard` object.

The above attributes are inherited and enhanced to create the `CCircuitBreaker` design class. Additional member variables include the circuit breaker's rating (in amps) and the electrical connections to both auxiliary and tripping supplies, if required. Upon instantiation of a new section (with the exception of spare sections) the user is required to enter the size of the load or energy source as this parameter cannot be determined via any other means.[27] From this single piece of information combined with the circuit breaker specification details, a completely specified circuit breaker object can be instantiated as part of the components associated with a section. The load size is checked and the necessary capacity breaker size is found by the member function `BreakerSize()`.

Having instantiated an appropriately sized circuit breaker object, the spring charging mechanism, if required, is connected to the auxiliary power supply of the switchboard followed by the connection of both the spring release and shunt trip coils to the tripping supply. All three loads are automatically fused with appropriately rated fuses. Several other management functions are also included within the public interface of `CCircuitBreaker` including the `Edit()` function that allows the user to edit the details of individual circuit breakers via a dialog.

5.3.3.8 Current and Voltage Transformers

The inclusion of current and voltage transformers (otherwise referred to as *sensing transformers*) within the model present a set of similar design issues comparable to those encountered by the modelling of auxiliary power supplies and their loads. Each sensing transformer, although not a source of energy, may be considered as such when devices are connected to their secondary windings. Therefore a similar approach to the modelling of auxiliary power supplies and loads is adopted to include sensing transformers and the devices connected to them (termed *burdens*).

Current and voltage transformers are a specialised application of the general concept of the transformer, therefore to model this feature and allow for future development of the software to include other types of transformers, an abstract class called `CTransformer` was developed. This class includes all the general features that characterise a transformer, such as primary and secondary winding ratings, capacity (in VA), number of phases and the ability to manage burdens connected to the secondary windings of the device. In a similar fashion to power supply loads, a

[27] Other leading dimensions are also required; for further details refer to Section 6.4.

class called `CTransformerBurden` was developed to define an interface between `CTransformer` objects and their burdens.

There are significant differences between how current and voltage transformers are specified within a switchboard installation. Hence it was decided to create two subclasses of `CTransformer` to model these differences. Thus `CCurrentTransformer` and `CVoltageTransformer` classes were created. However, as discussed in Section 2.1.3.4, there exists within switchgear design two distinct categories of sensing transformers; namely instrument grade and protection grade transformers. Therefore to include both of these important categories within the software model, a further four classes were derived, two each from both `CCurrentTransformer` and `CVoltageTransformer` classes. The complete class hierarchy is illustrated in Figure 5.24.

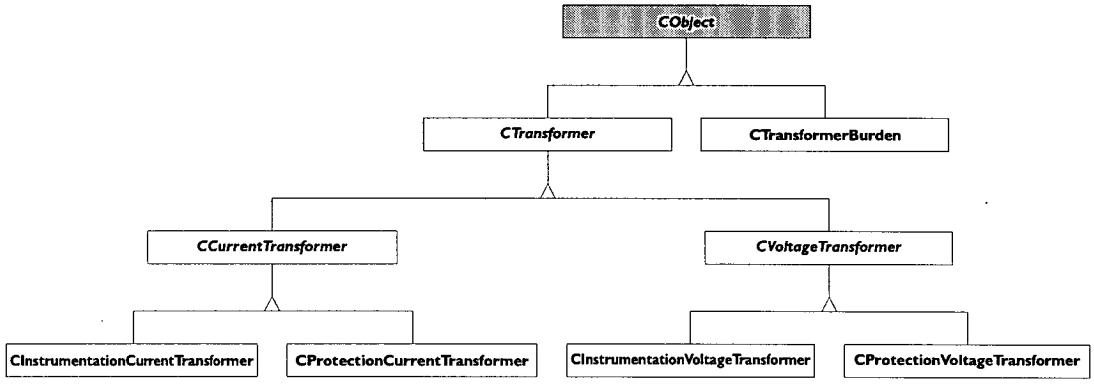


Figure 5.24: The class diagram for transformers and transformer burdens.

Note that the design class hierarchy depicted in Figure 5.24 omits any specification classes. The reasoning for not including specification classes for sensing transformers is two fold. Firstly, the inclusion of a specification class in the upper portion of the transformer class hierarchy would be meaningless due to the diverse operation of the lower devices that the classes represent. However, including specification classes at the lower specialised levels of the class hierarchy would lead to the creation and management of an additional four classes in an already complex class hierarchy for a relatively minor set of devices. This was a situation to be avoided.

Secondly, both instrumentation and protection grade transformers serve two distinct groups of devices – instrumentation and protection equipment. Individual devices of either group all have various, distinct sensing transformer requirements. For example, a protection device may require connection to a one, two or three phase current and/or voltage transformer dependant upon the function of the protection device. This is also the case for instrumentation devices. Therefore it is impossible to define in advance meaningful specifications for sensing transformers as the only attributes common to either current or voltage devices are the

secondary VA rating and accuracy class. Other transformer attributes, such as the capacity of the device, the primary rating and burden size, can only be determined at runtime. Thus the adoption of specification classes for each type of current or voltage transformer proved to be ineffective, and hence was rejected.

The instantiation of a current or voltage transformer object requires the determination of the required accuracy and secondary rating of the device.[28] This issue was resolved by including these leading dimensions within the specification classes of the objects that would require such transformers, specifically the `CInstrumentationSpec` and `CProtectionRelaySpec` classes.

The abstract class, `CTransformer`, provides a generalised definition of the function of a transformer. Through the use of inheritance and virtual functions, the subclasses of the `CTransformer` redefine this abstract definition into a set of fully functional concrete classes. For example, the `CCurrentTransformer` class redefines the function `SecondaryRating()` to ensure that it can only be defined as either 1 or 5A, however, the `CVoltageTransformer` class redefines the same function to return either 110 or 440V. Furthermore, the `CVoltageTransformer` class utilises the aggregation technique to include a set of primary fuses for the protection of the transformer. At the bottom of the inheritance structure the final concrete classes possess multi-argument constructors, the ability to automatically determine their appropriate size and accuracy (based upon the burdens placed on their secondaries) and several management functions including the instantiation of an edit dialog to allow users to view and edit these devices. The inter-class relationships between all the sensing transformers' design classes, their respective dialog classes, and instrumentation and protection design classes are illustrated in Figure 5.25.

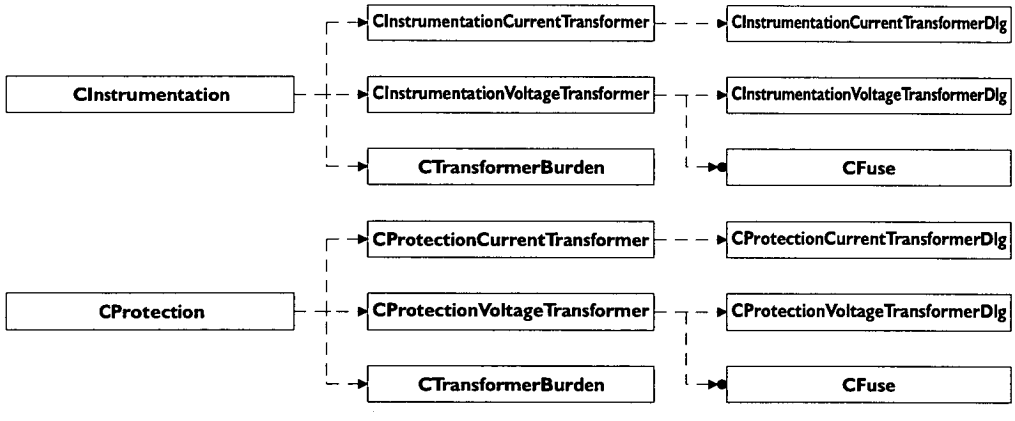


Figure 5.25: The class diagram for current and voltage transformers.

[28]The burden which the instrument, lamp or protection relay places on either a current or voltage transformer is considered part of that device's specification, not part of the transformer specification.

5.3.3.9 Instrumentation Equipment

Instrumentation devices provide accurate information concerning the electrical conditions of a section during the operation of a switchboard. The class diagram for instrumentation devices is depicted in Figure 5.26. The class hierarchy follows the familiar design pattern of specification class leading to a design class. The inclusion of protection grade sensing transformers allows instrumentation devices to be connected to these devices, a situation that occasionally arises for small, limited capacity switchgear installations.

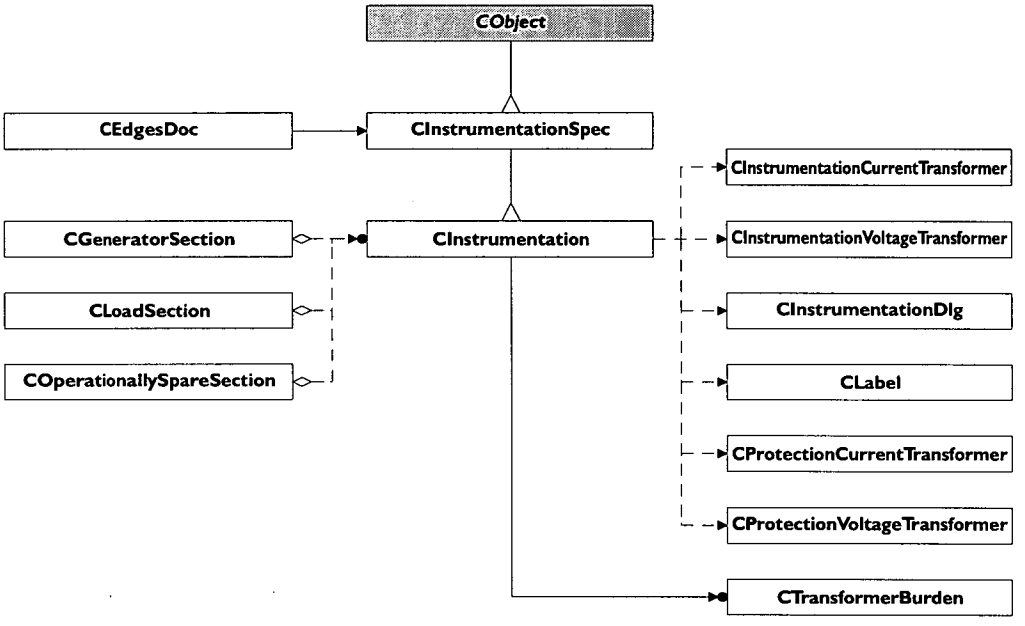


Figure 5.26: The class diagram for instrumentation devices.

The specification class for instrumentation devices provides storage for all attributes common to these devices, which includes current and secondary voltage ratings and respective burdens, physical instrument size, range, required accuracy, manufacturers details and applicable construction standards. In common with all constituent components, typical valid parameters are provided by default which the user may edit through the component specifications dialog box. The `CInstrumentation` design class augments the specification class by providing management functions and several member variables that include `CLabel` and `CTransformerBurden` objects, a part number, end scale and scale label text strings.

Instrumentation devices have a large number of different forms and functions. The data within the instrumentation specification class is insufficient to define a realisable, functional instrument. Therefore, the user or Edges must provide the other leading dimensions that allow an instrument to be defined. These dimensions are the function of the instrument, the required scale label and the number of

current and voltage transformers that the device requires to fulfil its function. The function attribute is a non-editable text string that identifies the parameters the instrument measures, *e.g.* power factor or current. This function attribute ensures that the instrument can always be uniquely identified within a section. The scale label identifies the required textual string that appears on the instrument. The current and voltage transformer parameters define the number of connections of each device that the instrument requires; zero, one, two, three or four are all valid arguments. Furthermore, a single instrument may have non-zero values for both these attributes; for example a power meter would be defined by having connection to three current transformers and a single voltage transformer.

By default, Edges automatically includes a current meter and a selector switch within each section, although the user may exclude this device from the final switchboard specification. To illustrate how instruments are instantiated, initialised and connected to sensing transformers via burdens, the interaction between these classes has been depicted in Figure 5.27 for the case of a voltage meter.

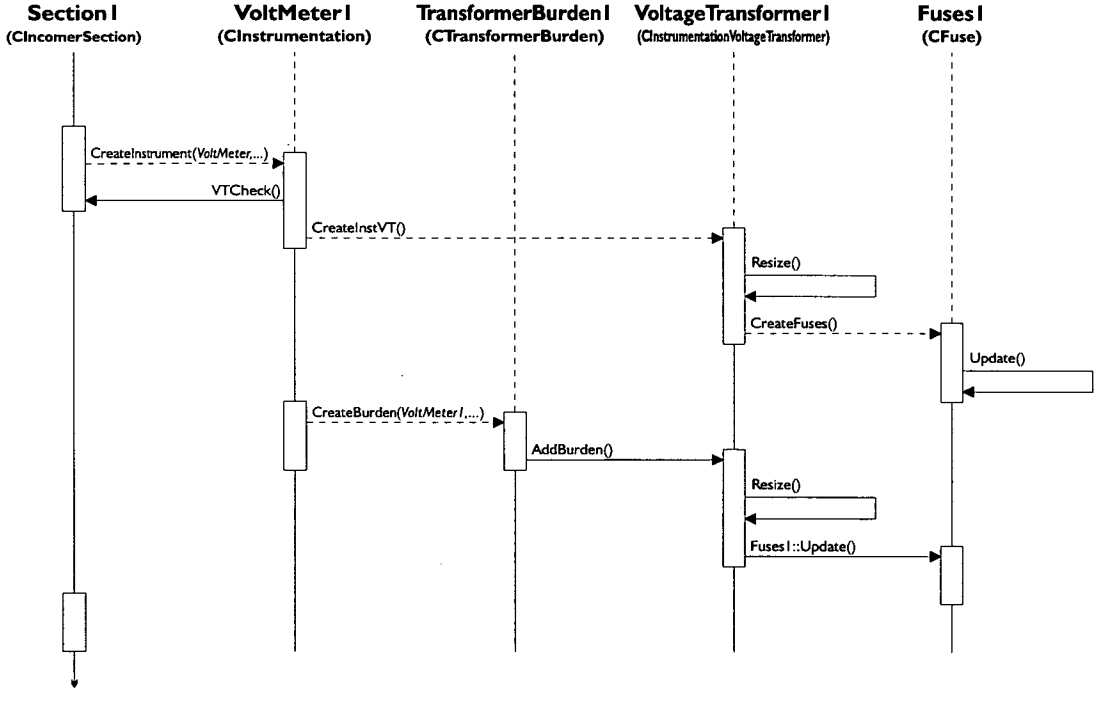


Figure 5.27: The interaction diagram for a voltage meter. Note that to simplify the complexity of this diagram, the initialisation of the instrument's label and the burden's fuse objects have been omitted.

The object interactions represented in Figure 5.27 are explained as follows. During its initialisation a section, in this case an incomer section, recognises from the constituent component specifications that it requires a three phase volt meter. Therefore `Section1` instantiates a volt meter object from the `CInstrumentation` class through a multi-argument constructor call which defines the device's properties.

The `VoltMeter1` object now knows that it requires a connection to a three phase instrumentation grade transformer, however, this object is not aware if such a device already exists. Thus `VoltMeter1` object queries the section that instantiated it, `Section1`, to determine if a suitable voltage transformer currently exists. In this case, no such transformer exists so the `VoltMeter1` object instantiates a suitably rated three phase transformer.

Upon instantiation, the instrumentation voltage transformer defines its capacity which, automatically, selects the smallest standard size followed by instantiating a set of fuses that in turn automatically obtain the current rating of the voltage transformer and rate themselves appropriately. Execution control now returns to the volt meter object which now knows that an instrumentation grade transformer exists: it can now proceed to make a connection to this transformer via a `CTransformerBurden` object with an appropriate rating. By including a reference to the voltage transformer, the burden then automatically connects and loads the voltage transformer. This action results in the transformer checking its capacity, accuracy and the fuses that protect it to ensure that they are all correctly rated; this is achieved by calling `Resize()` function. The volt meter has now been installed, specified and connected to a correctly rated voltage transformer which in turn is protected by appropriately rated fuses. Control will now return to the incomer section object, `Section1`, which will continue specifying other constituent components.

5.3.3.10 Protection Equipment

Protection equipment provides electrical protection against abnormal network operating conditions for the section in which the equipment is mounted by continually monitoring the generator, load or incomer via protection grade sensing transformers. If an abnormal condition is encountered, the protection device causes the section circuit breaker to trip, disconnecting and thereby protecting the section. The majority of protection equipment take the form of individual devices, referred to as relays, though modern protection devices are of the form of electronic multi-function devices. The class hierarchy for protection relays follows the specification/design class pattern characteristic of all constituent components, and is represented in Figure 5.28.

The specification class, `CProtectionRelaySpec`, stores the attributes common to all items of protection devices, which include the load imposed on the direct current tripping supply, current and voltage transformer requirements (number of phases, burdens per phase and preferred secondary ratings), manufacturers' details and applicable construction standards. The protection relay design class has

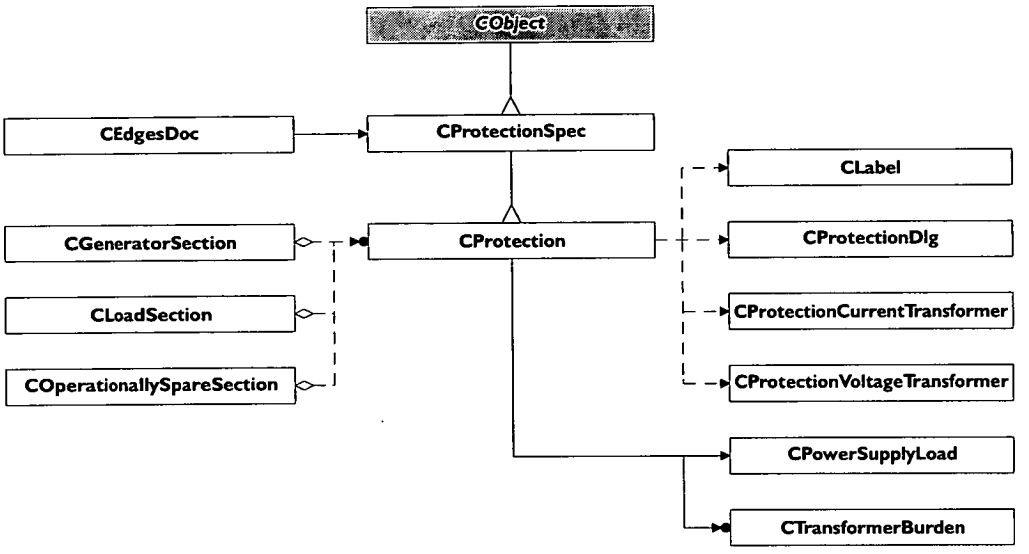


Figure 5.28: The class diagram for protection relays.

been developed in a similar manner to the instrumentation design class by including a function descriptor and part number text strings that allow any device to be uniquely identified. The objects `CLabel`, `CPowerSupplyLoad` and `CTransformerBurden` are included to allow the protection device to be labelled (if required) and connected to the tripping supply and to the appropriate protection grade sensing transformers.

The function descriptor allows the functions that the protection device performs to be included within each protection object, for example 'under and over voltage' or 'over current and earth fault'. By adopting this feature, the wide variety of protection devices can be specified from one design class, rather than a multitude of such classes; one for each type of protection device. In a similar fashion to instrumentation devices, protection relays may be connected to any number or combination of protection grade current or voltage transformers. These details, along with the functional descriptor, form the leading dimensions for the protection relays, and therefore must be provided by either Edges or the user (via a specialised dialog box).

Edges automatically defines all the protection equipment for each section to ensure that the entire switchboard is compliant to G59. The user may view the protection devices and edit their attributes or include additional protection devices where they deem it necessary.[29] The number and type of protection relays included within a switchboard section depends upon the section's function. Therefore when the user elects to add a new section to the switchboard, Edges can automatically define and electrically connect the appropriate protection relays for that section to

[29]There are only a limited number of occasions where the user might wish to include additional protection devices; this feature has been included within the software to enhance its flexibility.

the necessary and correctly defined protection grade current and/or voltage transformers, as well as connecting the relay to the tripping supply. A summary of the protection devices that Edges automatically defines for each section type are listed in Table 5.1.

Section Type	*Sub-Type*	*Protection relays installed*
Bus Section	n/a	None
Generator Section	Induction Generator	Over voltage
		Under voltage
		Over frequency
		Under frequency
		Over current
		Earth fault (>150kVA)
		Loss of mains
		Reverse real and reactive power (>250kVA)
	Synchronous Generator	Over voltage
		Under voltage
		Over frequency
		Under frequency
		Over current
		Earth fault (>150kVA)
		Loss of mains
		Reverse real and reactive power (>250kVA)
		Synchronising Relay
Incomer	n/a	Over current and earth fault
		Over voltage
Load Section	Induction Motor	Over current and earth fault
		Over voltage
	Passive Load	Over current and earth fault
		Over voltage
	Semi-Conductor	Over current and earth fault
		Over voltage
	Synchronous Motor	Over current and earth fault
		Over voltage
Operationally Spare Section	n/a	Over current and earth fault
		Over voltage
Spare Section	n/a	None

Table 5.1: The protection devices created automatically by Edges, listed according to section type. Note that the values in brackets indicate the size of the machine required for the protection device to be installed.

The interaction between a current operated protection relay object, in this case an over current relay, with a protection grade current transformer, a transformer burden and an auxiliary supply is illustrated in Figure 5.29. This figure visually indicates the object interaction required to include a protection device within the switchboard design.

The object interactions represented in Figure 5.29 are described as follows. The protection relay is instantiated by a section; in this case the section is an induction generator section which provides the leading dimensions sufficient to define the over current protection relay device. Once the protection relay has been instantiated, it checks to see if a protection grade current transformer already exists in this section. In this instance, no such transformer exists, resulting in the over current

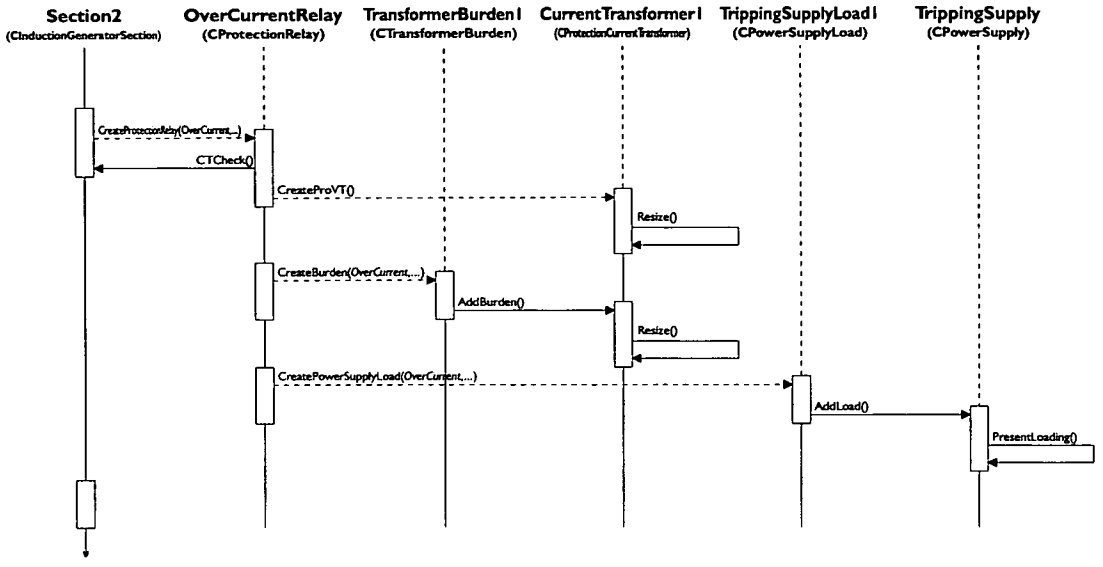


Figure 5.29: The interaction diagram for a over current relay.

relay object instantiating a suitably specified current transformer. Once instantiated, the current transformer initialises itself and automatically defines its capacity (in VA) by calling it's member function `Resize()`. Control now returns to the protection relay which, having instantiated a suitable transformer, instantiates a burden for the transformer via a several argument constructor call, allowing the transformer burden to correctly identify the current transformer to which it will connect the protection relay object.

The majority of protection relays require additional electrical power to operate; this power is obtained from the secure tripping supply. Therefore the next stage of initialisation for the protection relay is to connect to the tripping supply, via a `CPowerSupplyLoad` object. Hence the instantiation of the `TrippingSupply-Load1` object. Once again, by the use of a multi-argument constructor call, the power supply load object can automatically connect to the tripping supply with the correct load for the relay. This results in the `TrippingSupply` object calling its member function `PresentLoading()` to sequentially request the loading of each `CPowerSupplyLoad` object defined throughout the entire switchboard. It then calculates the new total loading on the supply which is then stored. The instantiation of the power supply load object completes the initialisation of the protection relay object resulting in control being passed back to the section object to complete its initialisation.

5.3.3.11 Summary

This concludes the discussion of the constituent components necessary for the construction and operation of a switchboard. It has been seen that basic design objects

may be combined to form more complex objects. For example instrumentation or protection devices are coded such that together they perform the basic design reasoning necessary to specify themselves, their related components (if any), and connections to either auxiliary supplies and/or sensing transformers.

5.4 Chapter Summary

This chapter has considered the key aspects of the object orient programming language C++, indicating the four mechanisms (abstraction/encapsulation, inheritance, polymorphism and aggregation) that allow this software technology to operate. An object oriented architecture for multi-component design was described, indicating how domain knowledge and the features of C++ allow design rationale to be captured in a software application, named Edges. The various key features that constitute a switchboard were individually considered and captured in specific classes. The architecture allows abstract, component specification classes to be reused and expanded into design classes which cooperate together to allow switchgear design to be modelled.

Having discussed the operation of the software at the language level in this chapter, it is the purpose of Chapter 6 to describe how the software operates from a user's point of view through the description of a case study, from which an overall appreciation of the operation of Edges will be given.

Chapter 6

System Operation

The previous chapter discussed the construction of, and interaction between, object oriented specification and design classes that allow Edges to perform switchgear design. This chapter extends this discussion from the point of view of the user by indicating how the project software is used in practice for designing and producing a complete specification for switchgear installation. Rather than discuss the software in general terms, a case study will be presented to focus the discussion of the software's operation. The chapter commences by outlining the main features of the application's graphical user interface, followed by the definition of the switchboard's general properties and the specification attributes of section components. Having defined and edited the overall features of the proposed switchboard, the definition of individual sections may proceed. Once one or more sections have been included within a proposed design, their constituent components may be adjusted to suit the users requirements before the final switchboard specification is produced. The chapter concludes with an overall summary of the software's operation.

6.1 Application Interface

One of the criteria that the project software has to fulfill[1] is that it utilises a standard graphical user interface that will operate on a personal computer. The Microsoft Windows operating system meets this criterion. By utilising the Developer Studio, an application that takes advantage of Windows interface features can be constructed. The graphical user interface created using Developer Studio for the project software, Edges, is depicted in Figure 6.1 during a typical design scenario.

The important features of the application interface are explained as follows. The *menu bar* provides access to all the main features of the application through a series

[1] As discussed in Section 4.3.

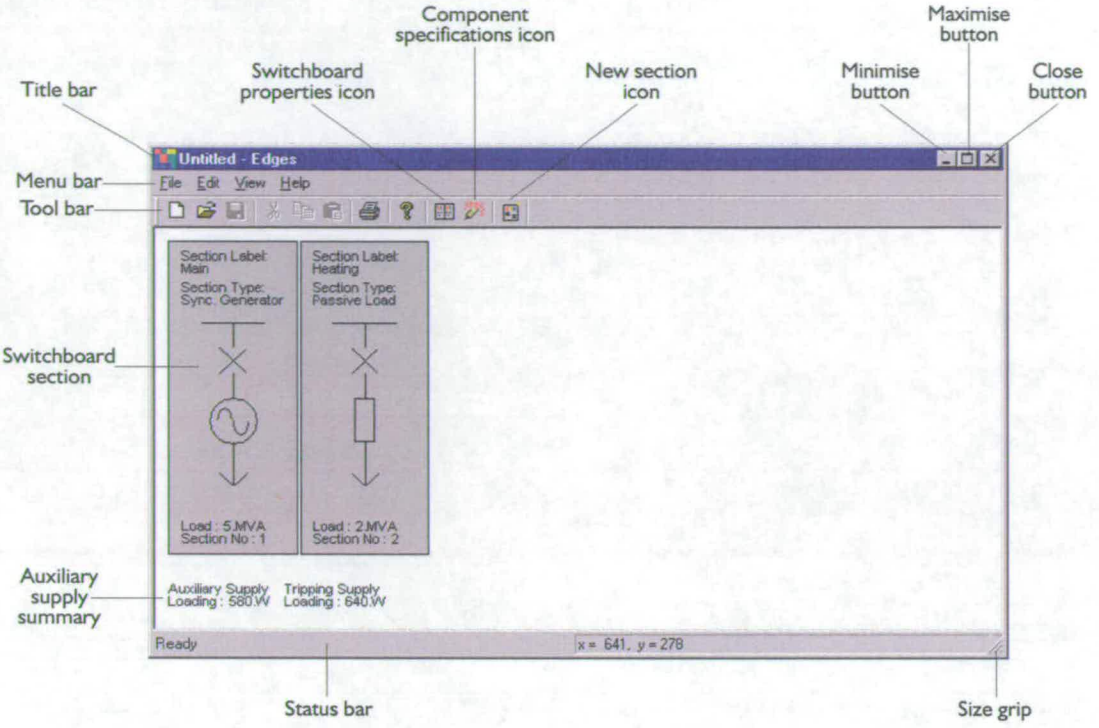


Figure 6.1: A screen shot of Edges during a design task, with the key features of the Windows graphical user interface highlighted and labelled.

of standard Windows drop down menus. However, the *tool bar* allows the user to access quickly the same features but through a series of coloured icons. The *status bar* at the bottom of the screen provides a description of the current status of the application and descriptions of menu or tool bar items highlighted by the mouse cursor. Two switchboard sections have already been defined and are drawn in the main portion of the window; each gray rectangle represents a switchboard section within which a graphical depiction of the section's type and key details are given. By moving the mouse pointer and selecting a section with the left mouse button, the user can edit the details and constituent components of that section, via the *edit section* dialog.

The following sections describe how the software would be used to define the paper mill switchboard illustrated in Figure 2.5 on page 21. It should be noted that due to space limitations the majority of dialog boxes and minor features of the software have been omitted to allow the discussion to focus on the key aspects of the switchgear design software.

6.2 Commencing the Design

Before commencing a switchboard design, the user must have a basic idea of the number and type of sections that will comprise the proposed switchboard and the type of components they wish to utilise within each section of the switchboard. These details will probably change during the course of the switchboard design; however, such changes are easily accommodated by the software.

To commence the design of a proposed switchboard, the user starts Edges which results in the main application window appearing on the Windows desktop. By default, Edges will automatically create a new design document with a default switchboard that does not contain any sections and a complete list of constituent component specifications (for devices such as circuit breakers, control switches and fuses) populated with appropriate parameters. The user could at this point immediately commence the definition of switchboard sections. However, the switchboard specification produced would be based upon these default switchboard properties and constituent component parameters. To maximise the benefit from the software, the user should edit the switchboard properties and component specifications appropriate for their proposed installation. Once these settings have been entered, they may be saved to disk for future use.

To edit the switchboard properties, the user selects the switchboard properties icon from the tool bar, using the mouse pointer, resulting in the dialog depicted in Figure 6.2 being displayed on screen. The dialog box consists of four *property pages* or tabs, one for each group of switchboard attributes; namely electrical, physical, location and climate. The electrical properties of the switchboard are visible in Figure 6.2, however, the values illustrated within each of the edit controls have been adjusted from the default values to suit the example of the paper mill switchgear installation conditions.

By selecting, say, the busbar rating edit control, a list of typical ratings appears to guide the user. In this case the user may select one of these standard ratings or enter a specific value into the edit control. The properties for both auxiliary supplies may be edited by selecting the appropriately labelled button, resulting in the corresponding power supply dialog appearing. The switchboard attributes that are available through the physical, location and climate tabs are summarised in Table 6.1.

Once the user has completed editing the general properties for all four aspects of the switchgear installation, they then select the OK button, causing the *switchboard properties* dialog to close and this information to be stored in the application document. If, at any point during the design process, the user changes any of the switchboard properties, the changes will be checked, stored and distributed

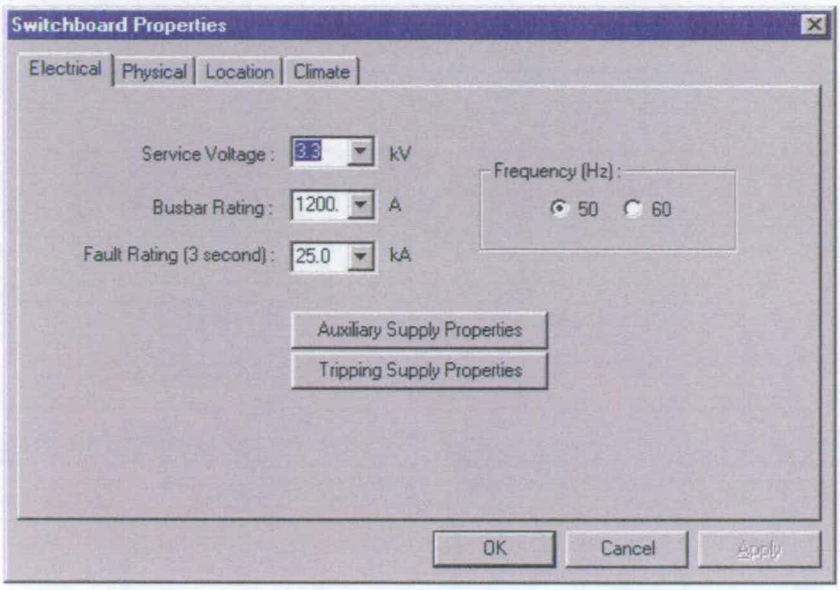


Figure 6.2: The switchboard properties dialog.

Attributes Group	*Attributes*
Physical	Manufacturer, model, ingress rating (IP number), paint colour, construction standards, label inclusion, circuit breaker isolation/withdrawal type (horizontal or vertical)
Location	Site name, site location, commencement and revision dates, revision number and project code
Climate	Installation altitude, average ambient temperature, peak humidity, anti-condensation heater inclusion options

Table 6.1: A summary of the physical, location and climate attributes that are associated with a switchboard installation.

through existing sections and their respective constituent components. Should the changes to the switchboard properties result in the specification of any constituent components becoming invalid, then the affected components will automatically re-evaluate their design parameters to account for the new design circumstances, thereby ensuring that the specification for the switchboard is consistent and accurate.

6.3 Component Specification

The user, having defined the general aspects of the proposed switchboard, then proceeds to edit the component specifications which serve as the basis upon which all constituent section components are created and specified to. The *component specification* dialog, depicted in Figure 6.3, appears by the user selecting the component specification icon from the tool bar or from the similarly named menu item, listed

under Edit.

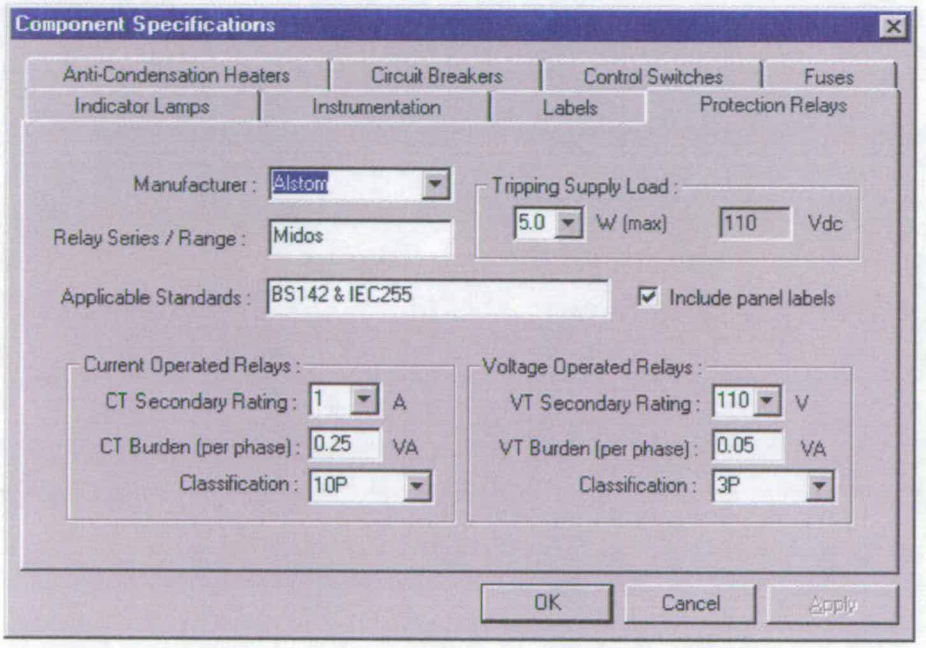


Figure 6.3: The component specification dialog with the protection relay property page displayed.

The component specification dialog is constructed in a similar fashion to the switchboard properties dialog, using a series of property pages, each of which correspond to a specific constituent component or group of components. The protection relay tab is displayed in Figure 6.3, indicating the relationships that are already established between objects. Note that the tripping supply voltage is included within this dialog as part of the tripping supply load specification for each protection relay, but the user can not edit the voltage (indicated by its "grayed out" status) it is included within the dialog for the user's information only. If the user wishes to change the tripping supply voltage, he/she must do so from the switchboard properties dialog. Any such change is automatically reflected in the component specification property pages.

Most of the options contained within the other tabbed component dialogs are self explanatory, but there are a few options that require further explanation; they are discussed as follows. Through the instrumentation tab, the user may define the general attributes for all instruments and also select and edit the type and number of instruments that are to be included within each section; the only exception to this rule being spare sections which are devoid of all constituent components. A similar option is not available under the protection devices property page due to the safety implications that such an option would pose. Therefore protection devices are predefined and installed automatically by the software to ensure that G59 protection compliance is met. The user is then free to augment these protection devices with

additional devices of their own choosing.

Furthermore, the specifications of protection and instrumentation devices are unusual as they not only define the common attributes for all protection and instrumentation devices, but also the sensing transformers to which they are connected. For all other component specifications, their respective property page relates only to the individual device in question. Note that for all sensing transformers, only the secondary ratings for the devices are present; the primary ratings and the capacity of the device will be determined at runtime once the device is instantiated and connected to the switchboard's busbars.

Also under the instrumentation property page, there is a tick box labeled "Use protection grade current and voltage transformers". This option relates to the design practice of limited capacity switchgear installations where only protection grade transformers are specified and onto which all instrumentation devices are also connected. This practice is adopted to simplify and reduce the cost of the final switchboard installation. By default this option remains un-checked resulting in both instrumentation and protection grade sensing transformers being installed, as required, within each section.

Under the climate property page, the user is asked for the altitude at which the switchboard will be installed and operated at. The number entered by the user does not have to be particularly accurate, as this parameter does not affect the electrical design of the switchboard unless it is intended that the switchboard is to be operated above 1000m above sea level. Above this height the switchgear manufacturer must be informed of the altitude so that correction factors can be applied to the intended switchboard specifications to re-calculate or upgrade the maximum voltage, current and thermal ratings of the installation.

Once the user is satisfied with the specification of the components, he/she then closes the dialog by selecting the OK button. This action results in all the component attributes being saved to the computer's memory and any previously defined section components being automatically updated to reflect the new component specifications. This measure ensures consistency in the proposed switchboard design.

6.4 Defining the Switchboard

Once the general properties and components of the switchboard have been refined to suit the installation, the user can proceed to define the sections that comprise the proposed switchgear installation. Rather than define an arbitrary example of a switchboard installation, this section will use the operational paper mill switchboard, illustrated in Figure 2.5, to demonstrate the software. It has been assumed that

the user has already entered the general and component properties for the switchboard including the specification of a current meter and voltage meter per section. Both instruments will have a selector switch to allow different phases to monitored.

To define a section, the user selects with the mouse pointer the new section icon or menu item. Either action results in the *new section selector* dialog appearing in front of the application window frame, as depicted in Figure 6.4. Note that not all the section types appear within the alphabetically sorted tree hierarchy presented to the user. In this instance, only the selection of either generator or incomer sections are valid, but this is contrary to the requirements of the paper mill switchboard whose first section is a load section, labelled Old Mill Feeder. The limited choice of section types is due to the fact that only generator or incomer sections are allowed to have sensing voltage transformers installed within their specification (this is standard switchgear design practice). However, in terms of the software model, if other sections were allowed to be defined before either a generator or incomer section was instantiated, then they could request a connection to a voltage transformer that did not yet exist within the software model – an unacceptable situation.

To resolve this limitation and to simplify the coding of the software, the user is required to define a generator or incomer section at the beginning of a design scenario, or after the definition of a bus section. In either case, this allows the voltage driven equipment in load or operationally spare sections to connect to an existing voltage transformer, or the option of specifying an appropriate voltage transformer.[2] Therefore the paper mill switchboard sections one and four, and six and eight have been transposed to allow the software to model the switchboard. The revised switchboard is illustrated in Figure 6.5. Once all of the sections of the entire switchboard have been defined, the sections that were transposed to facilitate the switchboard's definition in the software can be replaced in their original position before the final specification of the switchboard is produced.

As a result of this software limitation, the first section to be defined within the software model is in fact the fourth section of the paper mill switchboard – the incomer. The user selects the incomer section type from the tree hierarchy within the new section selector dialog, and confirms the selection by clicking the OK button at the bottom of the dialog. This action results in a dialog entitled *New Incomer Section* appearing with the leading dimensions that the user must complete to allow the section to be defined. The new incomer section dialog is illustrated in Figure 6.6, with the leading dimensions appropriate for the paper mill installation entered into the various dialog controls.

Each of the controls within the new incomer section dialog are explained as fol-

[2]The discussion in Section 7.2.3.2 indicates how this limitation may be resolved in future versions of the software.

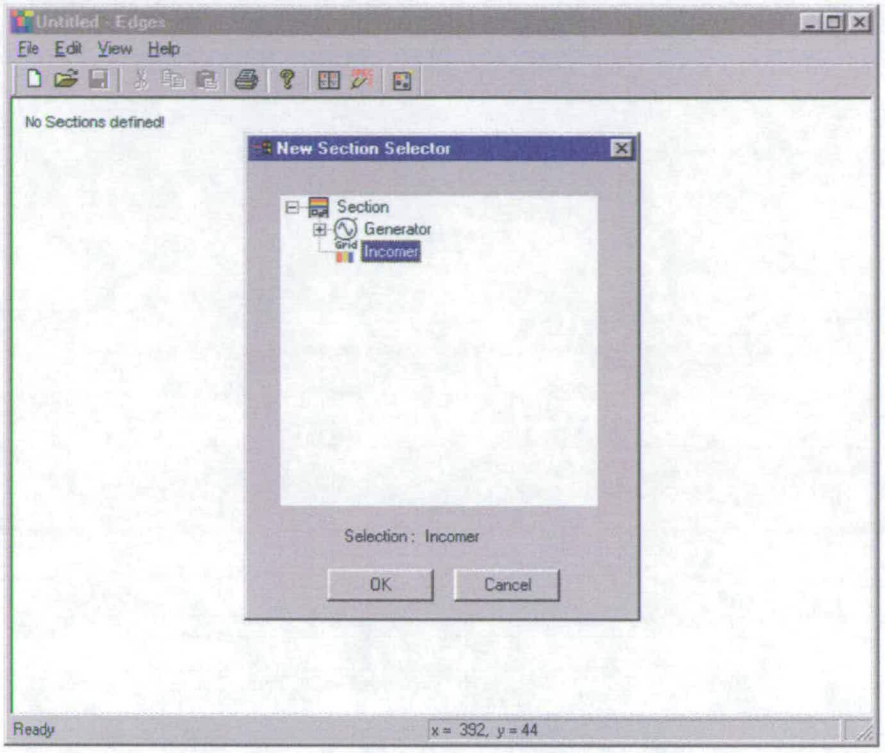


Figure 6.4: The new section selector dialog on the first visit. The tree structure allows the user to select the new section type.

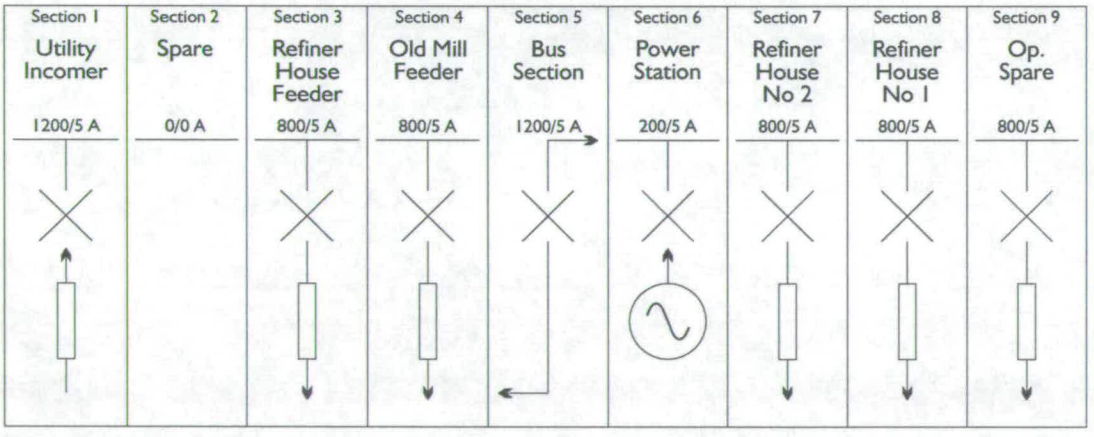


Figure 6.5: The revised layout of the paper mill switchboard to allow it to be modelled in Edges.

lows. The label edit control defines the section's label; this item is presented to the user upon the creation of all new sections. Any meaningful text string may be entered by the user. Below the label control, the user is asked to enter the capacity, line impedance and fault level of the incomer at the point of common connection (PCC). This information ensures that a set of busbars and a circuit breaker of appropriate electrical ratings, sufficient to carry and switch the incomer's load current,

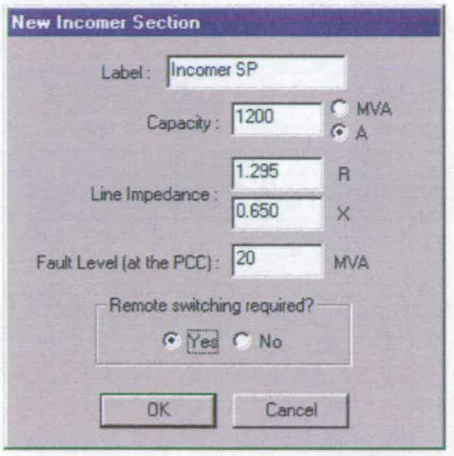


Figure 6.6: The new incomer section dialog.

are defined and specified for the section.

The last group of items on the dialog asks the user if remote switching is required; this option also appears on all new section dialogs with the exception of the *new spare section* dialog. Most switchgear installations have a mixture of locally or remotely controlled sections, therefore this question has to be presented to the user as each section is created. If remote switching is required, the following features are included automatically within the section specification:

1. A labelled local/remote control switch, and
2. A circuit breaker that includes an electrical spring charging mechanism.

If local switching is selected, then the local/remote control switch is omitted and the charging mechanism defaults to the mechanism defined by the circuit breaker component specification. In the case of the paper mill, it is likely that the incomer may be switched frequently and remotely, hence the selection of remote switching. Once all the fields are completed by the user, the inclusion of the section in the switchboard will be confirmed by selecting the `OK` button. During the instantiation of the incomer section (or any other section type), the following actions occur:

- Based upon the capacity of the section, an appropriate, standard sized circuit breaker is instantiated,
- From the circuit breaker specification, a set of busbars are specified and their rating is checked against existing busbars mounted in previously defined sections; the maximum through current between sections is determined and this defines the rating for all other busbars,

- The fault contribution from the section is calculated approximately[3] [124] and included with existing sections' fault contributions to ensure that the switchboard is within its fault rating and that of the incomer, and
- All necessary constituent components are created and, if required, connected to the appropriate auxiliary supplies or sensing transformers and, if appropriate, correctly fused.

A complete but summarised list of the components and their ratings automatically defined by the software for the incomer section is given in Table 6.2.

Item	*Rating*
Anti-condensation heater	230Vac, 40W
protected by cartridge fuse	230Vac, 1A
Busbars, set of 3 phase	1200A @ 50Hz
Circuit breaker:	
rated current	1200A @ 50Hz
rated voltage	3.3kV @ 50Hz
interrupting medium	SF_6
spring charging mechanism - motor	230Vac, 190W
spring release coil	110Vdc, 205W
shunt trip coil	110Vdc, 115W
Control Switches:	
Local/Remote control switch	230Vac, 10A
latching, selector with component label	
Trip/Neutral/Close control switch	110Vdc, 10A
latching, selector with component label	
Amp meter control switch	230Vac, 1A
make before break, latching, selector	
Volt meter control switch	230Vac, 1A
latching, selector	
Indicator Lamps:	
Busbar live lamps, 3 of,	110Vac, 5W each
with red filters	
voltage transformer properties: connection to VT1	
protected by cartridge fuses, 3 of	110Vac, 1A
component label	
Instrumentation:	
Amp meter:	
accuracy	1%
end scale value	1400A
scale	90°
scale label	A
size	96mm²
current transformer properties: connection to CT1	
burden (per phase)	2VA
component label	
Volt meter:	
accuracy	1%
end scale value	3.6kV
scale	90°
scale label	A
size	96mm²
voltage transformer properties: connection to VT1	
burden (per phase)	2VA
protected by cartridge fuses, 3 of	110Vac, 1A
component label	

Item	*Rating*
Protection Relays:	
Over current and earth fault relay:	
tripping supply load	110Vdc, 5W
current transformer properties: connected to CT2	
burden (per phase)	0.25VA
voltage transformer properties: not required	
component label	
Over voltage relay:	
tripping supply load	110Vdc, 5W
current transformer properties: not required	
voltage transformer properties: connected to VT2	
burden (per phase)	0.05VA
protected by cartridge fuses, 3 of	110Vac, 1A
component label	
Sensing Transformers:	
Current Transformer 1 (CT1):	
burden (per phase)	2VA
number of phases	3
capacity	2VA
classification (instrumentation)	1
primary rating	1200A
secondary rating	5A
Voltage Transformer 1 (VT1):	
burden (per phase)	7VA
number of phases	3
capacity	10VA
classification (instrumentation)	1
primary rating	3.3kV
secondary rating	110V
protected by cartridge fuses, 3 of	3.3kV
Current Transformer 2 (CT2):	
burden (per phase)	0.25VA
number of phases	3
capacity	2.5VA
classification (protection)	5P
primary rating	1200A
secondary rating	5A
Voltage Transformer 2 (VT2):	
burden (per phase)	0.05VA
capacity	10VA
classification (protection)	3P
number of phases	3
primary rating	3.3kV
secondary rating	110Vac
protected by cartridge fuses, 3 of	3.3kV
Section label	

Table 6.2: A complete list of the automatically specified components and their ratings that comprise the incomer section of the paper mill switchboard.

The second section of the paper mill switchboard is a spare section that only contains a set of live busbars. This section is added to the switchboard specification by once again choosing the new section selector option from the tool bar and

[3]The fault calculations assume that the switchgear installation is connected via a distribution transformer with a five percent reactance rating per phase.

selecting a spare section. The only information required from the user is to confirm the label text for the section.

The third section of the revised switchboard is the Refiner House Feeder. This load section supplies a large set of distributed induction motors via a transformer and a low voltage switchboard. The user selects the load section branch from the new section selector dialog, revealing a choice of four different load section types, as depicted in Figure 6.7. The major differences between the various load section types[4] are the use of specialised leading dimensions, where required, to assist in the definition of the section, and the calculation of their respective loading and fault contributions.[5] As the Refiner House Feeder supplies induction motors the user will select this option.

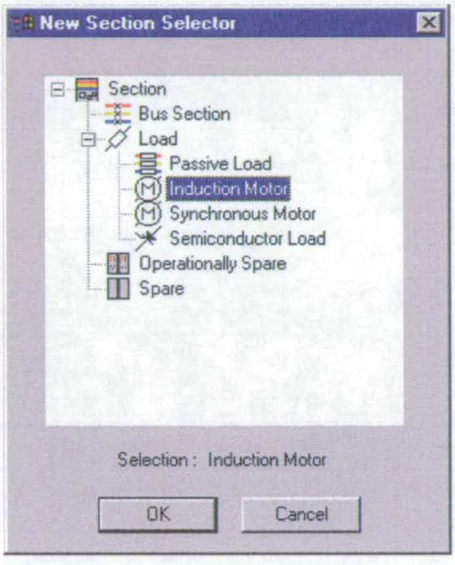


Figure 6.7: The new section selector dialog on the second visit. Note the revised tree structure.

Upon selecting the induction motor load, a dialog entitled *New Motor Section* will appear; this dialog is reproduced in Figure 6.8. The first three controls in this dialog, namely label, capacity and power factor, are identical in function to that of the new incomer section dialog. The next leading dimension requests the size of capacitor bank connected to the induction motor. This is intended for large induction motor installations where power factor correction capacitors are utilised to reduce the reactive power drawn from the supply. If the capacitor bank is large (>10%) in comparison with the capacity of the induction motor, the fault contribution of the motor becomes significant.[6] Indeed, there are recorded cases where such motors generate during loss of mains or fault conditions. In order to take account of

[4] All load sections have the same instrumentation and protection devices automatically included within their respective section specifications since their requirements for these devices are identical.

[5] Other minor differences do exist but relate to the management and control of the section objects.

[6] As discussed in Section 2.3.5.

such operating conditions, the user is asked to enter the size of the capacitor bank, if known. If the capacitor bank is greater than 10% of the motor's rating, loss of mains protection will be automatically included within the section's specification and fault calculations adjusted. This feature is specific to large, single induction motor installations. In the case of the paper mill, the distributed nature of the motor load means that the capacitor bank size can be safely defined as zero.

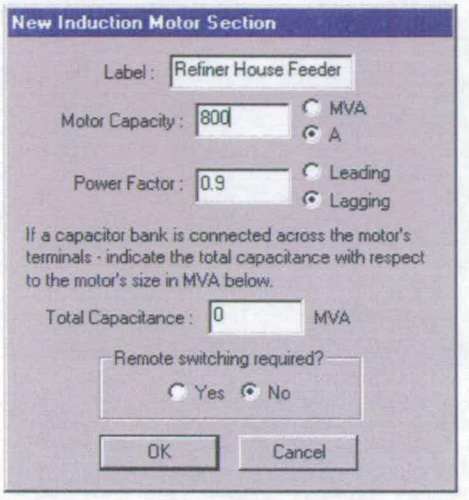


Figure 6.8: The new induction motor section dialog.

Finally the user has to specify if local or remote switching is required. In this instance, the Refiner House Feeder is switched infrequently and remote switching is not required, hence local switching is selected. Having entered the details into the various dialog controls, the user selects the OK button to confirm these details and instantiate the section.

The fourth, and also, the seventh and eighth sections of the paper mill switchboard are defined utilising an identical procedure as discussed for section three; these sections supply similar loads and are operated in an identical manner. All these sections are defined in the left to right order that they appear in Figure 6.5; they are simply omitted from any further discussion as they are included within the software model through an identical procedure to the one discussed for section three.

The fifth section of the switchboard is a bus section. By once again opening the new section selection dialog and selecting the bus section option from the section list, a bus section will be added to the switchboard specification. Bus sections require only two leading dimensions to be defined by the user; these dimensions are a text label and the selection of remote switching, if required. In this case, remote switching is not necessary, and by selecting the OK button at the bottom of the new bus section dialog, the section is added to the switchboard design. It is common in switchgear design practice that bus sections contain no instrumentation or pro-

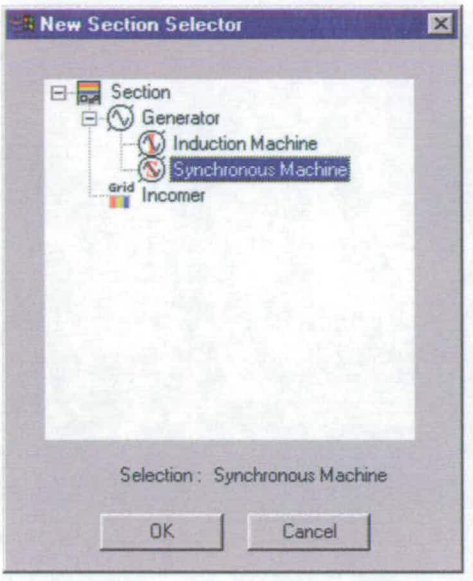


Figure 6.9: The new section selector dialog depicted about to include the Power Station section in the paper mill switchboard.

tection devices; this is reflected by the software, however, the user may add such devices manually if they are required. Note that the required capacity of the bus section is not requested. This is due to the fact that the rating of the circuit breaker included within this section is defined to be identical to the busbar rating of the switchboard. Therefore the bus sections circuit breaker can be automatically rated by the software.

The sixth section is the Power Station (an embedded generator) which in this case is a steam turbine driving a 3.3kV (1200A) synchronous machine. To include the machine within the switchboard specification, the user would select the synchronous machine option that appears under the generators branch within the new section selector dialog. This dialog is illustrated in Figure 6.9.

Having selected the appropriate generator, the new synchronous generator dialog will appear requesting the user to define the leading dimensions for the section. These dimensions include a section label, the generator's capacity and normal operating power factor, and if remote switching is required. The user will then duly enter these leading dimensions into the dialog. By default, remote switching has been selected by the software as generators are frequently switched, both locally and remotely. Upon selection of the OK button, the synchronous generator section is instantiated and included within the switchboard specification.

The synchronous generator section is the most technically and operationally complex section in any switchboard installation. Within Edges, this complexity is due to the large number of protection devices required that operate together to form

an electrical protection scheme that is G59 compliant. The constituent components, common to all other switchboard sections except spare sections, are also included within generator sections. In the case of the paper mill switchboard these components include an anti-condensation heater, a three phase set of busbars, a circuit breaker with an electric motor spring charging mechanism, two control switches and two instrumentation devices (an amp meter and a voltage meter).

In addition to the aforementioned constituent components, the Power Station switchboard section requires a set of busbar live indicator lamps and protection equipment. When the section is defined, its type and generating capacity are taken into account so that the protection scheme for the generator is compliant with G59. This also allows the protection scheme to be tailored to suit the requirements of the generator without the scheme being over-protective, hence expensive, and possibly problematic. The following protection relays were automatically defined for the paper mill generator:

- An over and under current relay,
- an over and under voltage relay,
- an under frequency relay,
- an over frequency relay,
- a loss of mains (ROCOF) relay,
- an earth fault relay,
- a reverse power relay, and
- a synchronising relay.

Note that a synchronising relay has been included in this instance but this relay would have been omitted if an induction machine had been selected. Otherwise no other differences would exist between an induction generator section of a similar power generation capacity and the above section.

All of the above constituent components within the Power Station section include, if required, correctly rated fuses (32 in total) and components labels (11 in total). Five sensing transformers and necessary connections to auxiliary supplies are also included in the section specification to allow all instrumentation and protection devices to operate. All of these components (including sensing transformers and connections to auxiliary supplies) are automatically created and appropriately specified by Edges. Table 6.3 lists all the components and their specifications included in the Power Station section.

Item	Rating
Anti-condensation heater	230Vac, 40W
protected by cartridge fuse	230Vac, 1A
Busbars, set of 3 phase	1200A @ 50Hz
Circuit breaker:	
rated current	1200A @ 50Hz
rated voltage	3.3kV @ 50Hz
interrupting medium	SF_6
spring charging mechanism - motor	230Vac, 190W
spring release coil	110Vdc, 205W
shunt trip coil	110Vdc, 115W
Control Switches:	
Local/Remote control switch	230Vac, 10A
latching, selector with component label	
Trip/Neutral/Close control switch	110Vdc, 10A
latching, selector with component label	
Amp meter control switch	230Vac, 1A
make before break, latching, selector	
Volt meter control switch	230Vac, 1A
latching, selector	
Indicator Lamps:	
Busbar live lamps, 3 of,	110Vac, 5W each
with red filters	
voltage transformer properties: connection to VT1	
protected by cartridge fuses, 3 of	110Vac, 1A
component label	
Instrumentation:	
Amp meter:	
accuracy	1%
end scale value	220A
scale	90°
scale label	A
size	96mm²
current transformer properties: connection to CT1	
burden (per phase)	2VA
component label	
Volt meter:	
accuracy	1%
end scale value	3.6kV
scale	90°
scale label	A
size	96mm²
voltage transformer properties: connection to VT1	
burden (per phase)	2VA
protected by cartridge fuses, 3 of	110Vac, 1A
component label	
Protection Relays:	
Over and under current relay:	
tripping supply load	110Vdc, 5W
current transformer properties: connection to CT2	
burden (per phase)	0.25VA
voltage transformer properties: not required	
component label	
Over and under voltage relay:	
tripping supply load	110Vdc, 5W
current transformer properties: not required	
voltage transformer properties: connection to VT2	
burden (per phase)	0.05VA
protected by cartridge fuses, 3 of	110Vac, 1A
component label	
Under frequency relay:	
tripping supply load	110Vdc, 5W
current transformer properties: not required	
voltage transformer properties: connection to VT2	
burden (per phase)	0.05VA
protected by cartridge fuses, 3 of	110Vac, 1A
component label	
Over frequency relay:	
tripping supply load	110Vdc, 5W
current transformer properties: not required	
voltage transformer properties: connection to VT2	
burden (per phase)	0.05VA
protected by cartridge fuses, 3 of	110Vac, 1A
component label	
(continued)	

Item	Rating
Loss of mains relay (ROCOF):	
tripping supply load	110Vdc, 5W
current transformer properties: connection to CT2	
burden (per phase)	0.25VA
voltage transformer properties: connection to VT2	
burden (per phase)	0.05VA
protected by cartridge fuses (VT only), 3 of	110Vac, 1A
component label	
Earth fault relay:	
tripping supply load	110Vdc, 5W
current transformer properties: not required	
voltage transformer properties: connection to VT2	
burden (per phase)	0.05VA
protected by cartridge fuses, 3 of	110Vac, 1A
component label	
Reverse power relay:	
tripping supply load	110Vdc, 5W
current transformer properties: connection to CT3	
burden (per phase)	0.25VA
voltage transformer properties: connection to VT2	
burden (per phase)	0.05VA
protected by cartridge fuses (VT only), 3 of	110Vac, 1A
component label	
Synchronising relay:	
tripping supply load	110Vdc, 5W
current transformer properties: not required	
voltage transformer properties: connection to VT2	
burden (per phase)	0.05VA
protected by cartridge fuses, 3 of	110Vac, 1A
component label	
Sensing Transformers:	
Current Transformer 1 (CT1):	
burden (per phase)	2VA
number of phases	3
capacity	2VA
classification (instrumentation)	1
primary rating	1200A
secondary rating	5A
Voltage Transformer 1 (VT1):	
burden (per phase)	7VA
number of phases	3
capacity	10VA
classification (instrumentation)	1
primary rating	3.3kV
secondary rating	110V
protected by cartridge fuses, 3 of	3.3kVac
Current Transformer 2 (CT2):	
burden (per phase)	0.25VA
number of phases	3
capacity	2.5VA
classification (protection)	5P
primary rating	1200A
secondary rating	5A
Voltage Transformer 2 (VT2):	
burden (per phase)	0.05VA
capacity	10VA
classification (protection)	3P
number of phases	3
primary rating	3.3kV
secondary rating	110Vac
protected by cartridge fuses, 3 of	3.3kVac
Current Transformer 3 (CT3):	
burden (per phase)	0.25VA
number of phases	1
capacity	2.5VA
classification (protection)	5P
primary rating	1200A
secondary rating	5A
Section label	

Table 6.3: A summary list of constituent components automatically defined by the creation of a synchronous generator section in the paper mill switchboard. A complete component specification for this section is given in Appendix B.

Sections seven and eight are load sections, defined in a similar manner as previously described for section three, and are omitted from any further discussion. The final section of the paper mill switchboard is an operationally spare section that allows the switchboard to accommodate additional loads. This section is added to the switchboard design by selecting the operationally spare option from the new section selector dialog. As with the other section types, the user must enter the leading dimensions for the section to allow its definition, which in this case is a section label, the required switching capacity of the section, and the use of remote switching, if required. Having completed these details within the new operationally spare section dialog, the user selects OK to confirm and add the section to the proposed switchgear specification.

Having completed defining the switchboard, the user may then swap sections one and four, and six and eight to return the order of sections to that of the original switchboard. It should be noted that the order of switchboard sections is of no consequence for the design or operation of the switchboard, but it does have visual or production implications for the general arrangement drawings and single line diagrams. The re-ordering operation is performed by selecting the section using the mouse pointer, right clicking and entering the number of the section with which the selected section is to be swapped with into the pop-up dialog that appears. Having swapped sections, the definition of the switchboard is now complete and the user may either save this design to disk or print a hard copy of the complete specification as shown in Tables 6.2 and 6.3. If at a future point the switchboard required further extension, this specification may be loaded from disk and additional sections added to it utilising the procedures described above.

It should be noted that the paper mill switchboard fails to utilise all the section types available within Edges. Passive, synchronous motor and semi-conductor loads, as well as, induction generators, were not required. Induction generators, however, have identical definition to synchronous machines. These load sections, if required, are defined in an identical fashion as any other section with a switchboard design, via the new section selector dialog. To include them in any switchboard specification, the user is required to enter their leading dimensions into the software; all three sections require a label, load capacity and power factor (except for passive loads where unity power factor is assumed) and the selection of remote switching, if required. In addition to these dimensions, the new semi-conductor load section dialog asks the user if the load has regenerative capability, through the use of a tick box control. If so, a reverse power relay is included to augment the section's standard protection devices and the fault calculations for the section are adjusted accordingly.

6.5 Section and Component Editing

Having defined a section, or a number of sections, the user may wish to ensure that each section's specification is appropriate to their needs, and if not, the user may wish to make some minor adjustments to the specified components where they see fit. Such operations are easily performed at any point during the design of a switchboard by selecting a section and placing the mouse pointer within the gray rectangle of the section to be edited and clicking the left mouse button. This action results in the *edit section* dialog appearing, as illustrated in Figure 6.10. From this dialog, all aspects of the section's configuration may be viewed and certain aspects of all constituent components edited.

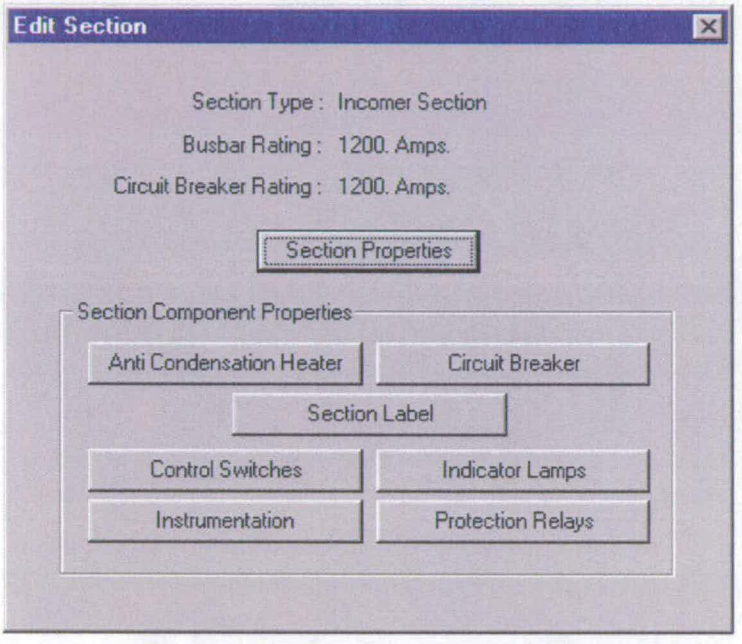


Figure 6.10: The edit section dialog.

The first three items of the section edit dialog display the type of section being edited, the present busbar rating for the switchboard and the rating of the circuit breaker included within the section. The user may then select one of eight buttons to allow the viewing and/or editing of the various aspects of the section's properties. The selection of the section properties button causes a dialog to appear that allows the user to edit the overall electrical properties of the section, such as the capacity and power factor of the section, as well as any specialist attributes associated with the section; for example, in the case of an incomer section, the fault level at the point of common connection. If the user edits any of these attributes the software automatically re-evaluates the circuit breaker's capacity and the switchboard's busbar ratings, re-sizing these components if required.

Below the section properties button are a group of buttons collectively labelled

section component properties. These allow the user to edit existing components automatically defined within the switchboard or augment these devices at the user's discretion. The editing of these devices will be discussed with respect to the number of instances of each device within an individual section. Only one instance of an anti-condensation heater, a circuit breaker and a section label may be defined due to the physical layout of a switchboard section. Therefore, if the user selects one of these devices to edit, they will be presented with an *edit component* dialog that allows the user to selectively edit the attributes of that device. An example of such a dialog is given in Figure 6.11 for a circuit breaker. Such component editing allows the expert user to tailor the requirements of individual devices specific to the section in which they are mounted.

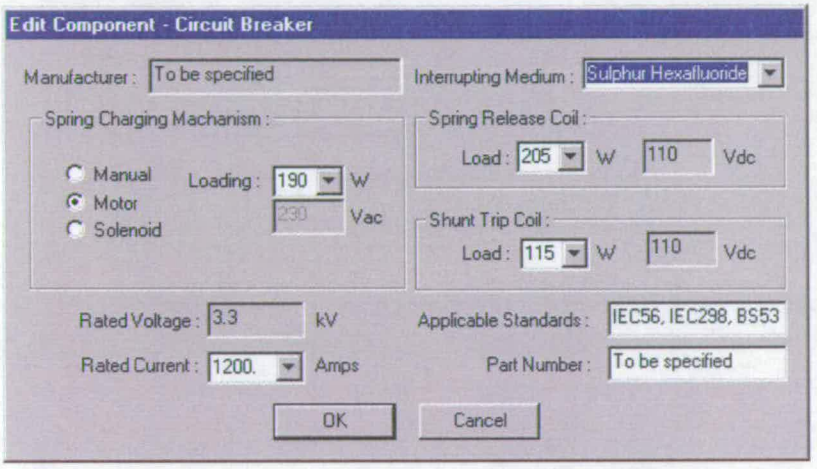


Figure 6.11: The edit component dialog for a circuit breaker.

The remaining four types of constituent components, namely control switches, indicator lamps, instrumentation and protection relays, may have multiple devices included within each section. Therefore the user must select which device within a functional group of components they wish to edit. This is performed through the components' respective *device editor* dialog. Figure 6.12 depicts the device editor dialog for protection relays, although the device editor dialogs for control switches, indicator lamps and instrumentation devices are identical except for appropriate dialog title and button text label changes. The check box to the left of each protection device allows the device to be either included (denoted by a tick mark) or excluded from the final switchboard specification. By selecting a device with the mouse pointer, indicated by the device name being highlighted, the user may edit the device by selecting the edit device button, resulting in the edit component dialog appearing. To illustrate this feature, the under voltage relay edit component dialog has been depicted in Figure 6.13.

In Figure 6.13, the under voltage protection relay is connected to a three phase voltage transformer. As this device does not require connection to a current trans-

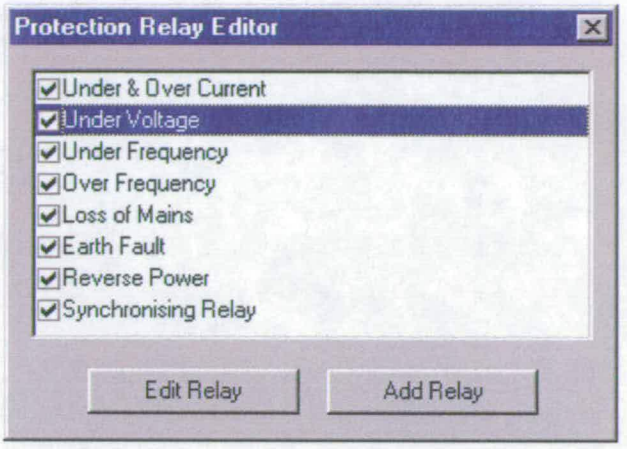


Figure 6.12: The protection relay editor dialog. The editor dialog lists the protection devices automatically defined for a synchronous generator section.

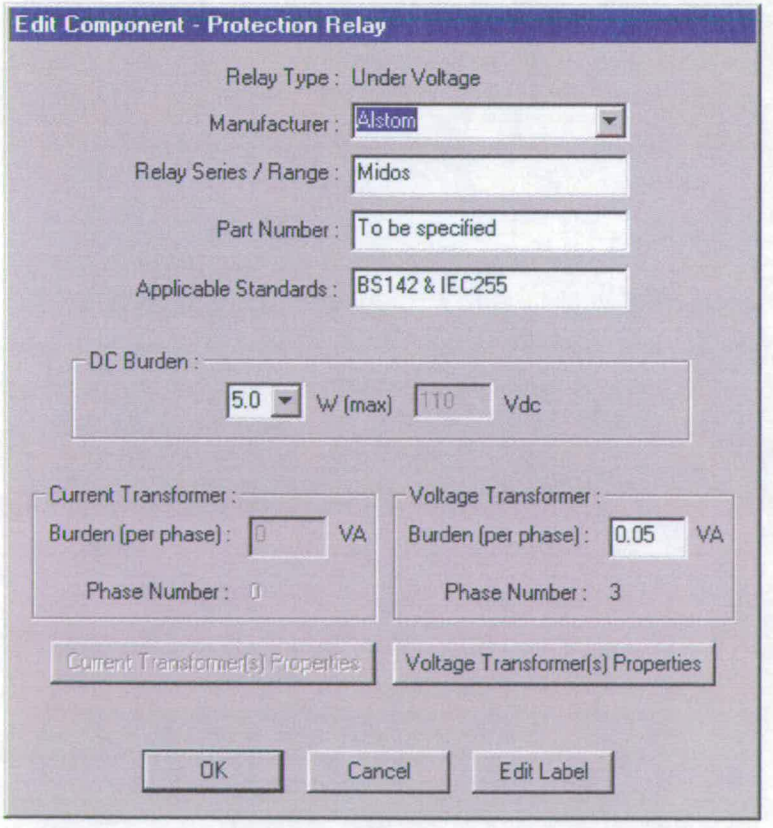


Figure 6.13: The device edit dialog for a protection relay.

former, the dialog controls for this device are inoperable. However, if the user wishes to view a summary of the voltage transformer's specification to which the protection relay is connected, he/she selects the voltage transformer(s) properties button, resulting in the similarly named dialog appearing; this dialog is shown in Figure 6.14. Note that from the *voltage transformer properties* dialog, the user may also view and edit the fuses that protect the primary windings of the voltage transformer.

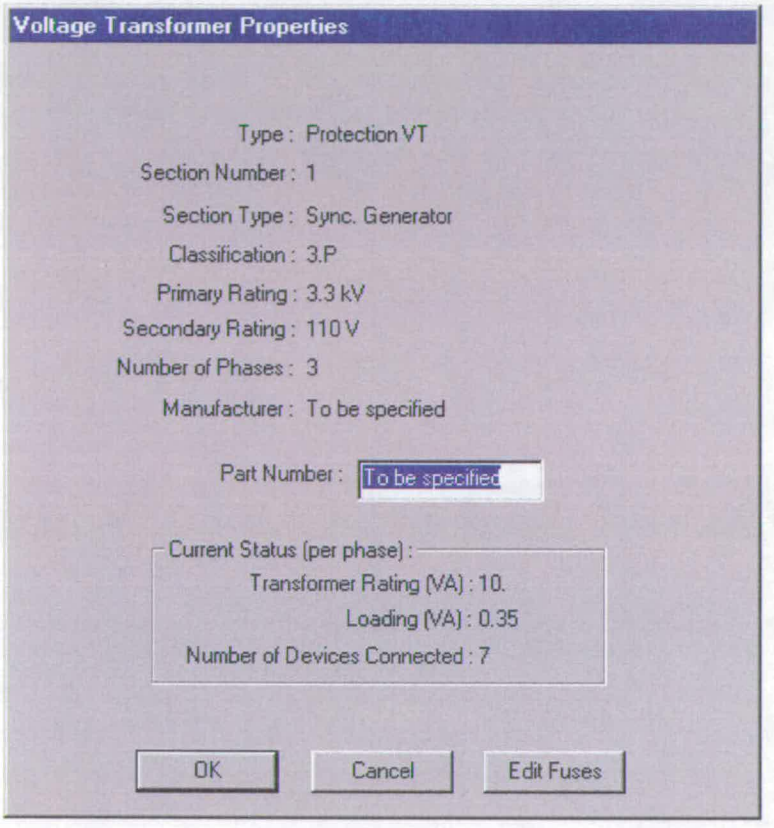


Figure 6.14: The voltage transformer properties dialog for a protection grade, three phase transformer.

Returning to the device editor dialog, the user may, alternatively, wish to add a device not defined automatically by the software. This operation is performed by the user selecting the add device button on the bottom right of the editor dialog. In a similar fashion to the definition of switchboard sections, the leading dimensions of any new device must be entered by the user via a specialised dialog. The dimensions required to define a control switch, indicator lamp, instrumentation device or protection relay are summarised in Table 6.4.

Having completed viewing or editing components within a component editor dialog, the user may return to the section edit dialog to check or modify any of the other components within a section. Once the user is satisfied with the specification of a particular section, they may close the section edit dialog and select another

Component	*Leading Dimensions*
Control switch	Function
	Label
Indicator lamps	Function
	Cover colour
	Connection to voltage transformer or auxiliary supply
	Number of lamps required
Instrumentation device	Function
	Scale label
	Number of current transformers required
	Number of voltage transformers required
Protection relay	Function
	Number of current transformers required
	Number of voltage transformers required

Table 6.4: A summary of the leading dimensions for multiply defined section components.

section to edit using the mouse pointer. All component changes are re-checked by the application to ensure the switchboard specification is consistent and correct. The switchboard specification may then be saved to disk and/or printed. This completes the discussion of editing section or component properties within Edges.

6.6 Chapter Summary

This chapter has indicated that through the use of the object oriented programming paradigm, an artificial design application for multi-component design may be constructed. By utilising the features of the C++ programming language, an architecture may be constructed that allows common design strategies, such as top-down design, to be modelled in a flexible, extendable framework. This software framework has been demonstrated for embedded generation switchgear design. The operation of Edges meets the criteria deemed necessary for the software to be of practical value. Due to the flexible nature of object oriented programming, the functionality captured within the software may be modified or extended to meet future switchgear design practice. The description of the operation of the software given in this chapter is not complete as not all the features of Edges have been included (such as error messages or user prompts), however, the key facilities of the software have been discussed, sufficient to allow the overall operation of Edges to be clearly indicated.

Chapter 7

Discussion and Conclusions

The objective of this project was to investigate the use of object oriented programming techniques to construct a software application that will allow the design rationale of multi-component systems to be modelled. The problem domain selected to examine if such a software architecture could be developed was embedded generation plant. The issues raised during the development of this project fall into four areas: the issues associated with the operation of the project software, development issues for the software, the benefits associated with the use of the object oriented programming paradigm for artificial design, and finally, the implications of this approach for use within other design domains. This chapter critically discusses these four categories of issues, from which conclusions are drawn as to the success of the work. The chapter finishes by suggesting future research, overall conclusions and a thesis summary.

7.1 Software Operation Issues

This section considers the issues associated with the operation of the project software, Edges. The case study presented in Chapter 6 illustrates the ability of Edges to perform switchgear design reasoning, producing a complete and accurate specification for the paper mill switchboard. As with any man made artefact, there are several operational issues that exist within the software. These aspects of Edges are discussed below.

7.1.1 Overall Operation

The software functions as originally envisaged, allow the definition and design of an entire switchboard to be easily performed utilising a graphical user interface. Each section automatically receives appropriate discrete electrical protection

that is compliant to G59. Should G59 be revised or superseded, however, the software can easily accommodate other protection schemes. External features, such as auxiliary power supplies, have also been included within the model. Constituent components necessary for the switchboard's operation are individually modelled along side supplementary components that assist in the switchboard's operation and improve its reliability. Information concerning the design of all components is derived from the switchboard properties or component specification objects. In cases where such information is not available, the user is requested to provide this information via a specialist dialog. Once completed, the switchboard specification may be printed.

In its present form, Edges only produces specifications for the electrical switchgear common to the majority of embedded generation installations. Other areas of embedded generation design, such as hydro or wind turbine and generator selection or combined heat and power plant design, or indeed any multi-component system, could be modelled by utilising the object oriented software architecture developed in this project. Any such extension to Edges' functionality would require the design process for that problem domain to be carefully studied, rationalised, coded and tested. However, such additional design functionality is well within the capability of the software architecture described in this thesis. If the design rationale included within Edges were extended to cover other design aspects of embedded generation, it would not effect the existing operation of the software unless specifically made to do so.

The specification data produced by Edges would be considerably enhanced by the automatic production of electrical switchboard schematics. By the inclusion of drawing functions within each design class, the production of schematics could be realised without affecting the existing design capability of the project software. However, some form of drawing layout management would have to be developed and coded to ensure that the schematics produced where legible – a task which is non-trivial.

An experienced, professional engineer may produce switchgear specifications that are superior to those presently produced by Edges. However, due to the escalating complexity of modern technology, to which switchgear is no exception, specialist design software such as Edges will play an increasingly important role. Such software will greatly assist the management of complex, repetitive design problems, allowing the designers of such systems to concentrate on the critical design issues associated with individual projects. The result of developing and utilising specialist design software is that specifications for artefacts will take shorter periods of time to produce, be more accurate and complete.

7.1.2 Over Design

By assuming high voltage design practice throughout the software, there will be a certain element of over design for low voltage installations. This will only occur in certain cases, especially as the total electrical generation capacity of such installations decreases. In cases where gross over design would occur, it would be normally accepted that a bespoke single section switchgear installation with limited G59 protection[1] would be purchased and installed. Embedded generation schemes that fall into this limited generation capacity category are currently uncommon since the costs of protection and connection to the distribution network makes them un-economical.

7.1.3 Editing Constituent Component Specifications

At present, if the user edits the specification of an constituent component via the component specification dialog,[2] all component specifications that have been edited are automatically updated. This behaviour has been selected to ensure consistency within the switchboard specification is guaranteed. Unfortunately this action has the side effect of deleting all individual component edits that the user may have performed prior to the component specification dialog being opened and specification information edited.

This behaviour of Edges is a limitation. A possible solution to this issue could be to present a warning to the user of the benefits or pitfalls of either approach during the software's use via a dialog. The user may then apply their discretion to either:

1. Accept the component specification modifications that they have made and update all similar component types, or
2. Ignore any component specification edits that they have made and perform these specification edits individually via the section edit dialog.

Only components whose specification have been edited within the component specification dialog are affected by this software feature. Un-edited specifications are not permeated through the existing switchboard design.

[1] Small embedded generation installations may enter into a discussion concerning the protection arrangements of the site with the local public electricity supplier. Such discussion can result in the public electricity supplier relaxing the protection requirements necessary to fulfill G59, dependant upon the local connection arrangements [280].

[2] Refer to Figure 6.3.

7.1.4 Protection Equipment

Presently, Edges only allows individual or discrete protection devices to be utilised in the design of a switchboard. However, many modern switchboard installations utilise multi-function protection equipment that also includes instrumentation functionality; see for example the Alsthom LGPG111 relay [7], the General Electric PQM[3] relay [93] or the Sepam range [105] of protection devices. Multi-function protection devices could be included within the software by coding a protection device function supervisor which could be incorporated into all section class definitions or be implemented as a separate object.

By including appropriate interface controls within the component specification dialog, the user could enter the functions that the multi-function protection device monitors, *e.g.* under and over voltage, earth fault, and loss of mains. These protection functions would be stored using the `m_function` member variable that already exists within `CProtection` objects. During the definition of a section, the multi-function protection device, if defined, would be selected in precedence over discrete protection devices. However, the protection engineer would ensure that any protection requirements not met by the multi-function protection device would be fulfilled by discrete protection devices, thereby ensuring that the protection for all switchboard sections was compliant to G59 regulations.

It should also be noted that, by default, directional overcurrent and neutral voltage displacement protection devices are omitted from automatic definition in the software. As both protection devices are are not mandatory under G59 and frequently they are not required due to the electrical conditions within which embedded generation operates, their inclusion within a switchboard design has been left to the discretion of the user.

7.1.5 Computing Resources

During execution, Edges does not rely upon external databases or third party applications for its operation. As a result, Edges does not require large amounts of memory, hard disk storage space or extremely powerful processors to execute. When initially started, Edges requires approximately 2.5 MB of random access memory to operate. This efficient use of memory is due to the dynamic memory management employed by the software. As a consequence of using dynamic memory management additional random access memory is required as each additional section is defined. To illustrate this point, after the definition of the paper mill switchboard,

[3]PQM is a acronym for Power Quality Management

as depicted in Figure 2.5, Edges occupied an additional 0.7 MB of memory. The efficient use of computing resources means that the software will operate on virtually any modern personal computer.

7.1.6 Software Maintenance

The project software in its current form requires no additional maintenance to ensure its operation. Access and use of all the facilities that Edges offers requires no programming skills, only familiarity with Microsoft Windows' graphical user interface features and controls. However, the modification or extension to the software's functionality will require the following skills:

1. A thorough understanding of the C++ programming language and the object oriented programming paradigm, and,
2. Familiarity with Microsoft's software application development tool, Developer Studio, and Microsoft's Foundation Classes.

Embedded generation developers with sufficient switchgear design experience or power engineers from the switchgear manufacturing sector with these software skills are not readily available. It is likely that the continued development of artificial design tools, especially for switchgear, will be developed by companies with sufficient skills and financial backing to warrant the significant cost and time resources required for their development, or by academic institutions.[4] Such companies are likely to be either medium or large scale enterprises or specialist consultancy based companies.

7.2 Software Development Issues

This section discusses issues that will impact upon the future development of the project software. Consideration is given to the economic impact of artificially intelligent design software, such as Edges, upon the electricity supply industry if it were adopted, developed and utilised. The possibility of porting the code that implements Edges to other computing platforms is then considered. Finally, the discussion examines the overall flexibility of the existing code base and that of the software's design architecture.

[4]Further discussion on this topic is given in Section 7.2.1.4.

7.2.1 Economic Considerations

As indicated at the beginning of the thesis,[5] the three key motivations for developing Edges was to:

1. Minimise the time taken to produce a switchboard specification,
2. Increase the accuracy of proposed switchboard specifications, and,
3. Devise a rationalised, effective design process for embedded generation switchgear design.

Each of these objectives are discussed below.

7.2.1.1 Efficient Design

As the software utilises a standard graphical user interface, developers will be able to access the full potential of Edges quickly and easily. The learning curve to use the software is very shallow in comparison with learning to design switchgear installations and the interpretation of the electrical regulations associated with embedded generation.

Including typical design parameters for the general properties and constituent components of a switchboard assists and guides users to the appropriate specification data required for their site. If such artificial design software systems were standardised and widely adopted by the switchgear manufacturing industry, individual manufacturers could issue prospective embedded generation developers with data files, via the internet. Such data files would contain the general properties and specification components which each individual manufacturer supplies. Therefore developers could define the switchboard they require and adjust the default section's specifications to meet their requirements (dependent upon the users experience). Once the switchboard specification has been finalised, the complete specification may be electronically returned to the manufacturer with appropriate tender or purchase order details.

Furthermore, the software rationalises the entire switchboard design process. This rationalisation process frees the developer from the tedious task of specifying and checking individually each switchboard component. Embedded generation developers may then focus upon the overall operation and function of a proposed switchboard and on the non-standard features of the installation.

[5] As detailed in Section 1.1.

7.2.1.2 Reduced Design Durations

The automation of switchgear constituent component specification, with the automatic evaluation of component ratings, combined with the auxiliary supplies' dynamic load information, allows the specification of a proposed switchboard to proceed with great speed in comparison with current techniques. A new switchboard installation can be specified in a matter of tens of minutes with the use of Edges, as opposed to several days or even weeks depending on the switchgear designer's experience and technical knowledge (especially in relation to G59). Such a reduction in design duration equates to substantial savings in time and, in turn, site development costs.

7.2.1.3 Increased Accuracy

Since each constituent component object is aware of its requirements both in terms of standard ratings and external connections or interdependencies, the interaction between objects within the software model ensures that the final switchboard specification is not only functional and complete, but, most importantly, accurate. The ability of designers to deal with complex domains, such as switchboard design, is limited[6] in comparison with modern electronic computers and programming paradigms. Thus Edges is more adept at tracking and checking the implications of a leading design dimension change than a human practitioner. The enhanced accuracy of the software over and above traditional switchgear design techniques, would, if the software were adopted, result in the risk and financial implications associated with inaccurate switchgear specifications being substantially reduced.

7.2.1.4 Future Development

It is also worth considering at this juncture the future development of artificial design tools, such as Edges. It is questionable which sector of the electricity supply industry would fund the continued development of such software for artificial switchgear design. The main beneficiaries from the development and deployment of switchgear design tools are embedded generation developers, as they are able to quickly produce switchgear specifications that are accurate and complete.

Therefore it is in the interest of embedded generation developers to support the development of software tools similar to Edges. However, the majority of developers do not possess the necessary programming skills and specialist switchgear design knowledge to further the development of such software, or indeed, possess

[6] As indicated in Section 3.3, based upon Miller's research [177].

the financial resources required.[7] Switchgear manufacturers or specialised power engineering consultants do have the resources and the possibility of recruiting the required programming skills to develop such tools. Unfortunately, both parties lack the financial incentives to do so as, if such a tool was adopted and proved successful, they would be reducing (or even eliminating) a valuable source of revenue – the consultancy related to new embedded generation sites. It is therefore likely and unfortunate that without legislative, regulatory or governmental assistance software for switchgear design, such as Edges, that could benefit the entire industry will remain an academic curiosity.

The probability of artificial design systems being adopted throughout an entire industrial sector are slim. However, the development of such software for use within individual companies is a significantly more likely prospect. The function of artificial design software within a company context would be to enhance sales of the company's products. Such software could significantly reduce the time taken to determine a customer's requirements and produce a complete specification, especially for complex products or services, thus allowing a competitive edge to be gained. Furthermore, specifications produced would be of a consistent and accurate standard. Indeed, third party companies could utilise such software to offer services outside of their area of expertise but closely related to their own range of products. For example, a company who manufactures hydro turbines could offer to design and specify a suitable generator and electrical switchgear by utilising an enhanced version of Edges, hence allowing the company to offer a turn-key service.

7.2.2 Portability

The distribution of the software in its current form to interested parties is straight forward. Due to the appropriate selection of compiler options available within Developer Studio, one executable file[8] includes all the Windows resources and programming code required to allow the software to execute on any Windows operating system. A simple install routine or application, if deemed necessary, could be easily developed to allow the automatic installation of the binary file and the placement of an icon included either on the Windows *desktop* or in the *Start Bar*.

The selection of Microsoft Windows as the target operating system remains appropriate. Other operating systems are utilised within the electricity supply industry but are restricted to specialist applications. The vast majority of embedded generation developers and switchgear manufacturers will possess, or have access to, a personal computer that uses the Windows operating system. However,

[7] As such financial resources would be a considerable burden for an individual developer in comparison with the benefits that such software bring.

[8] The size of the Edges binary file is approximately 600 kB.

since Edges is currently a self contained object oriented application it could be distributed directly over the internet by utilising the platform independent object oriented programming language, Java [189]. The development of such a version of Edges would be an extensive task, requiring the complete re-development of existing code, but a task that could be quickly achieved (within several months) due to the rationalised object oriented design architecture developed in this project. It is through this knowledge that other design domains may be modelled utilising any object oriented programming language, such as Java. Further investigation of the object oriented feature set of the Java language would be required to ensure that all language features used in coding the software with C++ have Java equivalents. The construction of graphical user interface control and features would also have to be explored.

7.2.3 Flexibility

The flexibility of the project software may be sub-divided into two distinct but interdependent categories: the flexibility of the existing code base and the flexibility of the object oriented design methodology, or architecture, implemented by the computer code. Both of these categories will be discussed below.

7.2.3.1 Programming Code

The review of software programming languages conducted in Chapter 3 indicates that the object oriented programming paradigm is the most flexible computer coding scheme yet developed. The experience gained from implementing this project confirms this to indeed be the case; all of the code reuse strategies within object oriented code have been utilised.[9] The specification and design classes that implement the switchgear design process have been deliberately developed to allow the greatest flexibility to be achieved. The components that constitute the design domain are modelled utilising individual classes, with increasingly complex but functionally similar components being defined in terms of more rudimentary components through the inheritance mechanism. However, to ensure that the software developed is flexible, the relationships between interrelated but dissimilar components are represented through the use of the aggregation mechanism.

New classes may, therefore, be easily added to the existing class hierarchies by an experienced programmer to allow new switchboard components to be defined. Such new components may be defined in terms of the specialisation of existing components through the use of inheritance, or by the incorporation and arrangement of existing components by utilising the aggregation mechanism. If a

[9]The strategies for code reuse are briefly described in Section 5.1.4.5.

choice between inheritance and aggregation occurs, then the aggregation mechanism should be attempted first thereby maximising the flexibility of the software. Either approach allows existing code to be utilised and extended to suit new design rationale. If existing specification or design classes prove to be inflexible, then their class definitions may be re-implemented or augmented to fulfill new design processes. This editing of classes can occur without affecting the operation of the rest of the application as long as the original object interface is maintained.

Design classes may even be completely removed should their functionality prove redundant. For example, if modern multi-function electronic protection instruments were to become standard, discrete protection devices could be easily removed from the switchgear design framework without affecting other, unrelated components. The object oriented design architecture also allows for component developments to be easily included within the existing software, as illustrated by the following example.

Modern electronic measurement instruments require connection to the switchboard's auxiliary supply as well as to sensing transformers. At present, all instrumentation devices are defined only to connect to the required current and voltage transformers. This is in line with current design practice, as the majority of instrumentation devices derive all their operational power from the sensing transformers. Electronic instrumentation devices can be easily included within the model by deriving new design classes from the `CInstrumentation` class that include a `CPowerSupplyLoad` object within the class definition. Re-definition of the multi-argument constructor for the new class will also be required. The new instrumentation devices may then be utilised within the application by the modification or creation of appropriate dialogs; a task that can be easily achieved through Developer Studio. Furthermore, the verification of new or edited objects need only be concerned with the operation of the objects in question and their interaction with related objects. All other independent objects do not need to be tested; an advantage, especially over expert system based design software.

7.2.3.2 Design Methodology

The flexibility of the design methodology implemented by the software is an important aspect of the project. The rationalised process for switchgear design[10] will, as a matter of course, require modification; design styles, switchboard components or regulations change, and these changes must be reflected in the software if it is to continue to be of service. Such developments may be accommodated by the object oriented approach and the design architecture within which the software is coded.

[10] As presented in Chapter 2.

Several limitations currently exist with the model as it currently stands; these are discussed below.

Presently, Edges uses generalised representations of the components that comprise the switchgear design domain, thereby allowing the functionality of these components to be captured and modelled directly utilising the language features of C++ without the need to rely upon an external data or knowledge base. This approach, unfortunately, does have a limitation. The software relies upon a purely functional design rationale. The design process modelled by the software can only perform design reasoning based upon the physical or functional requirements of the switchboard and its constituent components.

As a result, the software model includes only a limited proportion of the design knowledge that human practitioners draw upon as a basis to perform design reasoning. Switchgear designers will base their design decisions upon other component attributes or factors to optimise the final design to include not only the functional features that the switchboard must possess, but also to include one or more additional design objectives; a process referred to as *multi-faceted design*. Such objectives include, for example, the optimisation the switchgear specification for size or weight constraints, the minimisation of component numbers or, most importantly, the management of switchboard costs. To model multi-faceted design rationale requires a choice of each component type to be accessible and available to be queried within the software. The storage, organisation and searching through such a library of components can be achieved by employing database technology.

Therefore a database of switchboard components would be required that conform to the generalised component specifications already defined within the software. However, these components would extend the generalised component attributes to include supplementary information required to allow additional design reasoning to be performed by the software. The design rationale would be extended to take account of and utilise the database of components by including within the software additional member functions within existing design classes. The inclusion of such a database would be in effect extending the design features of Edges a stage further towards realistic design.

It should be noted that an object oriented design hierarchy combined with a database is not the only software technology that allows multi-faceted design to be modelled. However, it is the most flexible, adaptable and maintainable software technology within which such design software can be written. Due to the inherent volatility of the majority of design processes, this is a very important feature to consider when selecting an implementation technology for a software design application.

Expert systems, if programmed appropriately, can exhibit multi-faceted design

rationale, but such applications are difficult to devise, code and maintain. The success of design applications constructed utilising case based reasoning depends upon an organised library of previous design cases. If the library of previous cases exhibits instances of multi-faceted design and algorithms to resolve conflicts, then such applications could be described as performing multi-faceted design reasoning. Genetic algorithms may also perform multi-faceted design if suitable fitness functions can be coded to optimise for conflicting design criteria simultaneously. Finally, intelligent agents may certainly model multi-faceted design, as they are able to resolve conflicts and explore alternatives by their very nature. Unfortunately they require substantially more resources to develop in comparison with the object oriented design approach.

7.3 Object Oriented Design Issues

The selection of the object oriented programming paradigm was a central issue in this thesis. Having implemented an object oriented artificial design system, Edges, this section considers the implications associated with utilising this technology.

In general, design processes may be classified as either a rational, routine process based upon scientific reasoning, or a creative, inventive, and innovative process.[11] Researchers within the design methodology field have attempted to unify these contrary views on design; as yet, to no avail. However, in attempting to do so, the gap between these opposing or diverse positions on design methodology has narrowed and this convergence will certainly continue into the foreseeable future.

It was the combination of design domain rationalisation with expert system technology, enabled by the availability of inexpensive, powerful personal computers, that allowed the first generation of artificial design systems to be developed.[12] However, it was also the inflexibility and high maintenance of these systems,[13] combined with advances in programming languages and artificial intelligence techniques, that have facilitated advanced, second generation artificial design systems to be developed.

The key to developing any successful software is the containment and management of complexity. As the problems attempted by computational means become increasingly complex, the tools utilised to encode the problem domain within a piece of software must also be sufficiently powerful to keep pace. This extension can be observed through the development of programming paradigms.[14] The advent of sophisticated programming paradigms also stimulates the development

[11] As indicated in Section 2.2.1

[12] Refer to Section 3.2.1.1.

[13] As discussed in Section 4.4.1

[14] Refer to Section 3.3.

and construction of artificial intelligence techniques that embody ever advancing and complex behavioural patterns.

The design process for all but the most trivial of artefacts is a complex process which may be captured and successfully managed within object oriented software by one of two key language features:

1. By increasing the number of member functions within a class definition, thus allowing the class to manage a greater number of design responsibilities and-/or behaviours, and
2. By the inclusion of a greater number of classes, hence allowing additional aspects of the design domain to be incorporated within the software.

This ability to manage complexity, coupled with the flexibility of the object oriented programming paradigm has been eloquently summarised by Daniels [65]:

> *"The goal is to build in [object oriented] software a richer representation of the problem than is needed to meet the immediate needs, so that the software can adapt as needs change."*

The design process for all artefacts is open to change, and through the use of object oriented software artificial design systems can be constructed that allow such changes to be accommodated without having to re-code large sections of the entire application. Such flexibility includes:

- The revision of an object's behaviour by re-coding it's implementation,
- The extension of an object's behaviour through the use of inheritance (and inclusion of supplementary member functions and variables), and
- The creation of increasingly complex objects from the combination of existing objects, through use of the aggregation mechanism.

It is the ability of object oriented software to manage complexity and change that makes it a suitable software technology for the construction and development of artificial design systems. This is also reflected in the continued application and development of object oriented design software systems [26, 101, 149, 237]. However, the appropriate or correct operation of such software depends solely upon the understanding of the programmer of both the design domain and the object oriented programming paradigm.

The design domain described in this thesis focuses upon a rational and logical design process. The development of Edges illustrates that the switchgear design process can be rationalised and automated utilising the object oriented programming paradigm. However, the application of object oriented programming to rational design processes does not exclude this technology from being utilised to model the creative aspects of design domains. It is likely that in the near future, creative artificial design software applications will be developed that combine the powerful language features of object oriented systems with other artificial intelligent technologies. For example, an object oriented framework combined with an artificial neural network could allow certain aspects or choices within a design domain to be learned as a design scenario proceeds. Alternatively, the use of genetic algorithms within an object oriented framework could explore certain aspects of a design domain that can be expressed in terms of a fitness functions. No matter which artificial intelligence technology is utilised, an object oriented framework will provide the underlying structure to drive the process to a successful conclusion.[15]

Although the object oriented paradigm is a powerful approach to computer programming, it remains one of the most difficult software technologies to apply in practice. The creation of a functional, flexible and extendable class hierarchy remains a non-trivial task, even with the advent of multiple object oriented design methodologies.[16] The appropriate development of object oriented systems requires experience before suitable class arrangements for a problem domain become apparent. However, once established, the class arrangements or *patterns* defined may be re-applied within other domains [92].

The adoption and development of object oriented programming techniques has become widely accepted in mainstream commercial software development. This trend is set to continue, accelerated by the development of electronic commerce where the use of object oriented relational databases and information technology appliances are becoming heavily integrated to create increasingly complex, electronic systems. The object oriented paradigm allows the complexity of these systems to be overcome, managed and maintained.

7.4 Artificial Design Issues

The prior discussion has focused upon the operation and the object oriented aspects of the software. This section considers the developments described in this thesis in

[15]Object oriented programming languages include many features associated with frame-based expert systems. These include the ability to represent structured knowledge and inheritance. As a result, object oriented languages are frequently used to code artificial intelligent applications [266, 159].

[16]Refer to Section 4.4.8 and Appendix A.

the larger context of artificial intelligence and design.

7.4.1 General Design Knowledge

The development of Edges illustrates that narrow design domains can be captured within a software application. The experience gained through this project confirms Steinberg's assertion [256] that:

> *"Only a small fraction of design knowledge will apply to* ***all*** *domains, and that knowledge will be at a very abstract level and thus hard to apply to any given specific task."*

It is for this reason that no computer code library (implemented in any language) of common design knowledge has been created to assist researchers develop artificial design tools.[17] As exemplified by the development of this project, any such common design knowledge is simply coded, as required, within the context of the main problem domain, in this case switchgear design.

7.4.2 The Capturing of Design Knowledge

The field of artificial intelligence has developed many structures that are classified as possessing some discernible level of intelligence. The artificial intelligence community typically represents knowledge utilising one of two methods [185, 159]:

Explicit Representations: Systems that utilise this technique represent knowledge within carefully designed structures with search or goal driven algorithms to implement intelligence. Examples of this knowledge representation technique include expert systems, case based reasoning and genetic algorithms.

Connectionest Networks: The construction of specially arranged networks, constituted of highly interconnected, independent elements, that allow intelligent behaviour to emerge from the network. Examples of this knowledge representation technique include artificial neural networks and intelligent agents.

The project software, Edges, embodies both these knowledge representation techniques in order to perform intelligent design reasoning. Explicit knowledge is stored within an object's member functions allowing design rules specific to certain

[17] Common design knowledge that could be represented in such a library could include basic physical laws, geometric relationships or fundamental economics.

aspects of the design process to be captured. An example of this approach includes the determination of standard component ratings. The inter-relationships between objects and their interaction at runtime, combined with their explicit knowledge allows the collection of design classes to perform intelligent design reasoning within the switchgear problem domain. If considered in isolation, individual design classes would not be classified as possessing intelligence. However, their interaction with other design objects and specification objects allows intelligent decision making to be exhibited through the network of object interactions.

7.4.3 Responsibility

Throughout the development of the project software, one of the underlying goals of this research has been to attempt to explore and enhance machine intelligence but not to the detriment of human intelligence. Ormerod *et al* [201] argue that their software is aimed not as a replacement of designers, but as a tool to assist them handle and reuse information. Edges has been developed in a similar spirit; to allow users to focus on the crucial aspects of a switchgear installation while the software takes care of the details. The main objective of Edges is to assist switchgear designers, not replace them.

7.4.4 Application

The development of object oriented artificial design software requires considerable effort and expertise (in both the design application domain and object oriented programming). The number of individuals or groups who possess the skills necessary to develop such software is very limited, therefore the financial costs associated with the development of such software is considerable.

At present, Edges does not have the necessary communication functionality to obtain or store product information, either from an in-built database or directly from the internet. For artificial design software to become of commercial use, some form of communication functionality is highly desirable. The extension of artificial design systems based upon object oriented technology to include such database facilities is a straight forward process. However, the ability of the software to obtain relevant design information directly from the internet is, unfortunately, at present non-trivial.

The use of object oriented programming for the construction of artificial design software will have similar development costs compared to the application of other software technologies to fulfil similar objectives. Therefore it is more pertinent to question under what conditions should artificial design software be developed?

Based upon the experience gained through this project, design application domains require the following properties to be considered suitable for the development of artificial design systems:

- The design process for the artefact must be definable, highly structured and organised.
- The attributes upon which design decisions are based, for either the complete artefact or its components, must be quantifiable.

The above two points are discussed in more detail below.

To apply the object oriented programming technique to a design domain, a series of interrelated objects must be created which mirror all the real life aspects of the artefact in question sufficiently to successfully complete the design process. Such objects must be self contained and know how to interact with other objects to achieve the design objectives. In order to achieve this, the design domain must be definable to allow analysis of the domain to be achieved. Furthermore, the domain must be highly structured and organised so that class definitions can be determined and the interrelationships between objects defined. These features are common to the majority of multi-component systems.

During the design process, decisions will need to be made concerning all aspects of the artefact, which implies that all design decisions modelled in the software must be evaluated computionally. Therefore decisions concerning the artefact and all of its constituent components must be determinable within the software design application.

Application domains in which the process of developing artificial design software would be of financial benefit are characterised by:

1. High cost, low volume components.
2. Low cost, high volume components.
3. A design process of significant complexity requiring specialist designer skills.

If a proposed design problem domain possess either of the first two attributes and the third attribute then the domain is suitable for the development of artificial design software.

The object oriented programming paradigm can be utilised to encode any problem domain. However, it is in application domains which are characterised by

patterns of similar repeated behaviour that will successfully yield to the object oriented programming approach, the benefits of which include economy of programming effort, adaptability to future circumstance and maintainability. Such repeated patterns of behaviour are apparent within many multi-component systems.

For example, the general function of a switchboard section is to switch electrical loads, but the type of electrical load[18] determines the additional equipment required to be mounted within a section. It is this pattern of broadly similar devices (in this case sections) that have differing specifications dependant upon external factors (the load that the section supplies) that indicates the suitability of the object oriented programming technique to multi-component design. Such relationships may be easily expressed using the inheritance and polymorphism mechanisms.

Having developed an object oriented architecture within which multi-component design rationale can be captured and performed, it is worth considering if such reasoning could be achieved utilising another software implementation technology. Certainly the design reasoning described in this thesis for embedded generation switchgear design and specification could have been performed utilising an expert system. However, such a system would have been more difficult to construct, maintain and would have proved inflexible should the design rationale changed significantly. No other artificially intelligent technology or programming technique is suitable for modelling multi-component design processes.

The future development of software based artificial design systems will certainly continue as there are already substantial financial incentives for their construction and deployment. The intelligence and complexity of design reasoning capabilities of future software design systems will certainly continue to develop. At present, the main artificially intelligent technology being utilised for the development of design software is intelligent agents. This trend is set to continue as intelligent agents possess attributes, such as autonomy, social ability and proactiveness,[19] that are of great benefit to modelling design reasoning. It is likely that future software applications that are said to exhibit creative design reasoning will be based upon intelligent agent technology. It is also interesting to note that the majority of intelligent agent applications are coded using object oriented programming languages.

7.5 Future Research

The initial restriction of developing a software tool for switchgear design was selected as this is common to all embedded generation schemes. The basic switchgear

[18] For example induction motor or synchronous generator.
[19] Refer to Section 3.2.5.

design processes encapsulated in the software may be extended with reasonable effort due to the inheritant properties of object oriented programming. Possible extensions to the software's functionality are discussed below.

The automated generation of circuit diagrams or schematics of proposed switchgear installations would add a powerful feature to the software. This functionality could be included within the software by associating corresponding circuit symbols to each electrical component modelled in the software as part of their class definition. A similar approach could be utilised to produce a net list to allow proposed switchboards to be modelled within numerical simulation software applications, such as ERACS [79], to assess fault levels and stability issues.

The development and inclusion of a database of switchboard components would dramatically enhance the design capabilities of the software by allowing a choice of component models from which the most appropriate could be selected. Optimisation algorithms could be developed to allow the switchboard design process to be multi-faceted instead of purely functional design based. Such secondary design objectives include cost, reliability or size.

Finally, the existing software could be extended to include modelling of other aspects of embedded generation scheme design. The techniques for modelling design reasoning in this project can be applied to any multi-component system which includes many renewable energy technologies. For example the technology associated with small scale hydro schemes is particularly suitable, as the mature nature of hydro turbine technology means that the design process for the components that constitute a hydro scheme are well defined and understood. Therefore hydro power generation technology is a prime candidate for artificial design development.

7.6 Overall Conclusions

The project software, Edges, has proved to be an effective tool to capture and perform artificial design for embedded generation switchgear installations. The following key features allowed this successful development:

- The design process for switchgear design was able to be rationalised into a structured, organised arrangement of individual objects, relationships and interdependencies.
- All aspects of the electrical design, including regulations that affected this process, were included in the software model.

- The selection of an appropriate implementation technology, in this case object oriented programming, was crucial to the project's success.

Through the development of the project the following issues associated with developing artificial design software for multi-component systems became apparent:

- The detailed exploration of the design domain, combined with practical experience within it, is necessary for the successful development of any artificial design software application.
- The object oriented programming paradigm, with its rich syntax, provides an excellent basis within which to develop artificial design software.
- The flexible, extendable nature of well structured object oriented programming code ensures that the significant effort required to encode a problem domain is not wasted upon maintenance or excessive code editing, as the software evolves with the design processes.

If artificial design systems are to rival the ability of their human counterparts, the following architectures will have to be developed that allow:

- the continued and automated updating of domain knowledge, and,
- design process optimisation strategies that can redefine the design process for a domain in ways not envisaged by the developer.

Until such software applications are developed, artificial design software will simply mimic logical, rational design domains within which domain specific design metaknowledge can be captured through either a collection of specialist algorithms or one or more connectionist networks, or a combination of both.

This thesis demonstrates that switchgear design for embedded generation, or any multi-component artefact, can be captured and assisted within a software tool constructed utilising object oriented programming techniques.

7.7 Thesis Summary

The motivation for this thesis was the exploration and development of an object oriented software application to automate and emulate the design process. To this end, the application developed, Edges, fulfills this objective. In the process, the selection and use of the object oriented programming paradigm has been proven as

a tool capable of the task of capturing artificial design reasoning in a computable form with sufficient flexibility to allow the code developed to be adaptable and manageable. The future of artificial design research will certainly continue to utilise object oriented technology, either as a stand alone programming language or combined with other artificially intelligent technologies.

The future of embedded generation currently appears to be optimistic. It is hoped that the future development of embedded generation, especially low environment impact electricity generation technologies, will continue. The future advances gained from the development of such power generation technologies will hopefully be adopted throughout the world, assisted by artificial design systems, such as Edges. The fundamental aim of these software systems should be to accelerate the appropriate utilisation and deployment of renewable electricity generation technologies. For we owe it to future generations to become dependent upon sustainable electricity production.

Appendix A

Notation Guide

Throughout Chapter 5, illustrations have been extensively utilised to assist the explanation of the construction, organisation and operation of object oriented programming within the problem domain of switchgear design. The three notations used are described in this appendix and are as follows:

1. *Class diagrams* illustrate what classes exist, their structure and the relationships between them.
2. *Object diagrams* indicate the structure of a particular set of objects during execution.
3. *Interaction diagrams* depict the communication between objects over a limited time period.

The notations adopted in this thesis were originally published by Gamma *et al* [92]. The representation diagrams utilised by Gamma *et al* were derived from object oriented notations used throughout the software industry at the beginning of the 1990's, with only minor alterations. Both the class and object diagrams are based upon the object modelling technique (OMT) developed by Rumbaugh *et al* [231, 230, 53]. The notation for the interaction diagrams was founded upon the Booch method [31, 53].

The representation diagrams described in this appendix are not widely utilised throughout the software industry. The current industry standard object oriented design methodology is the Unified Modelling Language (UML).[1] Gamma *et al* no-

[1]By the early to mid 1990's, the software industry became concerned with the profusion and divergence of object oriented design methodologies and notations [112]. In response to this concern, the authors of the three major design methodologies at that time, namely Booch, Rumbaugh and Ivar Jacobson (who developed the OOSE – Object Oriented Software Engineering – method) joined forces in the autumn of 1995 to create a unified design methodology. By 1997, UML was officially released.

tation and the UML notation are both based upon two key oriented design methodologies; independently developed by Booch and Rumbaugh. The use of Gamma *et al* notation in this thesis has been selected due to its pictorial simplicity.[2]

The following sections will describe class, object and interaction diagrams, indicating how the elements within each of these diagrammatic notations represent the various facilities of the object oriented programming paradigm.

A.1 Class Diagrams

Simplified or large class diagrams omit any details concerning the member functions or variables for individual classes, as illustrated in Figure A.1(a). When additional details are required, a list of member functions and variables will be given below the class name, as shown in Figure A.1(b). However, it should be noted that such lists may or may not be a complete representation of the classes in question. Italic type face is used to indicate whether the class or member functions are abstract. The inclusion of type information (*e.g.* an integer, floating point number or a text string) is optional.

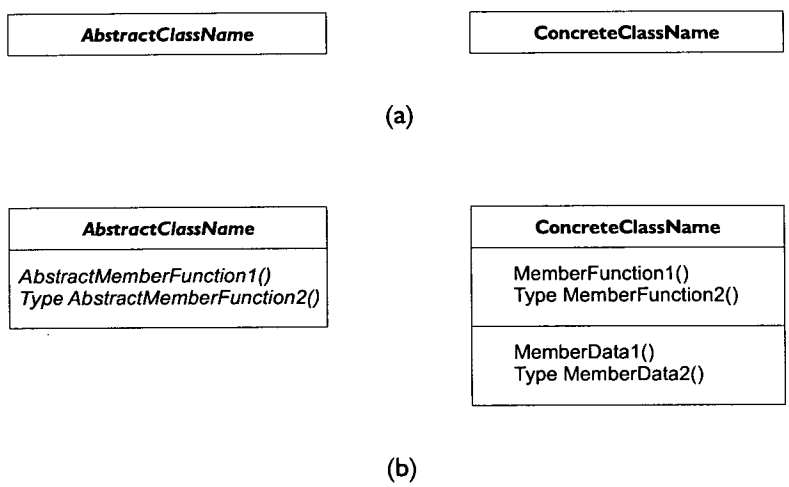


Figure A.1: Class notation diagrams without any class relationships indicated; (a) basic notation to represent abstract and concrete classes, (b) basic notation with member functions and variables indicated.

The various relationships between classes are illustrated in Figure A.2. Class inheritance is indicated by a triangle connecting the subclass to its superclass. A solid line with an arrow head indicates that the class from which the arrow head protrudes from has an object reference to the class pointed to. A diamond shape

[2] All three of Gamma *et al* notations are present within UML except that Interaction diagrams have been renamed *Scenario diagrams*.

at the beginning of such a line denotes aggregation. If the arrow head has a filled circle preceding it, then this relationship may be repeated on one or more occasions to different instances of the class pointed to: otherwise, the arrowhead on its own indicates a one to one relationship. A dashed line pointing to a class indicates that instances of that class are created by the class the dashed line protrudes from. Single or multiple creation relationships are indicated in the same manner as reference and aggregation relationships. Finally, annotations or pseudocode fragments may also be added to class diagrams as indicated by an unfilled circle joined to a paper icon via a dashed line.

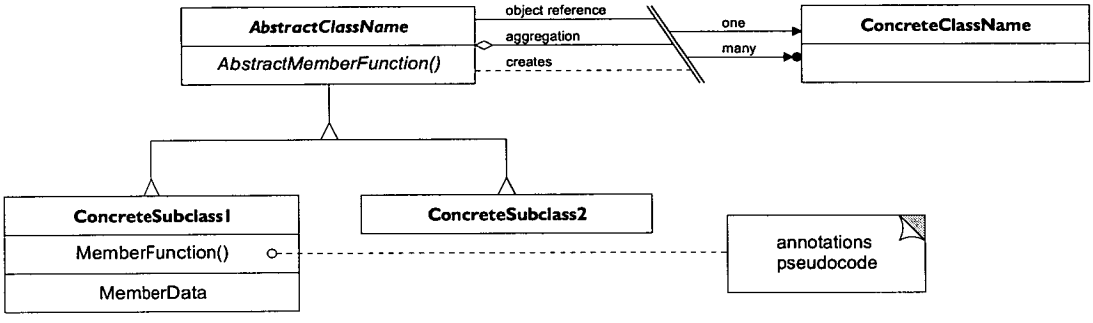


Figure A.2: Class notation diagram with all inter-class relationships depicted [92, (Adapted)].

The class diagram notation described above is the complete class diagram notation used by Gamma *etal.* However, since the project software consists of specially developed classes combined with Microsoft Foundation Classes (MFC), the class diagram notation has been extended to allow both class types to be differentiated from one another. MFC's are indicated by having a shaded background, as illustrated in Figure A.3.

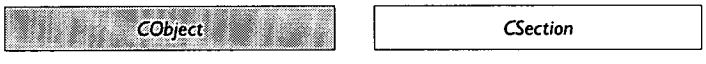


Figure A.3: Class notation diagram with both MFC and design classes depicted.

A.2 Object Diagrams

An object diagram depicts the arrangement of objects at a particular juncture during the execution of an object oriented software system. Figure A.4 illustrates that a rounded box is used to depict an object with references between objects indicated by a filled circle and connecting dashed line with an arrow head. A line separates

the object's name from any references to other objects, member functions or member variables.

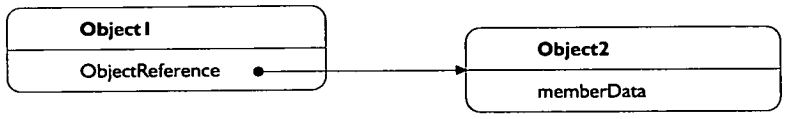


Figure A.4: Object diagram notation [92, (Adapted)].

A.3 Interaction Diagrams

An interaction diagram allows the order of communication requests between objects to be depicted graphically, as shown in Figure A.5. A solid vertical line indicates the existence of an object at a particular point in time; within interaction diagrams time flows from the top of the diagram downwards. Conversely, a dashed line indicates that the object has yet to be created. Upon receiving a request, an object will obtain execution control and this is represented by a vertical rectangle.

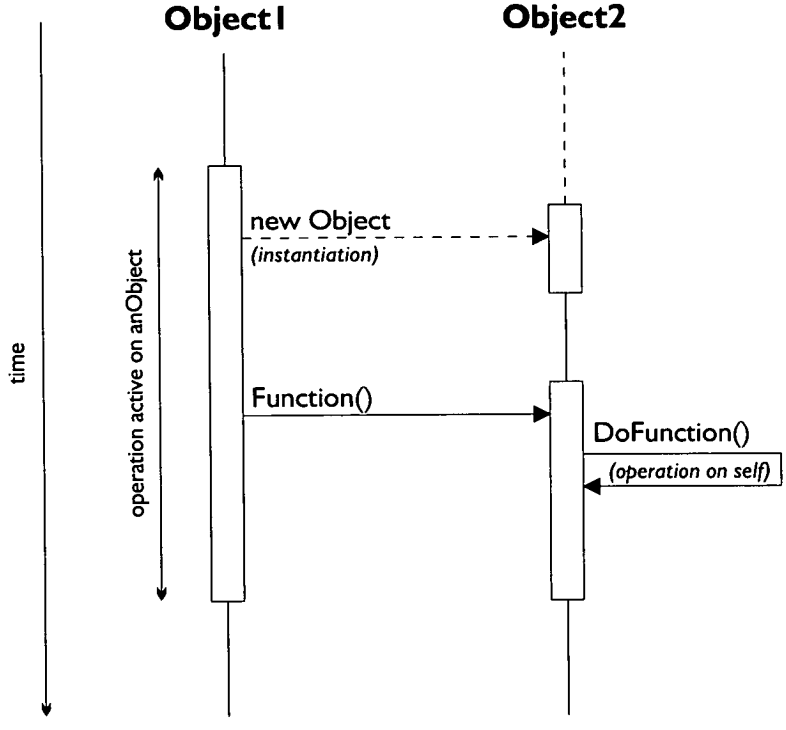


Figure A.5: Interaction diagram notation [92, (Adapted)].

Two types of horizontal lines appear in interaction diagrams; a dashed line indicates that a new object has been created (otherwise known as *instantiation*) and a

solid line indicates that a member function of the class pointed to has been called from the object the line protrudes from. Member function calls that are called and executed from within an object are indicated by a solid line and arrow pointing back to the object itself. The names of member function calls are indicated on their respective horizontal line.

Appendix B

Generator Specification

Table 6.3 in Chapter 6 provides a summary of the generator components automatically specified by Edges. The complete list of components, the number of each device required with their respective attribute ratings are given in Table B.1 on pages 187 to 194.

Component	*Quantity*	*Rating*
Miscellaneous		
Busbars	x3	
manufacturer		Yorkshire Switchgear
current rating		1200A
frequency rating		50Hz
Circuit breaker	x1	
manufacturer		Yorkshire Switchgear
part number		to be specified
interrupting medium		SF_6
current rating		1200A
voltage rating		3.3kV
frequency rating		50Hz
applicable standards		BS5311
spring charging mechanism	x1	
type		electric motor
power rating		190W
voltage rating		230Vac
spring release coil	x1	
power rating		205W
voltage rating		110Vdc
shunt trip coil	x1	
power rating		115W
voltage rating		110Vdc
Anti-condensation heater	x1	
power rating		40W
voltage rating		230Vac
manufacturer		to be specified
part number		to be specified
fuse protection	x1	
voltage rating		230Vac
current rating		1A
mounting type		cartridge
manufacturer		to be specified
part number		to be specified
applicable standards		BS88
Section label	x1	
label text		Power Station
text height		20mm
text colour		black
material		aluminium
Control Switches:		
Local/Remote	x1	
contact voltage rating		230Vac
contact current rating		10A
contact number		2
contact operation		latching - break before make
manufacturer		to be specified
range / series		to be specified
part number		to be specified
type		selector
applicable standards		IP65; CE mark
component label	x1	

Table B.1: A complete list of constituent components automatically defined by Edges for the synchronous generator section in the paper mill switchboard.

Component	*Quantity*	*Rating*
label text		Local / Remote
text height		10mm
text colour		white
material		plastic
Trip/Neutral/Close	x1	
contact voltage rating		230Vac
contact current rating		10A
contact number		2
contact operation		latching - break before make
manufacturer		to be specified
range / series		to be specified
part number		to be specified
type		selector
applicable standards		IP65; CE mark
component label	x1	
label text		Trip / Neutral / Close
text height		10mm
text colour		white
material		plastic
Amp meter phase selection switch	x1	
contact voltage rating		230Vac
contact current rating		1A
contact number		2
contact operation		latching - make before break
manufacturer		to be specified
range / series		to be specified
part number		to be specified
type		selector
applicable standards		IP65; CE mark
component label	x1	
label text		R / Y / B Off
text height		10mm
text colour		white
material		plastic
Volt meter phase selection switch	x1	
contact voltage rating		230Vac
contact current rating		1A
contact number		2
contact operation		latching - break before make
manufacturer		to be specified
range / series		to be specified
part number		to be specified
type		selector
applicable standards		IP65; CE mark
component label	x1	
label text		R / Y / B Off
text height		10mm
text colour		white
material		plastic
Indicator Lamps:		
Busbar live lamps	x3	
power rating		5W

Table B.1: *continued.*

Component	*Quantity*	*Rating*
voltage rating		110Vac
frequency rating		50Hz
manufacturer		to be specified
part number		to be specified
lamp filter colour		red
component label	x1	
label text		Busbars Live
text height		10mm
text colour		white
material		plastic
fuse protection	x3	
voltage rating		110Vac
current rating		1A
mounting type		cartridge
manufacturer		to be specified
part number		to be specified
applicable standards		BS88
power source		connection to VT1
Instrumentation:		
Amp meter	x1	
manufacturer		Alsthom T&D
series / range		Metrik
part number		to be specified
applicable standards		BS89 and IP52
size		96mm^2
scale		90°
accuracy		1%
end scale value		1400A
scale label		A
current transformer properties		connection to CT1
burden (per phase)		2VA
voltage transformer properties		not required
fuse protection		not required
component label	x1	
label text		Ampere Meter
text height		10mm
text colour		white
material		plastic
Volt meter	x1	
manufacturer		Alsthom T&D
series / range		Metrik
part number		to be specified
applicable standards		BS89 and IP52
size		96mm^2
scale		90°
accuracy		1%
end scale value		3.6kV
scale label		A
current transformer properties		not required
voltage transformer properties		connection to VT1
burden (per phase)		2VA
fuse protection	x3	

Table B.1: *continued.*

Component	*Quantity*	*Rating*
voltage rating		110Vac
current rating		1A
mounting type		cartridge
manufacturer		to be specified
part number		to be specified
applicable standards		BS88
component label	x1	
label text		Voltage Meter
text height		10mm
text colour		white
material		plastic
Protection Relays:		
Over and under current relay	x1	
manufacturer		Alsthom T&D
series / range		Midos
part number		to be specified
applicable standards		BS142 & IEC255
direct current burden		5W (110Vdc)
current transformer properties		connection to CT2
burden (per phase)		0.25VA
voltage transformer properties		not required
fuse protection		not required
component label	x1	
label text		Over and under current
text height		10mm
text colour		white
material		plastic
Over and under voltage relay		
manufacturer		Alsthom T&D
series / range		Midos
part number		to be specified
applicable standards		BS142 & IEC255
direct current burden		5W (110Vdc)
current transformer properties		not required
voltage transformer properties		connection to VT2
burden (per phase)		0.05VA
fuse protection	x3	
voltage rating		110Vac
current rating		1A
mounting type		cartridge
manufacturer		to be specified
part number		to be specified
applicable standards		BS88
component label	x1	
label text		Over and under voltage
text height		10mm
text colour		white
material		plastic
Under frequency relay		
manufacturer		Alsthom T&D
series / range		Midos
part number		to be specified

Table B.1: *continued.*

Component	*Quantity*	*Rating*
applicable standards		BS142 & IEC255
direct current burden		5W (110Vdc)
current transformer properties		not required
voltage transformer properties		connection to VT2
burden (per phase)		0.05VA
fuse protection	x3	
voltage rating		110Vac
current rating		1A
mounting type		cartridge
manufacturer		to be specified
part number		to be specified
applicable standards		BS88
component label	x1	
label text		Under frequency
text height		10mm
text colour		white
material		plastic
Over frequency relay		
manufacturer		Alsthom T&D
series / range		Midos
part number		to be specified
applicable standards		BS142 & IEC255
direct current burden		5W (110Vdc)
current transformer properties		not required
voltage transformer properties		connection to VT2
burden (per phase)		0.05VA
fuse protection	x3	
voltage rating		110Vac
current rating		1A
mounting type		cartridge
manufacturer		to be specified
part number		to be specified
applicable standards		BS88
component label	x1	
label text		Over frequency
text height		10mm
text colour		white
material		plastic
Loss of mains relay (ROCOF)	x1	
manufacturer		Alsthom T&D
series / range		Midos
part number		to be specified
applicable standards		BS142 & IEC255
direct current burden		5W (110Vdc)
current transformer properties		connection to CT2
burden (per phase)		0.25VA
voltage transformer properties		connection to VT2
burden (per phase)		0.05VA
fuse protection	x3	
voltage rating		110Vac
current rating		1A
mounting type		cartridge

Table B.1: *continued.*

Component	*Quantity*	*Rating*
manufacturer		to be specified
part number		to be specified
applicable standards		BS88
component label	x1	
label text		ROCOF
text height		10mm
text colour		white
material		plastic
Earth fault relay		
manufacturer		Alsthom T&D
series / range		Midos
part number		to be specified
applicable standards		BS142 & IEC255
direct current burden		5W (110Vdc)
current transformer properties		not required
voltage transformer properties		connection to VT2
burden (per phase)		0.05VA
fuse protection	x3	
voltage rating		110Vac
current rating		1A
mounting type		cartridge
manufacturer		to be specified
part number		to be specified
applicable standards		BS88
component label	x1	
label text		Earth fault
text height		10mm
text colour		white
material		plastic
Reverse power relay	x1	
manufacturer		Alsthom T&D
series / range		Midos
part number		to be specified
applicable standards		BS142 & IEC255
direct current burden		5W (110Vdc)
current transformer properties		connection to CT3
burden (per phase)		0.25VA
voltage transformer properties		connection to VT2
burden (per phase)		0.05VA
fuse protection	x3	
voltage rating		110Vac
current rating		1A
mounting type		cartridge
manufacturer		to be specified
part number		to be specified
applicable standards		BS88
component label	x1	
label text		Reverse power
text height		10mm
text colour		white
material		plastic
Synchronising relay		

Table B.1: *continued.*

Component	*Quantity*	*Rating*
manufacturer		Alsthom T&D
series / range		Midos
part number		to be specified
applicable standards		BS142 & IEC255
direct current burden		5W (110Vdc)
current transformer properties		not required
voltage transformer properties		connection to VT2
burden (per phase)		0.05VA
fuse protection	x3	
voltage rating		110Vac
current rating		1A
mounting type		cartridge
manufacturer		to be specified
part number		to be specified
applicable standards		BS88
component label	x1	
label text		Synchronising
text height		10mm
text colour		white
material		plastic
Sensing Transformers:		
Current Transformer 1 (CT1)	x1	
type		instrumentation
classification		1
burden (per phase)		2VA
capacity		2VA
number of phases		3
primary rating		1200A
secondary rating		5A
manufacturer		Yorkshire Switchgear
part number		to be specified
fuse protection		not required
Voltage Transformer 1 (VT1)	x1	
type		instrumentation
classification		1
burden (per phase)		7VA
capacity		10VA
number of phases		3
primary rating		3.3kV
secondary rating		110V
manufacturer		Yorkshire Switchgear
part number		to be specified
fuse protection	x3	
voltage rating		3.3kV
current rating		1A
mounting type		cartridge
manufacturer		Yorkshire Switchgear
part number		to be specified
applicable standards		BS88
Current Transformer 2 (CT2)	x1	
type		protection
classification		5P

Table B.1: *continued.*

Component	*Quantity*	*Rating*
burden (per phase)		0.5VA
capacity		2.5VA
number of phases		3
primary rating		1200A
secondary rating		5A
manufacturer		Yorkshire Switchgear
part number		to be specified
fuse protection		not required
Voltage Transformer 2 (VT2)	x1	
type		protection
classification		3P
burden (per phase)		0.35VA
capacity		10VA
number of phases		3
primary rating		3.3kV
secondary rating		110Vac
manufacturer		Yorkshire Switchgear
part number		to be specified
fuse protection	x3	
voltage rating		3.3kV
current rating		10A
mounting type		cartridge
manufacturer		Yorkshire Switchgear
part number		to be specified
applicable standards		BS88
Current Transformer 3 (CT3)	x1	
type		protection
classification		5P
burden (per phase)		0.25VA
capacity		2.5VA
number of phases		1
primary rating		1200A
secondary rating		5A
manufacturer		Yorkshire Switchgear
part number		to be specified
fuse protection		not required

Table B.1: *continued.*

Appendix C

Publications

The work described in this thesis has been reported in the following publications :

[C1] Paul McCabe and D.E. Macpherson. *The Application of Artificially Intelligent Techniques to the Design Process of Renewable energy Schemes*, 32nd Universities Power Engineering Conference (UPEC '97), Volume 2, pages 1126 – 1129, 10th – 12th September, 1997. Department of Electrical Engineering and Electronics, University of Manchester Institute of Science and Technology (UMIST), PO Box 88, Manchester, United Kingdom. Volume 1: ISBN 0-9523165-3-6, Volume 2: ISBN 0-9523165-4-4, Two Volume Set: 0-9523165-5-2.

[C2] Paul McCabe. *Application of Artificially Intelligent Techniques to the Design Process of Embedded Generation Schemes*, The Postgraduate Journal of the Department of Electronics and Electrical Engineering (PhDEE), pages 41 – 45, April 1998, Department of Electronics and Electrical Engineering, University of Edinburgh, Mayfield Road, Edinburgh, United Kingdom.[1]

[C3] Paul McCabe, D.E. Macpherson and A.R. Wallace. *An Artificially Intelligent Design Aid for the Specification of Protection Relays and Switchgear for Small Scale Hydro Plant*, Small Hydro 1998, pages 105 – 112, 16th – 18th November, 1998. Athens, Greece. International Water Power & Dam Construction and Wilmington Business Publishing, Wilmington House, Church Hill, Wilmington, Dartford, Kent, United Kingdom.

[1] This paper has not been reproduced in this volume due to its very similar structure and content to [C1].

C.1 UPEC '97

APPLICATION OF ARTIFICIALLY INTELLIGENT TECHNIQUES TO THE DESIGN PROCESS OF RENEWABLE ENERGY SCHEMES.

P.R. McCabe and D.E. Macpherson

University of Edinburgh, UK

ABSTRACT

The number of small-scale renewable energy schemes being utilised within the United Kingdom continues to grow, encouraged by Government legislation and public concern for the environment. The limited capacity of these schemes often results in poorly resourced projects which fail to attract the appropriate technical expertise at the design stage. Hence, once constructed the plant performs inadequately, becomes difficult to maintain, expensive to operate and has a significantly reduced life expectancy. These problems can be reduced by implementing artificially intelligent techniques during the design stage. This paper will outline the main features of the design process, discuss artificial intelligence and illustrate how this technology may be applied to the design process of renewable energy schemes.

1. INTRODUCTION

Recent Government legislation, in particular the Third order of the Non-Fossil Fuels Obligation (NFFO-3) and the Scottish Renewables Obligation (SRO), coupled with increasing environmental concerns and an ever increasing demand for electricity, has increased public interest in renewable sources of energy (e.g. wind, hydro, solar, energy crops, landfill gas) for electricity generation. This has been reflected in a substantial increase in the number of embedded generators[†] that have been connected to the distribution network. New embedded projects are becoming decreasingly small, down to a capacity of ten's of kilowatts. Due to their decreasing small size, many of these projects have insufficient capital resources to purchase professional expertise and are therefore proposed and implemented by people of inadequate technical experience. As a result many Public Electricity Suppliers (PES) are having to process large numbers of applications for new embedded generation schemes which are inadequately specified and, in some cases, fall a long way short of the required electrical regulations such as G59/1.

Past experience has indicated that the major costs associated with the development of embedded generation sites have been :

1. The amount of professional time required to electrically design and specify the plant.
2. The high cost of undetected errors made during the design of the plant.
3. Poor reliability, efficiency and maintainability of the plant.

Thus the design process is a crucial factor in the economic viability and ultimate performance of the proposed embedded scheme.

It is also apparent that the (automated) control and electrical protection systems for many renewable energy schemes are similar, both in electrical and functional terms. Therefore, it is proposed that to avoid inadequately designed and costed proposals a design aid could be utilised which would :

- ensure that the proposed embedded scheme was economically and technically viable,
- minimise the number of poorly prepared applications processed by the utilities,
- increase the overall performance of the scheme,
- produce a project costing.

For such an design aid to be effective it should embody basic computer aided design (CAD) functions and at least some of the deeper reasoning which is applied by human designers to the design process.

[†] Embedded generators are generally accepted as having ratings less than 10MW and connected at 33kV or below.

2. REVIEW OF THE DESIGN PROCESS

The fundamental process of the design of an artefact (a man-made object) is a distinctly human activity, and because of its' abstract nature many authors have diverse views upon the precise definition of design [1,2,3]. Due to the complex nature of design and for clarity it will be described, as opposed to defined, as *the creation or adaptation of an artefact, or group of artefacts, such that certain characteristics which have been predetermined become optimised features of the desired artefact.*

Before attempting to encode the design process artificially, some consideration should be given to the design methods used by human practitioners, as a successful system should incorporate at least some of these methods.

2.1 Methods of Design

When presented with a design task designers call upon various resources to focus the search for a suitable solution. These resources avoid the designer being overwhelmed by the number of different possibilities and minimise time wasted exploring unsuitable solutions. Such resources include the senses (*e.g.* sight, hearing and touch), previous design experience and knowledge relevant to the task, imagination, creativity and the ability to work with incomplete or defective information. All these resources are combined and utilised by the designer to

produce an optimum solution to the design task.

Designers employ distinct methods to divide the task into manageable sections. The principal methods are outlined below.

2.1.1 Top-Down Design
This is one of the most utilised human design practices due to its simplicity and effectiveness. The structure is decomposed into several main sections and ideally the interfaces between each section are completely defined. Thus the design of each section becomes a totally independent sub-problem. This process is repeated until all the sub-sections can be resolved individually. Re-composition of the sub-sections results in the solution of the original design task.

The main limitation with this method is that the interfaces between the sub-sections are invariably incomplete or permutate during the design process. The re-definition of an interface may not only affect adjoining sub-sections but the entire structure of the design.

2.1.2 Iterative Design
Also known as the 'Generate-and-Test' method, iterative design is a powerful but resource intensive process. A proposed solution is generated using quantities which satisfy some of the design constraints while the remaining variables assume approximate values. The prototype is then tested against the design criteria. If the proposed design fails the process is re-iterated utilising information gained from the testing procedure to modify the design parameters.

Although this approach allows for 'rules of thumb' to be applied (as long as they can be formalised) the system fails to take account of the overall purpose of the artefact as it is only interested in ensuring that the performance criteria have been met.

2.1.3 Parameteric Design
This procedure is now a widely used technique due to the advent of modern computer technology. The parameters and inter-relationships of the desired artefact are captured through a mathematical model of the design process. An artefact is generated by assigning values to the parameters and executing the model. This system is more akin to mathematical modelling than artificial design, but since the system can appear to produce new artefacts it is generally considered as having design capability.

It should be noted that this is not an exhaustive list, but the majority of design tasks may be represented by one or more of the above techniques. Each of the above methods has advantages, *e.g.* computational efficiency, and disadvantages, *e.g.* failure to capture the richness of the design task, therefore most designers use a combination of these methods and their personal resources to focus the design of a single artefact.

2.2 Computer Assisted Design
If a computer is to be effective in mimicking a human designer all of the resources available to the human designer have to be simulated, re-created or substituted artificially. Most computer-based design packages have only an alphanumerical database containing the key facts of the objects which they manipulate, in accordance with a set of design rules. This is only a fraction of the design information that human designers have access to, *e.g.* visual and physical stimuli related to the task. It is not surprising, therefore, that computer-based design software performs poorly without human intervention.

3. CURRENT DESIGN SOFTWARE

There are two distinct paradigms concerning artificial design :

1. ***Design as a routine process.***
   Including : Problem solving processes.
   Decision making processes.
   Optimisation processes.
2. ***Design as an inventive, innovative and creative process.***

It should be noted that a degree of overlap exists between the two areas. 'Design as a routine process' treats design as a straight-forward, repetitive, logical process requiring little creativity or thought. This approach has embraced Expert System based technology and applied it to areas where repetitive design decisions have to be made frequently and accurately. Application areas include VLSI layout, electrical circuit design, intelligent CAD systems and control system design. Within all of these areas the design environment is well defined and completely described by mathematical expressions; quantities which computers can process readily. Systems developed using this methodology have been very successful but all require human assistance to operate.

'Design as an inventive, innovative and creative process' utilises the innovative ideas of artificial intelligence, *e.g.* neural networks and genetic algorithms, to create novel approaches to represent and manipulate knowledge concerning the design. These systems require extensive development and computational resources, and have had varying degrees of success when applied to real-life applications. Such tasks include power system generation scheduling, pattern recognition and fault diagnosis of complex control systems.

Before considering how a design aid for renewable energy schemes is being implemented, a brief overview of current artificially intelligent techniques which are currently in use within power systems is presented.

4. ARTIFICIALLY INTELLIGENT TECHNOLOGIES

Since the mid' eighties there has been a divergence away from the investigation of power systems by traditional methods (*e.g.* rigorous mathematical modelling) towards the application of artificially intelligent based technology [4]. Presently the principle artificially intelligent technologies utilised within power system applications are expert systems, fuzzy systems (based upon the principle of fuzzy logic), artificial neural networks and genetic algorithms. Each one of these technologies will be

discussed with respect to their application to artificial design.

The most established of the artificially intelligent technologies, with the longest history of proven success, are expert systems, which allow specialist knowledge within a narrow domain to be captured in a machine interpretable form. This knowledge may be captured in the form of rules, frames, objects or a combination of these structures. The initial stages of the design of any artefact are generally well structured. Therefore it seems intuitive to use an expert system to help structure and define the initial design stages for an artefact before utilising other programming techniques for the more complex, ill-defined sections of the design task.

Fuzzy systems are based upon fuzzy logic where decisions can be made where for a given set of circumstances a degree of certainty may be found such that a certain condition exists. Thus fuzzy systems are able to deal with partial degrees of truth or vagueness. Fuzzy logic could be utilised within certain design tasks where all the possible conditions or relations could be defined and weighted within a limited domain, but by itself fuzzy logic is too limited and un-structured to deal with the richness required for artificial design.

Genetic algorithms can be generally defined as a computer simulation in which a population of abstract representations of candidate solutions to an optimisation problem are stochastically selected, recombined, mutated and then either eliminated or retained based upon their relative fitness [5]. Genetic algorithms are advanced random search methods or optimisers which require a set of governing equations. Since most design tasks are not easily converted into a set of mathematical equations, genetic algorithms appear to be of limited use for artificial design.

Artificial neural networks may be defined as mutli-layer networks constructed from layers of neurons which are extracted models of the biological neuron; they operate by taking the weighted sum of their inputs, and should this sum exceed a predetermined threshold the neurone fires, outputting the equivalent to a logic one or high. The tasks that artificial neural networks excel at are classification, recognition, optimisation and prediction. However, the main limitation with artificial neural networks is that they require extensive archives of standardised training data; thus implying that not only must most of the possible design options for a given task be calculated but that they also be classified and standardised into training data. Therefore the artificial neural network is operating as a classifier or optimiser as opposed to an intelligent design aid.

4.1 Object Oriented Design

One of the currently developing software paradigms, object-orientation, has emerged in recent years as a key technology of fundamental importance to artificial design. This non-artificially intelligent technology is founded upon the principle that objects identified in the problem specification can be represented in software; allowing the construction of an object-orientated model of the problem and mapping this model into a software design [6]. Several authors [7,8] have already utilised this technique with encouraging results.

In summary; the rigorous and precise structure of expert systems coupled with the flexibility of object-orientated software allows a fully featured, rich environment within which to develop a design aid. The following section describes the operation of the design aid for the electrical specification of embedded generation.

5. A DESIGN AID FOR THE ELECTRICAL SPECIFICATION OF EMBEDDED GENERATION

Work is currently progressing on an artificial design aid for the electrical specification of embedded generation schemes. The design aid is based upon an object-orientated software language, C++, with the inclusion of an expert system shell, CLIPS, which is used for the initial stages of the design and the validation of re-structuring or changes in specification. The development environment used is Microsoft® Windows NT operating system, but depending upon the computational demands of the software the code can be ported onto a UNIX platform if necessary.

The structure of the system is as follows; many of the proposed sites for embedded generation development may be represented by a small number of generic circuits (examples are shown in Figures 1 and 2). Within each of these generic circuits there are functional blocks, *e.g.* generator, transformer, protection systems, which are common to all embedded generation schemes. Each of these circuit blocks have been given characterising attributes, some general (*e.g.* cost, power rating) and some specific (*e.g.* electrical connections, control signals).

Initially the user is presented with a graphical user interface displaying the various generic circuit options. The user may select the circuit which best approximates his/her requirements and upon selection the user is then presented with a more detailed view of the circuit topology, indicating the various components or functional blocks. Each of these functional blocks may be selected allowing the user to enter the proposed specifications.

As the user enters the specification for a functional block into the design aid, she/he will have a choice of default values or components to select from. Once the specification for each component has been input, exit from the respective sub-menu will only be granted when sufficient information has been given to completely specify it's operation. This structure will ensure that all components or blocks are specified thus avoiding the absence or non-specification of components; thus avoiding a major cost associated with current embedded design practices.

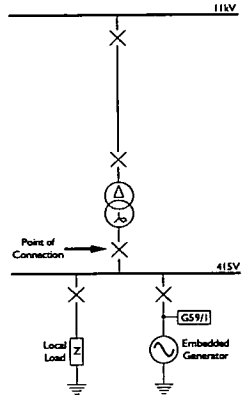


Figure 1 **: Example of a Generic Circuit; a long line with a distribution transformer and an embedded generator**

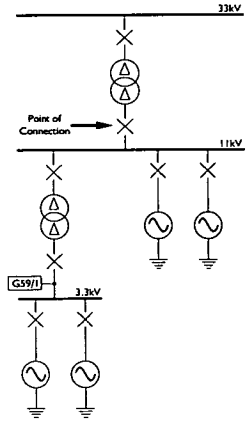


Figure 2 **: Example of a Generic Circuit; a Wind Farm with 3.3kV and 11kV generation.**

Either during or after specifying the proposed design the user can change the existing circuitry (derived from the generic circuit) or include additional circuitry. This may be accommodated by the adaptation of the attributes of adjoining circuit elements thus ensuring that the design aid is able to adjust for changing or unfamiliar circuit topologies.

Having completely entered the proposed design, including the circuit topology and specification for all the constitute components then a complete list of objects, within the software, and their respective attributes is created. This allows the overall design focus to be modified and applied to all the objects within the design specification. Therefore the final specification may be optimised for various options, *e.g.* cost, reduced maintaince or remote operation. To accommodate the users design knowledge, each of the changes suggested by the software to the respective objects or their characteristics is displayed and has to be vetted by the user. Hence the description of the software as a design aid opposed to a expert system for the design of renewable energy schemes.

6. CONCLUDING REMARKS

This paper has presented the main problems associated with embedded generation due to poor initial design. A brief analysis of the design process and methods of application has been indicated. A synopsis of current design software and artificially intelligent technologies has been presented along with a summary regarding their application to artificial design. Finally a design aid, which is currently under construction, for the electrical specification of embedded generation schemes has been described.

REFERENCES

1. Simon, H.A. *The sciences of the Artificial*, MIT Press, pp55-56, 1968.
2. Godart, T.F., Puttgen, H.B. and Jansen, J.F. *Expert system applications to power engineering control design.* Proceedings of the 5th IEEE International Symposium on Intelligent Control. IEEE Comput. Soc., Press 2 vol. xxii, pp. 1145-50, 1990.
3. Grierson, D.E. and Hajela, P. (Ed's). *Emergent Computing Methods in Engineering Design.* NATO ASI Series, Vol. 149, pp. 150-161, 1996.
4. Warwick, K., Ekwue, A. and Aggarwal, R. (Ed's). *Artificial Intelligence Techniques in Power Systems.* IEE Power Engineering Series 22, pp1-18, 1997.
5. Hondroudakis, A, Malard, J. and Wilson G.V. *An Introduction to Genetic Algorithms Using RPL2.* University of Edinburgh, p4, 1996.
6. Daniels, J. Object Methods and Tools. IEE Colloquium on 'Distributed Object Management'. London, UK, p3, 1994.
7. Booch, G. Object oriented design : with applications. Benjamin/Cummings Press, 1994.
8. Shlaer, S. Object lifecycles : modelling the world in states. Yourdon Press, 1992.

AUTHOR'S ADDRESS

The first author can be contacted at :

Department of Electrical Engineering
University of Edinburgh
King's Buildings
Mayfield Road
Edinburgh
EH9 3JL
Scotland, UK

C.2 Small Hydro '98

AN ARTIFICIALLY INTELLIGENT DESIGN AID FOR THE SPECIFICATION OF PROTECTION RELAYS AND SWITCHGEAR FOR SMALL SCALE HYDRO PLANT

Mr. P R McCabe, Dr. D E Macpherson and Dr. A R Wallace

Department of Electronics and Electrical Engineering
The University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh, United Kingdom EH9 3JL
Tel : +44 (0)131 650 5584 *Fax : +44 (0)131 650 6554* *Email : prm@ee.ed.ac.uk*

Abstract

Mini-hydro generators have been installed for many years as the power sources in decentralised rural electrification schemes in less developed countries. Within the liberalised Electricity Supply Industries in Europe, mini-hydro generators are being rehabilitated or installed to connect to the distribution grids. At all levels of technical sophistication, safe and reliable operation is of primary importance; these criteria must be addressed from concept to the commissioning of the plant.

Various computationally intelligent software packages have being developed and applied to the top-down design of mini-hydro schemes, but few attempt to specify in detail the switchgear, control and protection equipment, which are the key to the safe and reliable operation of the plant. There are many equipment manufacturers entering the market with the facilities and skills to produce cost effective switchgear, control and protection equipment. However, they may not have the experience in hydro-electric plant to ensure that operational, safety and reliability standards are always maintained. Errors or omissions in the design and specification of plant can at best be costly to repair or re-instate. At worst there can be profound implications for safety and reliability.

This paper describes the design procedures most suited to the specification of switchgear, control and protection systems, and identifies that object oriented programming can be utilised to embody the often subconscious design decisions that ensure plant is precisely and appropriately specified without omissions. The paper describes the application of C++ to the design of switchgear and protection equipment and shows examples of the breadth and depth of detail that must be indicated in a competent and secure design.

1. Introduction

Small scale hydro generators have a long history of reliable operation in decentralised rural electrification schemes throughout the world, especially in developing countries. This technology is currently expanding within the liberalised Electricity Supply Industries (ESI) in Europe due to the following factors -

- *Technical Advancements :* The development of and increased availability of control systems which allow the automatic, remote control of plant.
- *Increasing demand for electricity :* Demand continues to grow and has allowed for the exploration of natural resources previously thought of as non-viable.
- *Government legislation :* Government programs, such as the Thermie program in Europe or the Non-Fossil Fuels Obligations (NFFO 1 - 4) and the Scottish Renewables Orders (SRO 1 - 3) within the United Kingdom, have been strong incentives for developers.
- *Environmental Concerns :* The increasing public interest in the environmental effects of electricity generation has encouraged the growth of hydro power.

Within Europe, the number of mini and small scale hydro schemes is set to grow, with an estimated 800 TWh/year of economically feasible hydro power potential still undeveloped [1], of which an increasing amount will be embedded[†] as most large hydro resources become exploited. This anticipated growth is reflected throughout the world, particularly in the Pacific Rim, Russia, Middle East and South Asia, where rural electrification schemes based on hydro-electric generation have tremendous potential, with current estimates indicating 3600 TWh/year of economically viable hydro generation [2].

A common problem, even within developed countries, is that the maximum potential of the resource is often not realised due to poor performance of the plant. Upon close inspection, the main reason for many schemes poor performance may be traced back to the ad hoc design of the plant. This is often due to the developers of such small schemes :

[†] Embedded generators are generally accepted as having ratings less than 10MW and connected at 33kV or below.

- having very limited capital, thus being unable to seek specialist professional advice, and,
- relying upon a limited skill base which may not be up-to-date or sufficiently broad enough to ensure that the design of the hydro scheme is effective.

Plant developed in these circumstances is prone to mispecification, which is normally only discovered at the site during construction or commissioning, thus incurring expensive alterations or additional equipment. This condition is particularly acute in developing countries, where the potential for mini and small scale hydro is greatest but where many schemes are developed by people whose experience may be limited, especially concerning the electrical protection and control of the plant.

The majority of small hydro plant operates as a small generating station embedded in a larger electrical system. It is essential to the reliability and safe operation of the overall system that the hydro plant complies with relevant national / international standards regarding electrical protection and operation. Within the UK the legal guidelines are defined by the Electricity Association in documents G59/1 [3] and ETR113 [4]; however, designers of small hydro systems are frequently not electrical engineers, and may not be conversant with these standards and accepted design practices. Even with such documents the regulations can easily be misinterpreted by the unqualified. In many cases, particularly in developing countries, funds will not stretch to employing a specialist electrical power systems engineer in addition to the designer(s) of the civil works and electro-mechanical equipment. Resulting design problems can cause lengthy and expensive commissioning delays, and in some cases serious failure of the electrical system.

It is the aim of this project to assist designers of small hydro-electric plant by encapsulating in software the knowledge of experienced electrical engineers concerning the design of electrical control and protection equipment. The knowledge is converted into a set of design rules and combined with databases of available, standardised equipment, which may be supplemented to adapt to local needs. The software aims to ensure the compliance of proposed schemes with the relevant standards and regulations, thereby ensuring the successful licensing of new plant. It is expected that utilisation of this design aid will substantially improve the quality of design, and assist developers avoid expensive, unforeseen changes during commissioning. This knowledge will be accessible through the use of a personal computer.

Implementation of the software requires capturing some of the reasoning and thought processes of experienced human engineers as well as including the basic functionality common to all computer aided design (CAD) tools.

2. The Concept of Design

Defining the term design is difficult due to its general meaning and its abstract nature. Understanding the process of design is a central goal of engineers, scientists, and architects. Many authors have diverse views and definitions of design [5,6], but it is H.A. Simon who describes the spirit of the process most succinctly:

> *"The natural sciences are concerning with how things are." ... "Design, on the other hand , is concerned with how things ought to be, with devising artefacts [man made objects] to attain goals.[7]"*

The process of design can be described as the creation or adaptation of an artefact, or group of artefacts, such that certain characteristics which have been predetermined become optimised features of the desired artefact.

To create a design aid in the form of a software application, some consideration should be given to the design methods employed by human practitioners. Such methods include top-down design, iterative design, parametric design and probabilistic design. This is not an exhaustive list of techniques, but other design methods can generally be represented by one or a combination of these techniques.

Top-down design is the most utilised design technique due to its simplicity and effectiveness. The design task is decomposed into main sections, and the interfaces between each section defined. Therefore the design of

each section becomes a totally independent sub-problem. This process of divide and conquer is applied until all of the sub-sections can be resolved individually. Re-composition of all the sub-sections results in the solution of the original design task.

Each of the aforementioned design methods have advantages (*e.g.* computational efficiency or particular suitability to certain tasks) and disadvantages (*e.g.* a failure to model all the richness of the design task or an over-emphasis upon structure as opposed to creativity); therefore most human designers use a combination of these methods together with their personal skills and resources to focus upon the design of a artefact. It is the aim of the software to model the design process for electrical switchgear and protection equipment for small scale hydro schemes, allowing the user to be concerned only with the general parameters of the proposed project. The regulatory and intricate technical details, where many human designers make mistakes, are automated and checked by the software.

3. Artificially Intelligent Technology

The study of human cognitive processes is an area of research known as Artificial Intelligence (AI) which includes technologies such as expert systems, artificial neural networks, fuzzy logic, agent theory and genetic algorithms. Since the mid 'eighties there has been a divergence away from investigating power systems by traditional methods (*e.g.* rigorous mathematical modelling) towards the application of artificially intelligent technology [8]. Applications include the utilisation of neural networks for dispatching of electric power [9], genetic algorithms for service restoration in electric power distribution systems [10], and expert systems for distribution grid alarm processing [11].

Included within the area of artificial intelligence is object oriented design. As its name suggests, this area of research attempts to encapsulate human design decisions in software. Objects are constructed in software which are modelled upon the behaviour and inter-relations of their real life counterparts [12]. This technique has already been applied to other applications with encouraging results [13,14]. With additional data, a model of the problem space can be produced which can mimic the basic decisions of a human designer.

4. EDGES - a Design Aid for the Electrical Protection and Switchgear for Hydro Plant

The design aid, EDGES (Embedded Design Generation and Emulation Software), which is currently under development, is written in C++; an object oriented programming language which was first released in 1980 by Bjarne Stroustrup of AT&T Bell Laboratories. It has proved to be a very popular computer language, and can be implemented on all major computing platforms. The operating system selected for development was Microsoft Windows NT due to its stability and capacity to build applications for Windows 95 or 98 environments.

Many decisions for the electrical specification of switchgear may be broken down into :

- the designer's discretion,
- regulatory requirements, and
- technical limitations.

Thus by the careful study of the design process for the switchgear and protection of small scale hydro schemes, the decision space can be intelligently narrowed down and certain decisions can be automated by the software.

Although hydro-electric sites have very individual characteristics, upon close examination it becomes apparent that most sites have common electrical characteristics. For example, the grid configuration can normally be represented by one of a small number of generic circuits, as illustrated in Figure 1; this is not an exhaustive list of possible configurations, but indicates a first level of object abstraction or modelling.
Within each of these generic circuits there are components which are common to all hydro schemes, *e.g.* lines and/or cables, transformers, generators, circuit breakers and protection equipment. Again, each of these components may be split into sub-components or functional blocks. Protection equipment, even on a simple

scheme, includes an over-current relay, an under- and over-frequency relay and an over-voltage relay. Embedded schemes should also include loss of mains, earth fault, reverse power, directional over-current, and neutral voltage displacement (NVD) relays. At this level of abstraction it is possible to create each of these relays as objects in the model and assign characteristics to them; some of a general nature such as cost, size, weight and rating; some specific to relays such as operating time, protection class current or voltage transformer requirements; and electrical connectivity to other devices, such as control signals. If this process is repeated for the other circuit elements which construct a possible scheme then a complex model may be constructed from objects which have the same attributes as their real life counterparts.

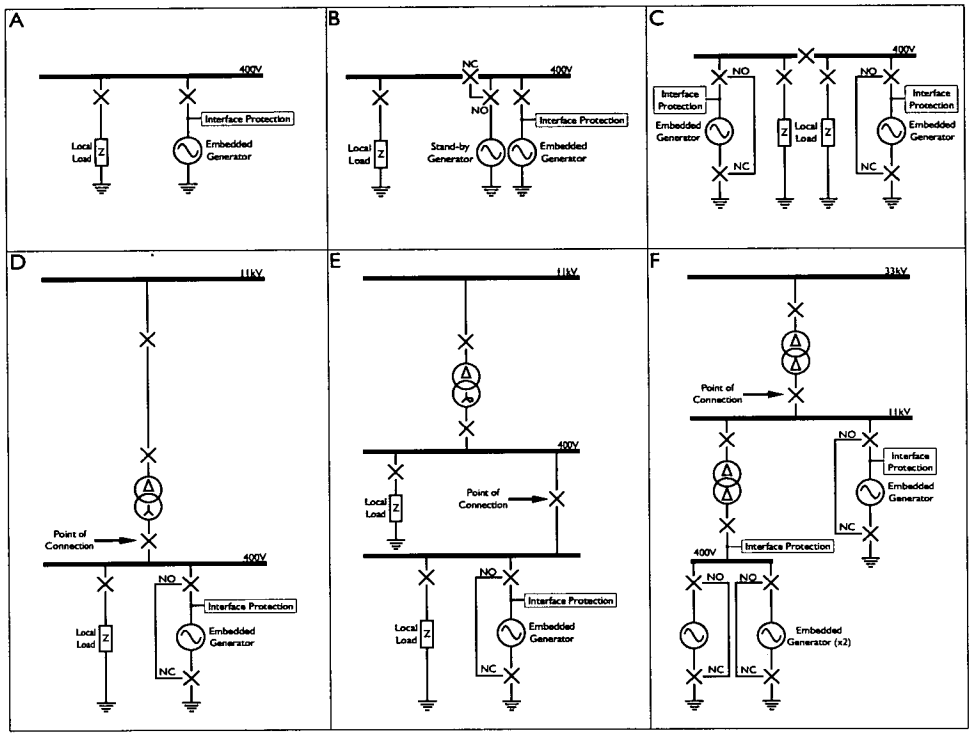


Figure 1 : A selection of generic circuits which characterise many typical topologies of small hydro schemes.

To analyse a design scenario the user should select a generic circuit to which he may add or remove circuit elements to suit his requirements, and then give some initial information concerning the electrical environment of the proposed hydro development. This information is submitted through the Electrical Data dialog box, as reproduced in Figure 2, and is stored in an object called Electrical Environment. If the data is known to be accurate, it should be entered as Confirmed Data; however, during initial exploration of a design scenario it is common not to have complete and/or accurate data for the scheme. In such cases a sensible estimate should be entered as Provisional Data: this allows the design to proceed, but flags to the user that the data should be checked at a later date.

Whenever a new device is connected to the design (for example a second generator or a measurement class current transformer) the device inherits the electrical parameters relevant to its operation from the object Electrical Environment. If insufficient information is available then EDGES will request the additional details from the user.

Interrelated groups of devices may also be modelled. For instance, the user may wish to add a new protection device to the design model, such as an under-and over-frequency relay. Having selected a new relay from the database, EDGES will insert the relay into the design, connect it to a single phase or three phase (depending upon the user's preference) protection class voltage transformer, ensure that this additional burden does not over load the voltage transformer (if this burden proves too great, the voltage transformer is up-graded to the next standard size, and the transformer fuse ratings re-checked), and connect the trip circuit of the relay to the main circuit breaker.

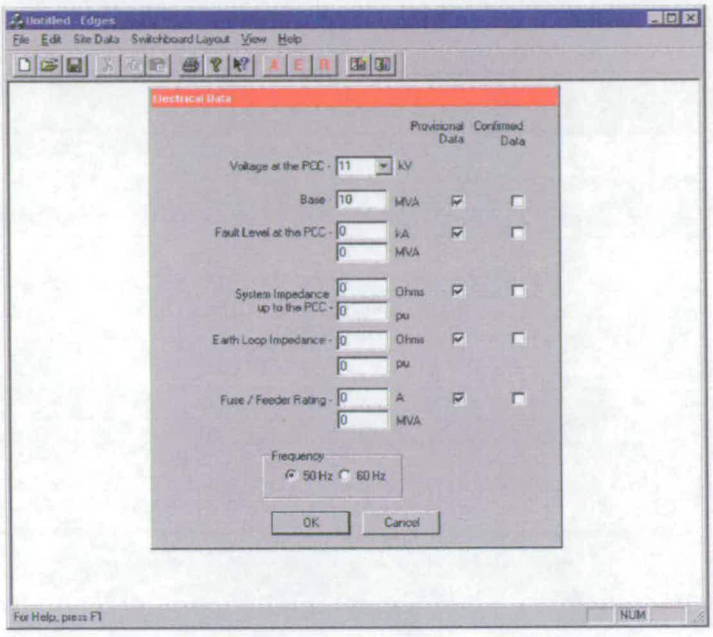


Figure 2 : The Electrical Data Dialog (a screen shot from EDGES).

Thus by the modelling of circuit components, the inheritance structure of EDGES ensures that all the necessary components for a proposed scheme are selected and specified correctly, hence avoiding costly modifications and omissions. During the design process EDGES also allows for varying degrees of user control, depending upon the users experience. Considering the previous example of the addition of a protection relay; an expert user may be prepared to accept that the voltage transformer will be slightly over-loaded and choose not to perform the up-grade, whereas an inexperienced user would not be able to make such a judgment, and would therefore be made aware of the automatic up-grade. However, throughout the design process all major decisions are vetted my the user.

The grouping of objects also allows various electrical standards to be applied. As previously mentioned, within the United Kingdom, all small scale hydro schemes embedded into the distribution grid have to comply with G59/1. By selecting G59/1 from the standards menu, EDGES will insert all the protection relays required into the design prototype based upon the plant size, number of generators and purpose. Once a complete design has been chosen and examined by the developer, then EDGES will produce a complete specification of all equipment and components, including spares items such as fuses and contactors. The specification for the hydro plant may then be procured and constructed.

5. Future Development

It is hoped that once current work upon EDGES is complete the follow features could also be added :

- Extend EDGES to analyse hydrological data to aid turbine selection,
- Expand concepts embodied in EDGES to the control system of the plant,
- Include a flexible cost database to allow for a basic economic analysis,
- Construct an interface to a third party electrical power systems analysis software, such as EMTP/ATP or ERACS, to examine in greater detail the effect of the scheme on the electrical grid system (stability, fault currents, voltage drops etc.).

6. Conclusions

The main advantages of EDGES to developers are :

- More rational, consistent design specifications are produced, based upon standard components,
- Omissions in the specification of a scheme are removed,
- Reduced design time on each project, as time consuming checking of components is automated,
- Reduced commissioning times by ensuring a complete and checked design is tendered,
- Increased up-time during operation and extended lifetime of the plant as standard and proven components are selected and designed to suitable standards,
- Long term-cost effective designs can be produced very quickly and alternative possibilities explored.

EDGES could have been implemented using an artificially intelligent technique based upon expert systems. The advantages of using an object oriented design approach as opposed to an expert system approach are :

- The code, C++, may be ported onto a large range of computing platforms (except for the graphical user interface) whereas an expert system shell is normally restricted to one computing platform,
- Extensive checking and rule validation is unnecessary as the intelligent behaviour is encapsulated within easily understood objects,
- The behaviour of individual objects may be modified without affecting the rest of the model,
- Fast operation, as computationally expensive rule checks are avoided,
- Once constructed objects may be re-used or advanced objects constructed from basic objects thus allowing the re-use of code and economy of expression,
- Object oriented systems are more resilient to change.

Hydro power development throughout the world will continue to expand, especially in developing countries where experience and knowledge concerning the design and implementation of new plant is scarce. It is hoped that through the use of EDGES mini and small scale hydro development will become quicker to design, cheaper to implement, more reliable and give longer service to the communities which they serve.

References

1. Bartle, A. and Taylor, R. (Ed's). *Hydropower and Dams 1997 World Atlas and Industry Guide,* The International Journal on Hydropower and Dams, Friary Press, UK, pp. 4-5, 1997.
2. Ibid. 1, pp. 5-6.
3. Electricity Association, *Engineering Recommendation G. 59 : Recommendations for the Connection of Embedded Generation Plant to the Regional Electricity Companies' Distribution Systems (with Amendments, 1992 and 1995),* Electricity Association : Engineering and Safety Publications, UK, 1991.
4. Electricity Association, *Engineering Report No. 113, Revision 1, Notes of Guidance for the Protection of Embedded Generation Plant up to 5MW for Operation in Parallel with Public Electricity Companies' Distribution Systems,* Electricity Association : Engineering and Safety Division Publications, UK, 1995.
5. Godart, T.F., Puttgen, H.B. and Jansen, J.F. *Expert system applications to power engineering control design.* Proceedings of the 5th IEEE International Symposium on Intelligent Control. IEEE Computer Soc., Press 2 vol. xxii, pp. 1145-50, 1990.
6. Grierson, D.E. and Hajela, P. (Ed's). *Emergent Computing Methods in Engineering Design.* NATO ASI Series, Vol. 149, pp. 150-161, 1996.
7. Simon, H.A. *The sciences of the Artificial,* MIT Press, pp55-56, 1968.
8. Warwick, K., Ekwue, A. and Aggarwal, R. (Ed's). *Artificial Intelligence Techniques in Power Systems.* IEE Power Engineering Series 22, pp1-18, 1997.
9. King, T.D., El-Hawary, M. E. and El-Hawary F. *Optimal Environmental Dispatching of Electric Power Systems via an Improved Hopfield Neural Network Model,* IEEE Transactions on Power Systems, Vol. 10, No. 3, August, 1995.
10. Furuyama, Y. and Chiang, H. D. *A Parallel Genetic Algorithm for Service Restoration in Electric Power Distribution Systems,* International Journal of Electrical Power and Energy Systems, Vol. 18, No. 2, pp. 275-282, 1996.
11. Young, D. J., Lo, K. L., McDonald, J. R., Howard, R. and Rye, J. *Development of a Practical Expert System for Alarm Processing,* IEE Proceedings C, Vol. 139, No. 5, pp. 437-447, 1992.
12. Daniels, J. *Object Methods and Tools.* IEE Colloquium on 'Distributed Object Management'. London, UK, p3, 1994.
13. Booch, G. *Object oriented design : with applications.* Benjamin/Cummings Press, 1994.
14. Shlaer, S. Object lifecycles : modelling the world in states. Yourdon Press, 1992.

References

[1] Raj Aggarwal and Yonghua Song. Artificial neural networks in power systems: Part 1 - General introduction to neural computing. *The Institute of Electrical Engineers (IEE) Power Engineering Journal*, 11(3):129 – 134, June 1997. ISSN: 0950-3366.

[2] Raj Aggarwal and Yonghua Song. Artificial neural networks in power systems: Part 3 - Examples of applications in power systems. *The Institute of Electrical Engineers (IEE) Power Engineering Journal*, 12(6):279 – 287, December 1998. ISSN: 0950-3366.

[3] Varol Akman, Paul J.W. ten Hagen, and Tetsuo Tomiyama. Design as a Formal Knowledge Engineered Activity. Technical Report CS-R8744, Center for Mathematics and Computer Science (CWI), P.O. Box 4079, 1009 AB Amsterdam, The Netherlands, 1987.

[4] Jarmo T. Alander. Indexed bibliography of genetic algorithms in power engineering. Report 94-1-POWER, Department of Information Technology and Production Economics, University of Vaasa, P.O. Box 700, FIN-65101 Vassa, Finland, 16th August 1999. Obtained from the internet: www: www.uwasa.fi/~jal or ftp: ftp.uwasa.fi/cs/report94-1/gaPOWERbib.ps.Z.

[5] Christopher Alexander. *Notes on the Synthesis of Form*. Harvard University Press, Cambridge, Massachusetts, US, 1964. ISBN: 067-4627504.

[6] W. J. J. Allen. 'Quality Improvement in Printed Board Assembly Design'. In *Intelligent Design Systems*, number 97/016 in Colloquium Proceedings, pages 3/1 - 3/4. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1997.

[7] Alsthom T&D Protection and Control Limited, St. Leonard Works, Stafford, ST17 4LX, UK. *LGPG111: Digital Integrated Generator Protection Relay*, 1996. Publication Code: R4106E.

[8] P.M. Anderson. *Power System Protection*. IEEE Press Power Engineering Series. McGraw-Hill and Institute of Electrical and Electronic Engineers (IEEE) Press, 1999. ISBN: 0-7803-3427-2.

[9] Teresa Mary Anderson. *MICADO: A System of Decision Support Software for Micro Hydro Power in Nepal*. PhD thesis, The Energy Systems Group, Department of Electrical Engineering, The University of Edinburgh, Kings Buildings, Mayfield Road, Edinburgh, Scotland, UK, EH9 3JL, 1992.

[10] Tim Anderson. The Developing World. *Personal Computer World*, 23(4):214 – 230, April 2000. Internet homepage: http://www.vnunet.com.

[11] Nick Angelopoulos. Electrical fuses. In Richard C. Dorf, editor, *The Electrical Engineering Handbook*, chapter 1.4, pages 39 – 45. CRC Press Inc., London, Tokyo, 1993. ISBN: 0-8493-0185-8.

[12] Morris Asimow. *Introduction to Design*. Prentice-Hall Series in Engineering Design. Prentice-Hall, Englewood Cliffs, N.J, 1962. Library of Congress Catalogue Card Number: 62-10550.

[13] Electricity Association. *Engineering Recommendation G.5 : Limits for Harmonics in the United Kingdom Electricity Supply System (3rd Revision)*. Electricity Association : Engineering and Safety Publications, 30 Millbank, London, SW1P 4RD, UK, 1976.

[14] Electricity Association. *Engineering Recommendation P.28 : Planning Limits for Voltage Fluctuations caused by Industrial, Commercial and Domestic Equipment in the United Kingdom Electricity Supply Industry*. Electricity Association : Engineering and Safety Publications, 30 Millbank, London, SW1P 4RD, UK, 1989.

[15] Electricity Association. *Engineering Recommendation G.59/1 : Recommendations for the Connection of Embedded Generation Plant to the Regional Electricity Companies' Distribution Systems (with Amendments, 1992 and 1995)*. Electricity Association : Engineering and Safety Publications, 30 Millbank, London, SW1P 4RD, UK, 1991.

[16] Electricity Association. *Engineering Recommendation G.75 : Recommendations for the Connection of Embedded Generation Plant to Public Electricity Suppliers' Distribution Systems above 20kV, with outputs over 5MW*. Electricity Association : Engineering and Safety Publications, 30 Millbank, London, SW1P 4RD, UK, 1991.

[17] Electricity Association. *Engineering Report No. 113, Revision 1, Notes of Guidance for the Protection of Embedded Generation Plant up to 5MW for Operation in Parallel with Public Electricity Companies' Distribution Systems*. Electricity Association : Engineering and Safety Division Publications, 30 Millbank, London, SW1P 4RD, UK, 1995.

[18] Djordje Atanackovic, Donald T. McGillis, Francisco D. Galiana, John Cheng, and Lester Loud. An Integrated Knowledge Based Model for Power System Planning. *Institute of Electrical and Electronic Engineers (IEEE) Expert*, 12(4):65 – 71, July - August 1997. ISSN: 0885-9000.

[19] Ahmed I. Baaleh and Ahmed F. Sakr. 'Application if Genetic Algorithms in Power System Stabilizer Design'. In Ibrahim Imam, Yves Kodratoff, Ayman El-Dessouki, and Moonsis Ali, editors, *Multiple Approaches to Intelligent Systems, Twelfth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-99)*, number LNAI 1611 in Lecture Notes in Artificial Intelligence; subseries of Lecture Notes in Computer Science, pages 165 – 174, Cairo, Egypt, 31st May – 3rd June 1999. International Society of Applied Intelligence, in cooperation with AAAI,ACM/SIGART, CSCSI, IEE, INNS, JSAI and SWT, Springer - Verlag, Berlin. ISBN: 3-540-66076-3.

[20] P. Baker. University of manchester institute of science and technology (umist) power systems club seminar. Mr Baker was is representative from the Department of Trade and Industry, 13th September 1999.

[21] H.G. Balcombe. The assessment and management of older oil-filled switchgear. *The Institute of Electrical Engineers (IEE) Power Engineering Journal: Special feature: Managing Older Oil-Filled Switchgear*, 11(6):234 – 238, December 1997. ISSN: 0950-3366.

[22] G. Barbian and G. Schlageter. Coda-a groupbase-system for cooperative design applications. In M. Huhns, M.P. Papazoglou, and G. Schlageter, editors, *Proceedings of International Conference on Intelligent and Cooperative Information Systems*, pages 220 – 228, Rotterdam, Netherlands, 12th – 14th May 1993. Institution of Electrical and Electronics Engineers, Computer Society Press, Los Alamitos, CA, USA. ISBN: 0-8186-3135-X.

[23] Vladimir Bazjanac. Architectural Design Theory: Models of the Design Process. In William R. Spillers, editor, *Basic Questions of Design Theory*, pages 3 –19. North-Holland Publishing Company and American Elsevier Publishing Company, Inc., 1974. ISBN: 0-7204-2818-1 or 0-444-10739-8.

[24] K. Beck and W. Cunningham. A Laboratory for Teaching Object Oriented Thinking. *SIGPLAN Notices (Association for Computing Machinery, Special Interest Group on Programming Languages)*, 24(10):1 – 6, October 1989. ISSN: 0362-1340.

[25] A. Bennett. 'Co-ordination of Control and Protection'. In *The Effective Response of a Public Electricity Network to Independent Generators*, number 93/093 in Colloquium Proceedings, pages 3/1 – 3/4. Institution of Electrical Engineers, Savoy Place, London, UK, 23rd April 1993.

[26] J. Bento, B. Feijó, and D.L. Smith. Engineering Design Knowledge Representation Based on Logic and Objects. *Computers and Structures*, 63(5):1015 – 1032, 1997. ISSN: 0045-7949.

[27] Ralph Bergmann, Sean Breen, Mehmet Göker, Michel Manago, and Stefan Wess. *Developing Industrial Case Based Reasoning Applications: The INRELA – Methodology*. Number LNAI 1612 in Lecture Notes in Artificial Intelligence; subseries of Lecture Notes in Computer Science. Springer - Verlag, Berlin, 1999. ISBN: 3-540-66182-4.

[28] N. Bianchi and S. Bolognani. Design Optimisation of Electric Motors by Genetic Algorithms. *Institute of Electrical Engineers (IEE) Proceedings: Electric Power Applications; Special Issue on Machines and Drives*, 145(5):475 – 483, September 1998. ISSN: 1350-2352.

[29] A.J Billington and C.S. Cox. 'Computational Aids for Control Systems Analysis and Design: an Object Orientated Approach'. In *International Conference on Control '94*, pages 443 – 448, Coventry, UK, 21st – 24th March 1994. Institution of Electrical Engineers, Savoy Place, London, UK. IEE Conference Proceedings Number : 389. ISBN: 0-85296-610-5.

[30] Rob Black. *Design and Manufacture: An integrated approach*. Macmillan Press Ltd, Houndmills, Basingstoke, Hampshire, UK, 1996. ISBN: 0-333-60915.

[31] Grady Booch. *Object Oriented Design : with Applications.* The Benjamin/Cummings series in Object-Oriented Software Engineering. Addison-Wesley Publishing Company, 2725 Sand Hill Road, Menlo Park, CA 94025, second edition, 1994. ISBN: 0-8053-5340-2.

[32] Bimal K. Bose. Expert system, Fuzzy Logic, and Neural Network Applications in Power Electronics and Motion Control. *Proceedings of Institute of Electrical and Electronic Engineers (IEEE)*, 85(8):1303 – 1323, August 1994. ISSN: 0018-9219.

[33] L.W. Boxer. 'Environmental and Political Aspects of Renewable Energy'. In *International Conference on "Renewable Energy - Clean Power 2001"*, pages 7 – 12, London, UK, 17th – 19th November 1993. Institution of Electrical Engineers, Savoy Place, London, UK. ISBN: 0-85296-605-9.

[34] Dan Braha and Oded Maimon. The Design Process: Properties, Paradigms and Structure. *Institution of Electrical and Electronics Engineers (IEEE) Transactions on Systems, Man and Cybernetics; Part A: Systems and Humans*, 27(2):146 – 166, March 1997. ISSN: 1083-4427.

[35] Mark F. Bramlette and Eugene E. Bouchard. Genetic Algorithms in Parametric Design of Aircraft. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, chapter 10, pages 109 – 123. Van Nostrand Reinhold, New York, 1991. ISBN: 0-442-00173-8.

[36] Ivan Bratko. *PROLOG: Programming for Artificial Intelligence.* International Computer Science Series. Addison-Wesley, Wokinghan, England, 2 edition, 1990. ISBN: 0-201-41606-9.

[37] British Standards Institution, Chiswick High Road, London, W4 4AL. *British Standard 7625: Specification for Voltage Transformers*, 1993. Equivalent to IEC 60186:1986; implements CENELEC HD 554 S1:1992; ISBN: 0-580-21573-3.

[38] British Standards Institution, Chiswick High Road, London, W4 4AL. *British Standard 7626: Specification for Current Transformers*, 1993. Equivalent to IEC 60185:1987; implements CENELEC HD 554 S2:1993; ISBN: 0-580-21575-X.

[39] British Standards Institution, Chiswick High Road, London, W4 4AL. *British Standard 5311: High Voltage Alternating Current Circuit Breakers*, 1996. Equivalent to IEC 56:1987 + A1:1992 + A2:1995 + A3:1996 (modified); implements CENELEC HD 348 S7:1998;.

[40] Rodney A. Brooks. Elephants Don't Play Chess. In Pattie Maes, editor, *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*, A Special Issues of Robotics and Autonomous Systems, pages 3 – 15. Massachusetts Institute of Technology (MIT) Press and Elsevier Science Publishers, 1990. ISBN: 0-262-63135-0 paperback.

[41] David C. Brown. Understanding the Nature of Design. *The Institute of Electrical and Electronic Engineers (IEEE) Expert*, 12(2):14 – 16, March/April 1997. ISSN: 0885-9000.

[42] C++ Builder. ©Borland Inprise Inc., 2000. Internet homepage: http://www.borland.com/.

[43] I.W. Burgess and R.J. Plank. 'Computer Aided Design of Structural Frames'. In B.J. Davies, editor, *Proceeding of the Seventh Annual Design Engineering Conference (DES 84)*, pages 287 – 294, Birmingham, UK, 25th – 27th September 1984. IFS (Conferences) Ltd, IFS (Publishing) Ltd and North - Holland Publishing Company. ISBN: 0-903608-74-x.

[44] Matthew I. Campbell, Jonathan Cagan, and Kenneth Kotovsky. A–Design: An Agent-Based Approach to Conceptual Design in a Dynamic Environment. *Research in Engineering Design: Theory, Applications and Concurrent Engineering*, 11(3):172 – 192, 1999. Published by Springer - Verlag. ISSN: 0934-9839.

[45] Susan Carlson-Skalak, Michael D. White, and Yong Teng. Using an Evolutionary Algorithm for Catalog Design. *Research in Engineering Design: Theory, Applications and Concurrent Engineering*, 10(2):63 – 83, 1998. Published by Springer - Verlag. ISSN: 0934-9839.

[46] Lorna Carmichael and Audrey Mangum. 'Automating the Design Process: a Design Process Model Implementation'. In *AUTOTESTCON '94. IEEE Systems Readiness Technology Conference. 'Cost Effective Support Into the Next Century'*, pages 427 – 431, Anaheim, CA, USA, 20th – 22nd September 1994. Institution of Electrical and Electronics Engineers (IEEE), New York, NY, USA. ISBN: 0-7803-1910-9.

[47] B. Chandrasekaran, John R. Josephson, and V. Richard Benjamins. What are Ontologies, and Why Do We Need Them? *The Institute of Electrical and Electronic Engineers (IEEE) Intelligent Systems and their Applications*, 14(1):20 – 25, January / February 1999. ISSN: 0885-9000.

[48] Stephen P. Chapman. *Electric Machinery Fundamentals*. Electrical Engineering Series. McGraw-Hill International, second edition, 1991. ISBN:0-07-010914.

[49] X.R. Chen, N.C. Pahalawaththa, U.D. Annakkage, and C.S. Kumble. Design of Decentralised Output Feedback TCSC Damping Controllers by using Simulated Annealing. *Institute of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 145(5):553 – 558, September 1998. ISSN: 1350-2360.

[50] CLIPS. ©Software Technology Branch, National Aeronautics and Space Administration (NASA). Internet homepage: http://www.siliconvalleyone.com/clips.htm.

[51] Windsor Coles. Oil-filled switchgear with dependent manual initiation operating mechanisms – enforcements aspects. *The Institute of Electrical Engineers (IEE) Power Engineering Journal: Special feature: Managing Older Oil-Filled Switchgear*, 11(6):229 – 233, December 1997. ISSN: 0950-3366.

[52] Gary Connor. *The Externalities in Electricity Generation*. PhD thesis, Department of Electronic and Electrical Engineering, Faculty of Science and Engineering, University of Edinburgh, Energy Systems Group, Department of Electronic and Electrical Engineering, The University of Edinburgh, Kings Buildings, Mayfield Road, Edinburgh, Scotland, UK, EH9 3JL, September 2000.

[53] Rational Software Corporation. Unified Modeling Language (UML) Quick Reference for Rational Rose 4.0, January 1997. Obtained (at no cost) from Rational Software, Kingswood, Kings Ride, Ascot, SL5 8AD. Tel: 01344 295000. Internet: http://www.rational.com/uk.

[54] H. Cotton and H. Barber. *The Transmission and Distribution of Electrical Energy*, chapter 14: Circuit Breakers, pages 289 – 327. The English Universities Press Ltd, St. Paul's House, Warwick Street, London, EC4, third edition, 1970. ISBN: 0-340-04924-3.

[55] R.D. Coyne, M.A. Rosenman, A.D. Radford, M. Balachandran, and J.S. Gero. *Knowledge - Based Design Systems*. Addison-Wesley Teknowledge Series in Knowledge Engineering. Addison-Wesley Publishing Company, 1990. ISBN: 0-201-10381-8.

[56] Lucy Craig. 'Independent Generator Perspective: Wind Turbines as Embedded Generators'. In *System Implications of Embedded Generation and its Protection and Control*, number 98/277 in Colloquium Proceedings, pages 3/1 – 3/4. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1999. Colloquium was held in The Midlands Engineering Center, Austin Court, Birmingham, UK.

[57] G.D. Creamer and R. Leonard. 'The Design and Application of an Advanced CAD Method for Customer Quotations'. In B.J. Davies, editor, *Proceeding of the Sixth Annual Design Engineering Conference (DES 83)*, pages 159 – 169, Birmingham, UK, 4th – 6th October 1983. IFS (Conferences) Ltd, IFS (Publishing) Ltd and North - Holland Publishing Company. ISBN: 0-444-86817-8.

[58] Nigel Cross. *Engineering Design Methods*. John Wiley and Sons, Chichester, New York, Brisbane, Toronto, Singapore, second edition, 1994. ISBN: 0-471-94228-6.

[59] Nigel Cross and Anita Clayburn Cross. Expertise in Engineering Design. *Research in Engineering Design: Theory, Applications and Concurrent Engineering*, 10(3):141 – 149, 1998. Published by Springer - Verlag. ISSN: 0934-9839.

[60] Nigel Cross and Robin Roy. *Design methods manual*. Technology Second Level Course. Man-made Futures, Design and Technology; Units 13-16. Open University Press, PO Box 48, Milton Keynes, MK7 6AB, 1975. ISBN: 0-335-02908-6.

[61] Peter Crossley, Steve Haslam, and Nick Jenkins. 'Evaluation of a New Type of Protection Relay for Wind Farms'. In *System Implications of Embedded Generation and its Protection and Control*, number 98/277 in Colloquium Proceedings, pages 7/1 – 7/6. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1999. Colloquium was held in The Midlands Engineering Center, Austin Court, Birmingham, UK.

[62] Iraj Dabbaghchi, Richard D. Christie, Gary W. Rosenwald, and Chen-Ching Liu. AI Application Areas in Power Systems. *The Institute of Electrical and Electronic Engineers (IEEE) Expert*, 12(1):58 – 66, January – February 1997. ISSN: 0885-9000.

[63] Edward Dabrowski. 'Embedded Generation Connection Issues – A REC's View'. In *System Utilisation*, pages 1 – 13. Yorkshire Electricity Group plc, Wetherby Road, Scarcroft, Leeds, 9th December 1996.

[64] Janet Daley. 'Design Creativity and Understanding of Objects'. In Nigel Cross, editor, *Developments in Design Methodology*, chapter 4.4, pages 291 – 302. John Wiley : Chichester, 1984. Originally published in *Design Studies*, volume 3, number 3, pages 133 – 137. Butterworth and Company (Publishers) Ltd.

[65] John Daniels. 'Object Design Methods and Tools'. In *Distributed Object Management*, number 98/007 in Colloquium Proceedings, pages 3/1 – 3/3. Institution of Electrical Engineers, Savoy Place, London, UK, 14th January 1994.

[66] S. Dasgupta. 'Herbert Simon's "Science of Design": two decades later'. In *First International Conference on Intelligent Systems Engineering*, pages 171 – 178, Edinburgh, UK, 19th – 21st August 1992. Institution of Electrical Engineers, Savoy Place, London, UK. IEE Conference Proceedings Number : 360. ISBN: 0-85296-549-4.

[67] Dilvan de Abreu Moreira and Les T. Walczowski. Using Software Agents to Generate VLSI Layouts. *The Institute of Electrical and Electronic Engineers (IEEE) Intelligent Systems and their Applications*, 12(6):26 – 32, November / December 1997. ISSN: 0885-9000.

[68] Anjan K. Deb. 'Object Oriented Expert System Estimates Line Ampacity'. *Institute of Electrical and Electronic Engineers (IEEE) Computer Applications in Power*, 8(3):30 – 35, July 1995. ISSN: 0895-0156.

[69] Kalyanmoy Deb and Mayank Goyal. 'Optimizing Engineering Designs using a Combined Genetic Search'. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 521 – 528, East Lansing, MI, 19th – 23rd July 1997. Michigan State University, Morgan Kaufmann Publishers Inc, 340 Pine Street, San Francisco, California. ISBN: 1-55860-487-1.

[70] Delphi. ©Borland Inprise Inc., 1995. Internet homepage: http://www.borland.com/.

[71] Roger Dettmer. *'Forum pounders embedded generation conundrum'*. IEE News, Institution of Electrical Engineers (IEE), Savoy Place, London, UK, May 2000. Number 157, Page 3, ISSN: 0308-0684.

[72] Roger Dettmer. Wind of Change. *Institution of Electrical Engineers (IEE) Review*, 46(3):21 – 24, May 2000.

[73] George E. Dieter. *Engineering Design: A Materials and Processing Approach.* McGraw-Hill Series in Mechanical engineering. McGraw-Hill Inc., New York, London, Tokyo, second edition, 1991. ISBN: 0-07-100829-2.

[74] R. H. Dinger. 'A Systematic Approach to Object Development'. In *Northcon/93*, pages 136 – 141, Portland, OR, USA, 12th – 14th October 1993. Institution of Electrical and Electronics Engineers, New York, NY, USA. ISBN: 0-7803-9972-2.

[75] John R. Dixon and Corrado Poli. *Engineering Design and Design for Manufacturing: A Structured Approach.* Field Stone Publishers, 311 Field Hill Road, Conway, Massachusetts, USA, 1995. ISBN: 0-9645272-0-0.

[76] Eric A. Domeshek, Marcia F. Herndon, Andrew W. Bennett, and Janet L. Kolodner. 'A Case-Based Design Aid for Conceptual Design of Aircraft Subsystems'. In *Proceedings of the Tenth Conference on Artificial Intelligence for Applications*, volume 1, pages 63 – 69, San Antonia, TX, USA, 1st – 4th March 1994. Institution of Electrical and Electronics Engineers, Computer Society Press, Los Alamitos, CA, USA. ISBN: 0-8186-5550-X.

[77] Alex B. Duffy. The "What" and "How" of Learning in Design. *The Institute of Electrical and Electronic Engineers (IEEE) Expert*, 12(3):71 – 76, May/June 1997. ISSN: 0885-9000.

[78] Alex H. B. Duffy. 'An Approach to Developing Design Technology'. In *Design Technology : An Integrated Approach to Design of T & D Plant*, number 98/287 in Colloquium Proceedings, pages 1/1 – 1/13. Institution of Electrical Engineers, Savoy Place, London, UK, 17th June 1998.

[79] ERA Technology, To follow. *ERACS – Electric Power Systems Analysis Suite*, 1995. Report Number: 92-0716, Revision 1.1.

[80] A. Ertas and J.C. Jones. *The Engineering Design Process.* John Wiley and Sons Inc., 1993. ISBN: 0-471-51796-8.

[81] Gareth Evans and Anthony Price. Regenysis: The Impact of Electrical Storage as Embedded Generation. In *Embedded Generation*, On-Day Conference in the 'Hot Topic' series. The Institute of Electrical Engineers (IEE), Savoy Place, London, UK WC2R 0BL, 28th February 2000. Regenysis is part of National Power's npower division.

[82] H-Y Fan. An inverse design method of diffuser blades by genetic algorithms. *Proceedings of the Institution of Mechanical Engineers Part A – Journal of Power and Energy*, 212(A4):261 – 268, 1998. ISSN: 0957-6509.

[83] F. Fantozzi and U. Desideri. Simulation of power plant transients with artificial neural networks: application to an existing combined cycle. *Proceedings of the Institution of Mechanical Engineers Part A – Journal of Power and Energy*, 212(A5):299 – 313, 1998. ISSN: 0957-6509.

[84] Alan R. Feuer. *MFC Programming.* The Advanced Windows Series. Addison-Wesley Developers Press, May 1997. ISBN: 0-201-63358-2.

[85] Donald G. Fink and H. Wayne Beaty, editors. *The Standard Handbook for Electrical Engineers*, chapter 10: Power System Components. McGraw-Hill, New York, London, Madrid, Sydney and Tokyo, fourteenth edition, 2000. ISBN: 0-07-022005-0.

[86] Lester H. Fink, Kan-Lee Liou, and Chen-Ching Liu. From generic restoration actions to specific restoration strategies. *Institution of Electrical and Electronics Engineers Transactions on Power Systems*, 10(2):745 – 752, May 1995. ISSN: 0885-8950.

[87] D.S. Fitton, R.W. Dunn, R.K. Aggarwal, A.T. Johns, and A. Bennett. Design and Implementation of an Adaptive Single Pole Autoreclosure Technique for Transmission Lines using Artificial Neural Networks. *The Institute of Electrical and Electronic Engineers (IEEE) Transactions on Power Delivery*, 11(2):748–56, April 1996. ISSN: 0885-8977.

[88] David B. Fogel. 'Evolutionary Computation : An Introduction, Some Current Applications, and Future Directions'. In Shun ichi Amari and Nikola Kasabov, editors, *Brain-Like Computing and Intelligent Information Systems*, chapter 12, pages 275 – 292. Springer, 1997. ISBN: 981-3083-58-1.

[89] Mike Foley and Anjan Bose. 'Object - Oriented On-Line Network Analysis'. *Institute of Electrical and Electronic Engineers (IEEE) Transactions on Power Systems*, 10(1):125 – 132, February 1995. ISSN: 0885-8950.

[90] Martin Fowler and Scott Kendall. *UML Distilled: Applying the Standard Object Modeling Language*. The Addison-Wesley Object Technology Series. Addison-Wesley, Reading, Mass., Harlow, 1997. ISBN 0-201-32563-2.

[91] Francisco D. Galiana, Donald T. McGillis, and Miguel A. Marin. Expert systems in transmission planning. *Proceedings of the Institute of Electrical and Electronic Engineers (IEEE)*, 80(5):712 – 726, May 1992. ISSN: 0018-9219.

[92] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *'Design Patterns : Elements of Reusable Object-Oriented Software'*. Addison-Wesley Professional Computing Series. Addison-Wesley, One Jacob Way, Reading, Massachusetts, 1995. ISBN: 0-201-63361-2.

[93] General Electric Power Management, 215 Anderson Avenue, Markham, Ontario, Canada L6E 1B3. *PQM: Power Quality Meter*, July 2000. Available on the internet at: http://www.geindustrial.com/pm/brochure/pqm/index.htm.

[94] E. Georgin, F. Bordin, S. Lœsel, and J.R. McDonald. 'Case - Based Reasoning Applied to Fault Diagnosis on Steam Turbines'. In Ian D. Watson, editor, *Progress in Case - Based Reasoning, First United Kingdom Workshop*, number LNAI 1020 in Lecture Notes in Artificial Intelligence; subseries of Lecture Notes in Computer Science, pages 175 – 184, Salford, UK, 12th January 1995. AI-CBR and British Computer Society Specialist Group on Expert Systems, Springer - Verlag, Berlin. ISBN: 3-540-60654-8.

[95] E. Georgin, F. Bordin, and J.R. McDonald. 'Using Prototypes in Case Based Diagnosis of Steam Turbines'. In *Case Based Reasoning: Prospects for Applications*, number 95/047 in Colloquium Proceedings, pages 1/1 – 1/4. Institution of Electrical Engineers, Savoy Place, London, UK, 7th March 1995. Organised by Professional Group C4 (Artificial Intelligence) for the Computing and Control Division, ISSN: 0963-3308.

[96] John S. Gero and Mary Lou Maher, editors. *Modeling Creativity and Knowledge-Based Creative Design*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1993. ISBN:.

[97] T.F. Godart, H.B. Puttgen, and J.F. Jansen. 'Expert System Applications to Power Engineering Control Design'. In *Proceedings of the 5th IEEE International Symposium on Intelligent Control*, volume xxii, pages 1145 – 1150,

Philadelphia, PA, USA, 5th – 7th September 1990. IEEE Computer Society, Press 2. ISBN: 0-8186-2108-7.

[98] Ashok K. Goel. Design, Analogy and Creativity. *The Institute of Electrical and Electronic Engineers (IEEE) Expert*, 12(3):62 – 70, May/June 1997. ISSN: 0885-9000.

[99] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine learning*. Addison - Wesley Publishing Company, Inc, 1989. ISBN: 0-201-15767-5.

[100] Peter Gordon. 'Developments from the R&D Community: Control and Protection of Diesel Gen-Sets'. In *System Implications of Embedded Generation and its Protection and Control*, number 98/277 in Colloquium Proceedings, pages 6/1 – 6/5. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1999. Colloquium was held in The Midlands Engineering Center, Austin Court, Birmingham, UK.

[101] S.R. Gorti, A. Gupta, G.J. Kim, R.D. Sriram, and A. Wong. An Object-oriented Representation for Product and Design Processes. *Computer Aided Design*, 30(7):489 – 501, June 1998. ISSN: 0010-4485.

[102] The United Kingdom Government. *Utilities Act*. The Stationery Office Limited, 51 Nine Elms Lane, London, SW8 5DR, July 2000. ISBN: 0-10-542700-4. Also available on the internet at http://www.legislation.hmso.gov.uk/acts/acts2000/20000027.htm.

[103] D. Green. 10GW by 2010? – Prospects for CHP in the UK. In *Conference on CHP 2000: Co-Generation for the 21st Century*, number 1998-1 in IMechE Conference Transactions, pages 3 – 10, IMechE Headquarters, London, UK, 18th – 19th February 1998. Institution of Mechanical Engineers (IMechE), Mechanical Engineering Publications Limited for IMechE. ISSN: 1356-1446, ISBN: 1-86058-141-2.

[104] Donald E. Grierson. Conceptual Design using Emergent Computing Techniques. In Donald E. Grierson and Prabhat Hajela, editors, *Emergent Computing Methods in Engineering Design : Applications of Genetic Algorithms and Neural Networks*, volume 149 of *NATO ASI Series F, Computer and Systems Sciences*, pages 150 – 161. Springer-Verlag in cooperation with NATO Scientific Affairs Division, London, Berlin, New York, 1996. ISBN: 3-540-60873-7.

[105] Groupe Schneider, 19 Waterman Avenue, Toronto, Ontario, M4B 1Y2. *Protection and Control: Sepam Range (2000, 1000, 100)*, February 1998. Internet copies of this manual are available from www.schneider.ca.

[106] Anurag P. Gupta and Daniel P. Siewiorek. 'M1: A Small Computer System Synthesis Tool'. In *Sixth Conference on Artificial Intelligence Applications*, volume 1, pages 230 – 236, Santa Barbara, CA, USA, 5th – 9th May 1990. Institution of Electrical and Electronics Engineers, Computer Society Press, Los Alamitos, CA, USA. ISBN: 0-8186-2032-3.

[107] Peter Hammond and John C. Davenport. 'A Logic-Based Model of Prosthesis Design'. In *Intelligent Design Systems*, number 97/016 in Colloquium

Proceedings, pages 4/1 – 4/3. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1997.

[108] Edward B. Haugen. *Probabilistic Mechanical Design*. John Wiley and Sons Inc., New York, Chichester, Toronto, 1980. ISBN: 0-471-05847-5.

[109] Simon Haykin. *Neural Networks: A Comprehensive Foundation*, chapter 1, pages 1 –41. Macmillan College Publishing Company, New York, 1994. ISBN: 0-02-352761-7.

[110] G. Heggie and H.T. Yip. 'A Multi-function Relay for Loss of Mains Protection'. In *System Implications of Embedded Generation and its Protection and Control*, number 98/277 in Colloquium Proceedings, pages 5/1 – 5/4. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1998. Colloquium was held in The Midlands Engineering Center, Austin Court, Birmingham, UK.

[111] Sharam Hekmatpour. *Introduction to LISP and Symbol Manipulation*. Prentice Hall International Ltd, New York and London, 1988. ISBN: 0-13-486192-2.

[112] Brain Henderson-Sellers, Colin Atkinson, and Don Firesmith. Viewing OML as a Variant of the UML. In Robert France and Bernhard Rumpe, editors, *¡¡UML¿¿'99 - The Unified Modeling Language: Beyond the Standard; the second international conference proceedings*, volume LNAI 1723 of *Lecture Notes in Artificial Intelligence*, pages 49 – 66, Fort Collins, CO, USA, 28th – 20th October 1999. Springer - Verlag, Berlin, Heidelberg, New York, London. ISBN: 3-540-66712-1, mboxISSN: 0302-9743.

[113] Brain Henderson-Sellers and Ian Graham. Open: toward method convergence? *The Institute of Electrical and Electronic Engineers (IEEE) Computer*, 29(4):86 – 89, April 1996. ISSN: 0018-9162.

[114] James A. Hendler. Intelligent Agents: Where AI Meets Information Technology. *The Institute of Electrical and Electronic Engineers (IEEE) Expert*, 11(6):20 – 23, December 1996. ISSN: 0885-9000.

[115] Percy H. Hill. *The Science of Engineering Design*. Holt, Rinehart and Winston Inc., New York, London, Sydney, 1970. ISBN: 0-03-081390-5.

[116] P. Hoang and K. Tomsovic. Design and Analysis of an Adaptive Fuzzy Power System Stabilizer. *The Institute of Electrical and Electronic Engineers (IEEE) Transactions on Energy Conversion*, 11(2):455 – 461, June 1996. ISSN: 0885-8969.

[117] G. Hodgkinson. 'System Implications of Embedded Generation and its Protection and Control: PES Perspective'. In *System Implications of Embedded Generation and its Protection and Control*, number 98/277 in Colloquium Proceedings, pages 1/1 – 1/15. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1998. Colloquium was held in The Midlands Engineering Center, Austin Court, Birmingham, UK.

[118] J.E. Hodgson, K.R.W. Bell, and A.R.Daniels. Expert system based interactive power system planning facilities. *The Institute of Electrical Engineers (IEE) Power Engineering Journal*, 9(5):215 – 222, October 1995. ISSN: 0950-3366.

[119] John Holland. *Adaption in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* Massachusetts Institute of Technology (MIT) Press, second edition, 1992. ISBN: 0-262-082136 (Hardcover), 0-262-58111-6 (Paperback).

[120] Jan Rune Holmevik. *The History of Simula.* PhD thesis, Center for Technology and Society, University of Trondheim, N-7055 DRAGVOLL, Norway, 1995. Available on the internet: www.javasoft.com/people/jag/SimulaHistory.html.

[121] Ivor Horton. *Beginning Visual C++ 5.* Wrox Press Ltd., 30 Lincoln Road, Olton, Birmingham, UK B27 6PA, 1997. ISBN: 1-861000-08-1.

[122] Kamel Idir, Liuchen Chang, and Heping Dai. A Neural Network-Based Optimization Approach for Induction Motor Design. In *IEEE Canadian Conference on Electrical and Computer Engineering,* volume 2, pages 951 – 954. Institute of Electrical and Electronic Engineers (IEEE), New York, NY, USA, 26th – 29th May 1996. ISBN: 0-7803-3143-5.

[123] British Standards Institution and the Institute of Electrical Engineers. *Requirements for Electrical Installations (BS 7671:1992): IEE Wiring Regulations.* Institute of Electrical Engineers, Savoy Place, London, UK, 16th edition, 10th May 1992. ISBN: 0-85296-557-5.

[124] Instrument Data Communications Pty Ltd, 46 Central Road, Worcester Park, surrey KT4 8HY. *Practical Power Systems Protection for the Electrical Industry,* third edition, May 1998.

[125] Masahiro Inuiguchi, Tadahiro Miyake, Masatoshi Sakawa, and Isao Shiromaru. 'A Genetic Algorithm for Planning Coal Purchase of a Real Electric Power Plant'. In Takeshi Furuhashi and Yoshiki Uchikawa, editors, *Fuzzy Logic, Neural Networks, and Evolutionary Computation,* number LNAI 1152 in Lecture Notes in Artificial Intelligence; subseries of Lecture Notes in Computer Science, pages 198 –214, Nagoya, Japan, 14th – 15th November 1995. Institute of Electrical and Electronic Engineers (IEEE) and Nagoya University World Wisepersons Workshop, Springer - Verlag, Berlin. ISBN: 3-540-61988-7.

[126] Tarek Y. Ismail. 'The Implications of Embedded Generation on the NGC Transmission System'. In *System Implications of Embedded Generation and its Protection and Control,* number 98/277 in Colloquium Proceedings, pages 2/1 – 2/5. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1999. Colloquium was held in The Midlands Engineering Center, Austin Court, Birmingham, UK.

[127] L.C. Jain. 'Introduction to Knowledge-Based Systems'. In L.C. Jain, editor, *Proceedings of Electronic Technology Directions to the Year 2000,* pages 18 – 27, Adelaide, SA, Australia, 23rd – 25th May 1995. Institute of Electrical and Electronic Engineers (IEEE) Computer Society Press, Los Alamitos, CA, USA. ISBN: 0-8186-7085-1.

[128] Java. ©Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303 USA, 1995. Internet homepage(s): http://www.sun.com/; http://www.sun.co.uk; http://java.sun.com/.

[129] Frank Jay, editor. *The IEEE Standard Dictionary of Electrical and Electronics Terms.* Institute of Electrical and Electronic Engineers (IEEE) Inc., New York, NY, third edition, 1984. ANSI/IEEE Standard 100-1984, ISBN: 471-80787-7.

[130] N. Jenkins. Embedded Generation; part 1. *Institution of Electrical Engineers (IEE) Power Engineering Journal*, 9(3):145 – 150, June 1995. ISSN: 0950-3366.

[131] N. Jenkins. Embedded Generation; Part 2. *Institution of Electrical Engineers (IEE) Power Engineering Journal*, 10(5):233 – 239, October 1995. ISSN: 0950-3366.

[132] N. Jenkins and G. Strbac. 'Effects of Small Embedded Generation on Power Quality'. In *Issues in Power Quality*, number 95/222 in Colloquium Proceedings, pages 6/1 – 6/4, Coventry, UK, 28th November 1995. Institution of Electrical Engineers, Savoy Place, London, UK.

[133] Nick Jenkins, Ron Allan, Peter Crossley, Daniel Kirschen, and Goran Strbac. *Embedded Generation.* Number 31 in Power and Energy Series. The Institute of Electrical Engineers (IEE), Michael Faraday House, Six Hills Way, Stevenage, Herts SG1 2AY, 2000. ISBN: 0-85296-774-8.

[134] A.S. Jhutty. 'Embedded Generation and the Public Electricity System'. In *System Implications of Embedded Generation and its Protection and Control*, number 98/277 in Colloquium Proceedings, pages 9/1 – 9/14. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1999. Colloquium was held in The Midlands Engineering Center, Austin Court, Birmingham, UK.

[135] John Christopher Jones. 'A Method of Systematic Design'. In Nigel Cross, editor, *Developments in Design Methodology*, chapter 1.1, pages 9 – 31. John Wiley : Chichester, 1984. Originally published in J.C. Jones and D. Thornley (Editors) *Conference on Design Methods*, Pergamon Press Ltd, Oxford, 1963.

[136] Kappa-PC. ©IntelliCorp, Inc., 1975 El Camino Real West, Mountain View, CA 94040-2216 Tel: (650) 965-5500 Fax: (650) 965-5647 , 1992. Internet homepage: http://www.intellicorp.com/kappa-pc/default.htm.

[137] Mladen Kezunovic and Igor Rikalo. 'Detect and Classify Faults Using Neural Nets'. *Institute of Electrical and Electronic Engineers (IEEE) Computer Applications in Power*, 9(4):42 – 47, October 1996. ISSN: 0895-0156.

[138] H. Khatib. Electricity in the global energy scene. *Institution of Electrical Engineers (IEE) Proceedings – A – Science Measurement & Technology*, 140(1):24 – 28, January 1993. ISSN: 0960-7641.

[139] Hisham Khatib. 'Trends and Future Demand for Electric Power Generation'. In *International Conference on Opportunities and Advances in International Power Generation*, number 419 in IEE Conference Publications, pages 1 – 6, Durham, UK, 18th – 20th March 1996. Institution of Electrical Engineers, Savoy Place, London, UK. ISBN: 0-85296-655-5.

[140] Hiroaki Kitano. Nausicaä and the Sirens: A Tale of Two Intelligent Autonomous Agents. *The Institute of Electrical and Electronic Engineers (IEEE) Intelligent Systems and their Applications*, 11(6):60 – 61, December 1996. ISSN: 0885-9000.

[141] George Klir and Bo Yuan. 'Basic Concepts and History of Fuzzy Set Theory and Fuzzy Logic'. In Enrique H. Raspini, Piero P. Bonissone, and Witold Pedryiz, editors, *Handbook of Fuzzy Computation*, chapter A1.1, pages A1.1:1 – A1.1:9. Institute of Physics Publishing Ltd, Dirac House, Temple Back, Bristol, BS1 6BE, 1998. ISBN: 0-7503-0427-8.

[142] David J. Kruglinski, Scot Wingo, and George Shepherd. *Programming Microsoft Visual C++*. Microsoft Press, One Microsoft Way, Redmond, Washington, fifth edition, 1998. ISBN: 1-57231-857-0.

[143] Raymond Kurzweil. *The Age of Intelligent Machines*, chapter 6, pages 175 – 219. MIT Press, London, England, 1990. ISBN: 0-262-11121-7.

[144] Robert Lafore. *C++ Interactive Course*. Waite Group Press Inc. (a division of Sams Publishing), 200 Tamal Plaza, Corte, Madera, CA 94925, 1996. ISBN: 1-57169-063-8.

[145] Susan E. Lander. Issues in Multiagent Design Systems. *Institute of Electrical and Electronic Engineers (IEEE) Expert*, 12(2):18 – 26, March - April 1997. ISSN: 0885-9000.

[146] M.A. Laughton. 'Artificial Intelligence Techniques in Power Systems'. In Kevin Warwick, Arthur Ekwue, and Raj Aggarwal, editors, *Artificial Intelligence Techniques in Power Systems*, volume 22 of *IEE Power Engineering Series*, chapter 1, pages 1 – 18. Institution of Electrical Engineers, Savoy Place, London, UK, April 1997. ISBN: 0-85296-897-3.

[147] Therese Lawlor-Wright. 'Design for Testability of Printed Circuit Board Assemblies'. In *Intelligent Design Systems*, number 97/016 in Colloquium Proceedings, pages 1/1 – 1/4. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1997.

[148] Brain Lees. 'Engineering Design Support through Case-Based Reasoning'. In *Intelligent Design Systems*, number 97/016 in Colloquium Proceedings, pages 8/1 – 8/3. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1997.

[149] Wen-Yau Liang and Peter O'Grady. Design with Objects: an approach to object - oriented design. *Computer Aided Design*, 30(12):943 – 959, October 1998. ISSN: 0010-4485.

[150] Alstom T&D Distribution Switchgear Limited. VMX Indoor Circuit Breaker. Manufacturer's brochure, Higher Openshaw, Manchester, United Kingdom. M11 1FL, 1993. Reference Number: LA214-11D.

[151] Alstom T&D Distribution Switchgear Limited. The Metrik Indicating Instrument Guide. Manufacturer's brochure, Higher Openshaw, Manchester, United Kingdom. M11 1FL, 1998. Publication Number: I3-G21. Complete range of indicating instruments for both panel and switchboard mounting to monitor all power system parameters.

[152] Hawker Siddeley Switchgear Limited. Hawkvac 15; Medium Voltage Indoor Metalclad Vacuum Switchgear. Manufacturer's brochure, Newport Road, Blackwood, Gwent, United Kingdom. NP2 2XH, January 1998. Reference Number: B10.

[153] S. Limyingcharoen, U.D. Annakkage, and N.C. Pahalawaththa. 'Fuzzy logic based unified power flow controllers for transient stability improvement'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 145(3):225 – 232, May 1998. ISSN: 1350-2360.

[154] K.A. Lo, L.J. Peng, J.F. Macqueen, A.O. Ekwue, and D.T.Y. Cheng. 'Hybrid approach using counterpropagation neural network for power - system network reduction'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 144(2):169 – 174, March 1997. ISSN: 1350-2360.

[155] K.L. Lo and Sabir S. Ghauri. 'Expert System for the Design of Distribution Networks'. In *APSCOM-91. 1991 International Conference on Advances in Power System Control, Operation and Management*, number 348 in Conference Proceedings, pages 437 – 442, Hong Kong, 5th – 8th November 1991. Institution of Electrical Engineers, Savoy Place, London, UK. ISBN: 0-86341-246-7.

[156] K.L. Lo and I.Nashid. Expert systems and their application to power systems. part 1: Components and methods of knowledge representation. *The Institute of Electrical Engineers (IEE) Power Engineering Journal*, 7(1):41 – 45, February 1993. ISSN: 0950-3366.

[157] K.L. Lo and I.Nashid. Expert systems and their application to power systems. part 2: Search methods and languages. *The Institute of Electrical Engineers (IEE) Power Engineering Journal*, 7(3):141 – 44, June 1993. ISSN: 0950-3366.

[158] K.L. Lo, Y.J. Lin, and W.H. Siew. 'Fuzzy – logic method for adjustment of variable parameters in load – flow calculation'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 146(3):276 – 282, May 1999. ISSN: 1350-2360.

[159] George F. Luger and William A. Stubblefield. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison - Wesley Longman Inc., Harlow, England, third edition, 1998. ISBN: 0-805-31196.

[160] R.T. Lythall. *The J. & P. Switchgear Book: An Outline of Modern Switchgear Practice for the Non-Specialist User*. Iliffe Books Limited, sixth edition, 1969.

[161] D.E. Macpherson and E.A. Reddy. Expert System Based Switched Mode Power Supply Design. In *Sixth International Conference on Power Electronics and Variable Speed Drives*, pages 471 – 476. Institution of Electrical Engineers, Nottingham, UK, 23rd – 25th September 1996. Conference Publication No. 429. ISBN: 0-85296-665-2.

[162] S. Madan and K.E. Bollinger. Applications of artificial intelligence in power systems. *Electric Power Systems Research*, 41(2):117 – 131, May 1997. Elsevier Science S.A.

[163] Mary Lou Maher and Andrés Gómez de Silva Garza. Developing case-based reasoning for structural design. *The Institute of Electrical and Electronic Engineers (IEEE) Expert*, 11(3):42 – 56, June 1996. ISSN: 0885-9000.

[164] Mary Lou Maher and Andrés Gómez de Silva Garza. Case - Based Reasoning in Design. *The Institute of Electrical and Electronic Engineers (IEEE) Expert*, 12(2):34 – 41, March - April 1997. ISSN: 0885-9000.

[165] Michael Marcothy and Henry F. Ledgard. *Programming Language Landscape; Syntax, Semantics and Implementation*, chapter 1, pages 1 – 19. Science Research Associates (SRA) Inc., second edition, 1986. ISBN: 0-574-21945-5.

[166] Paul McCabe. Industrial experience, 19th – 27th July 1999. Experience gained from working at Caldwells Paper Mill, part of Inveresk plc, in Inverkeithing, Fife, Scotland.

[167] Warren S. McCulloch and Walter Pitts. A Logical Calculus of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 5:115 – 133, 1943. Reproduced in: 'Neurocomputing: Foundations of Research', Edited by James A. Anderson and Edward Rosenfeld, chapter 2, pages 15 – 27. MIT Press, Cambridge, London, 1988. ISBN: 0-262-01097-6.

[168] McGraw-Hill, editor. *McGraw-Hill Encyclopedia of Science and Technology*, volume 5 (DAC – ELA). McGraw-Hill Inc., New York, London, Sydney, Tokyo, eighth edition, 1997. ISBN: 0-07-911504-7 (set).

[169] Chris McMahon and Jimmie Browne. *CADCAM: From Principles to Practice.* Addison-Wesley Publishing Company, England, New York, Tokyo, Amsterdam, 1993. ISBN: 0-201-56502-1.

[170] General Electric Company (GEC) Measurements. *Protective Relays Application Guide.* Balding + Mansell Limited, London & Wisbech, GEC Measurements, St. Leonard's Works, Stafford, England, second edition, 1975.

[171] A.J. Medland and A. Jones. 'An Approach to Design Based upon Functional Logic'. In B.J. Davies, editor, *Proceeding of the Seventh Annual Design Engineering Conference (DES 84)*, pages 225 – 264, Birmingham, UK, 25th – 27th September 1984. IFS (Conferences) Ltd, IFS (Publishing) Ltd and North - Holland Publishing Company. ISBN: 0-903608-74-x.

[172] Cecil Thomas Melling. Nationalisation of Electricity Supply 1947: Reasons and Problems. *The Institute of Electrical Engineers (IEE) Power Engineering Journal*, 12(2):73 – 80, April 1998. ISSN: 0950-3366.

[173] William Mettrey. 'Expert Systems and Tools: Myths and Realities'. *Institute of Electrical and Electronic Engineers (IEEE) Expert*, 7(1):4 – 12, February 1992. ISSN: 0885-9000.

[174] Norman Meyrowitz, editor. *Proceedings of OOPSLA '86: Object Oriented Programming Systems, Languages and Applications*, SIGPLAN Notices, San Diego, California, November 1986. Association for Computing Machinery. Volume 21, Number 11, ISSN: 0362-1340.

[175] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer - Verlag, second edition, September 1992. ISBN: 3-540-58090-5.

[176] Microsoft, editor. *Microsoft Visual C++ 6.0 MFC Library Reference*, volume 1 and 2. Microsoft Press, Redmond, US, 1998. ISBN: 1-57231-8651.

[177] G. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *The Psychological Review*, 63(2):86, March 1956. .

[178] Robert Milne and Chris Nelson. 'Electrical Drive Diagnosis using Case Based Reasoning'. In *Case Based Reasoning: Prospects for Applications*, number 95/047 in Colloquium Proceedings, pages 3/1 – 3/4. Institution of Electrical Engineers, Savoy Place, London, UK, 7th March 1995. Organised by Professional Group C4 (Artificial Intelligence) for the Computing and Control Division, ISSN: 0963-3308.

[179] Séan R. Mitchell, Roy Jones, and Chris Hinde. An Initial Data Model, Using the Object-Oriented Paradigm, for Sculptured-Feature-Based Design. *Research in Engineering Design: Theory, Applications and Concurrent Engineering*, 7(1):19 – 37, 1995. Published by Springer - Verlag. ISSN: 0934-9839.

[180] Dunja Mladenic. Text - Learning and Related Intelligent Agents: A Survey. *The Institute of Electrical and Electronic Engineers (IEEE) Intelligent Systems and their Applications*, 14(4):44 – 54, July / August 1999. ISSN: 0885-9000.

[181] Osama A. Mohammed, Riaz S. Merchant, and Fuat G. Üler. Utilizing Hopfield Neural Networks and an Improved Simulated Annealing Procedure for Design Optimization of Electromagnetic Devices. *Institute of Electrical and Electronic Engineers (IEEE) Transactions on Magnetics*, 29(6):2404 – 2406, November 1993. 0081-9464.

[182] Osama A. Mohammed, Dong C. Park, Fuat G. Üler, and Chen Ziqiang. Design Optimization of Electromagnetic Devices using Artificial Neural Networks. *Institute of Electrical and Electronic Engineers (IEEE) Transactions on Magnetics*, 28(5):2805 – 2807, September 1992. 0081-9464.

[183] Jörg P. Möller. *The Design of Intelligent Agents: A Layered Approach*, chapter 2, pages 7 – 43. Number LNAI 1177 in Lecture Notes in Artificial Intelligence; Subseries of Lecture Notes in Computer Science. Springer - Verlag, Berlin Heidelberg, 1996. ISBN: 3-540-62003-6.

[184] H. Monsef, A.M. Ranjbar, and S. Jadid. 'Fuzzy Rule-Based Expert System for Power System Fault Diagnosis'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 144(2):186 – 192, March 1997. ISSN: 1350-2360.

[185] Hans P. Moravec. *Mind Children : the Future of Robot and Human Intelligence.* Harvard University Press, Cambridge, Massachusetts, London, December 1988. ISBN: 0-674-57618-7.

[186] M.M. Morcos and W.R. Anis Ibrahim. 'Electric Power Quality and Artificial Intelligence: Overview and Applicability'. *Institute of Electrical and Electronic Engineers (IEEE) Power Engineering Review*, 19(6):5 – 10, June 1999. ISSN: 0272-1724.

[187] A. Moyes, G.M. Burt, J.R. McDonald, J.R. Capener, J.N. Dray, and R. Goodfellow. 'Combining design and operational knowledge to enhance generator plant diagnostics'. *Institution of Electrical Engineers Proceedings on Generation, Transmission and Distribution*, 143(3):300 – 304, May 1996. ISSN: 1350-2360.

[188] Alan F. Murray. Neural Networks – A Computing Revolution? Department of Electronics and Electrical Engineering, University of Edinburgh, Mayfield Road, Edinburgh, EH9 3JL, 1993.

[189] Patrick Naughton and Herbert Schildt. *Java 1.1: The complete Reference.* Osborne McGraw-Hill, 2600 Tenth Street, Berkeley, California 94710 USA, 1998. ISBN: 0-07-882436-2.

[190] D.T. Ndumu and H.S. Nwana. Research and Development challenges for agent - based systems. *Institution of Electrical Engineers (IEE) Proceedings: Software Engineering; Special Issue on Agent Based Systems*, 144(1):2 – 10, February 1997. ISSN: 1364-5080.

[191] John P. Nelson, P.K. Sen, and Andrew Leoni. The New Science of Protective Relaying in the Petro-Chemical Industry. *Institute of Electrical and Electronic Engineers (IEEE) Industrial Applications Magazine*, 6(2):34 – 42, March / April 2000. ISSN: 1077-2618.

[192] Andrew F. Neyer, Felix F. Wu, and Karl Imhof. Object - Oriented Programming for Flexible Software: Example of a Load Flow. *Institute of Electrical and Electronic Engineers (IEEE) Transactions on Power Systems*, 5(3):689 – 696, August 1990. ISSN: 0885-8950.

[193] H.S. Nwana and D.T. Ndumu. An Introduction to Agent Technology. In Hyacinth S. Nwana and Nader Azarmi, editors, *Software Agents and Soft Computing: Towards Enhancing Machine Intelligence; Concepts and Applications*, number LNAI 1198 in Lecture Notes in Artificial Intelligence; Subseries of Lecture Notes in Computer Science, pages 3 – 26. Springer - Verlag, Berlin Heidelberg, 1997. ISBN: 3-540-62560-7.

[194] Institution of Electrical Engineers (IEE). *'IEE submission adds a note of realism to debate on EU green energy plans'.* IEE News, Institution of Electrical Engineers, Savoy Place, London, UK, April 1999. Number 145, Page 3, ISSN: 0308-0684.

[195] Office of Electricity Regulation (OFFER). Fifth Renewables Order for England and Wales. London, September 1998.

[196] Office of Electricity Regulation (OFFER). Third Scottish Renewables Order. London, December 1998.

[197] Office of Electricity Regulation (OFFER). NFFO News. *New REview: The Quarterly Newsletter for the United Kingdom New and Renewable Energy Industry*, 1(43), February 2000. Available from the United Kingdom Government's Department of Trade and Industry (DTI) in either printed or electronic format. The electronic version is available in the internet from the DTI site at http://www.dti.gov.uk/NewReview/index.html.

[198] Department of Trade and Industry (DTI). UK Energy in Brief. Room 1128, Energy Policy and Analysis Unit 3a, 1 Victoria Street, London, SW1H 0ET, December 1999. DTI/Pub 4613/8k/12/99/AR, URN 99/220.

[199] Masaru Ohki, Eiji Oohira, Hiroshi Shinjo, and Masahiro Abe. Design Support to Determine Range of Design Parameters by Qualitative Reasoning. Technical Report TR-0790, Institute for New Generation Computer Technology (ICOT), Central Research Laboratory, Hitachi Ltd., 1–280 Higashi - Koigakubo, August 1992. ICOT, Mita Kokusai Bldg, 21F, 4–28 Mita, 1-Chome, Minato-ku, Tokyo, 108, Japan.

[200] P. O'Kane and B. Fox. 'The Impact of Loss of Mains Detection on Emergency Reserve Requirement'. In *System Implications of Embedded Generation and its Protection and Control*, number 98/277 in Colloquium Proceedings, pages 8/1 – 8/7. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1999. Colloquium was held in The Midlands Engineering Center, Austin Court, Birmingham, UK.

[201] Thomas C. Ormerod, John Mariani, Gary Spiers, Linden Ball, and Louise Maskill. 'Supporting the Process of Re-use in Innovative Design Environments'. In *Intelligent Design Systems*, number 97/016 in Colloquium Proceedings, pages 9/1 – 9/3. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1997.

[202] Grantham K.H. Pang. Fuzzy Neural Networks for Control Design. In Donald E. Grierson and Prabhat Hajela, editors, *Emergent Computing Methods in Engineering Design : Applications of Genetic Algorithms and Neural Networks*, volume 149 of *NATO ASI Series F, Computer and systems sciences*, pages 204 – 221. Springer, Berlin, New York, 1996. ISBN: 3-540-60873-7.

[203] Christiaan J.J. Paredis and Pradeep K. Khosla. 'Agent-Based Design of Fault Tolerant Manipulators for Satellite Docking'. In *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, Conference Proceedings, pages 3473 – 3480, Albuquerque, New Mexico, April 1997. Institution of Electrical and Electronics Engineers, New York, NY, USA. ISBN: 0-7803-3612-7-4.

[204] Ian C. Parmee and Harish D. Vekeria. 'Co-operative Evolutionary Strategies for Single Component Design'. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 529 – 536, East Lansing, MI, 19th – 23rd July 1997. Michigan State University, Morgan Kaufmann Publishers Inc, 340 Pine Street, San Francisco, California. ISBN: 1-55860-487-1.

[205] I.C. Parmee and M.J. Denham. The Development of Genetic Adaptive Search Tools for Engineering Design. In Donald E. Grierson and Prabhat Hajela, editors, *Emergent Computing Methods in Engineering Design : Applications of Genetic Algorithms and Neural Networks*, volume 149 of *NATO ASI Series F, Computer and systems sciences*, pages 82 – 103. Springer, Berlin, New York, 1996. ISBN: 3-540-60873-7.

[206] David Parsons. *Object Oriented Programming with C++*. Letts Educational, Aldine Place, London W12 8AW, second edition, 1997. ISBN: 1-85805-232-7.

[207] Lawrence J. Peters. *Software Design: Methods and Techniques*. Yourdon Press, 1133 Avenue of the Americas, New York, NY, 1981. ISBN: 0-917072-19-1.

[208] Charles J. Petrie. Agent – Based Engineering, the Web, and Intelligence. *The Institute of Electrical and Electronic Engineers (IEEE) Expert*, 11(6):24 – 29, December 1996. ISSN: 0885-9000.

[209] N.B.P. Pillips, J.O. Gann, and M.R. Irving. 'Object-Oriented Design and Implementation of Power System Analysis Software'. In Kevin Warwick, Arthur Ekwue, and Raj Aggarwal, editors, *Artificial Intelligence Techniques in Power Systems*, volume 22 of *IEE Power Engineering Series*, chapter 3, pages 45 –

67. Institution of Electrical Engineers, Savoy Place, London, UK, April 1997. ISBN: 0-85296-897-3.

[210] J-C. Popieul and J-C. Anguè. 'From AMS Design to Realization: Methods and Tools'. In *Proceedings of the IECON '93. International Conference on Industrial Electronics, Control, and Instrumentation*, volume 1, pages 553 - 558, Maui, HI, USA, 15th - 19th November 1993. Institution of Electrical and Electronics Engineers; Industrial Electronics and Control Division, New York, NY, USA. ISBN: 0-7803-0891-3.

[211] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brain P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition, 1992. ISBN: 0-521-43108-5.

[212] R. Ramakumar, I. Abouzahr, and K. Ashenayi. 'A Knowledge-based Approach to the Design of Integrated Renewable Energy Systems'. *Institution of Electrical and Electronics Engineers Transactions on Energy Conversion*, 7(4):648 - 659, December 1992. ISSN: 0885-8969.

[213] Nutakki D. Rao and Yimin Zhang. 'An Intelligent Front End for Secondary Power Distribution System Design'. *Institution of Electrical and Electronics Engineers Transactions on Power Delivery*, 7(2):927 - 933, April 1992. ISBN: 0-7803-0219-2.

[214] Khaled Rasheed and Haym Hirsh. 'Using Case-Based Learning to Improve Genetic - Algorithm - Based Design Optimization'. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 513 - 520, East Lansing, MI, 19th - 23rd July 1997. Michigan State University, Morgan Kaufmann Publishers Inc, 340 Pine Street, San Francisco, California. ISBN: 1-55860-487-1.

[215] Enrique H. Raspini and E.H. Mamdani. 'Why Fuzzy Logic?'. In Enrique H. Raspini, Piero P. Bonissone, and Witold Pedryiz, editors, *Handbook of Fuzzy Computation*, chapter A1.1, pages A2.1:1 - A2.1:9. Institute of Physics Publishing Ltd, Dirac House, Temple Back, Bristol, BS1 6BE, 1998. ISBN: 0-7503-0427-8.

[216] B. Ravindranath and M. Chander. *Power System Protection and Switchgear*. Wiley Eastern Limited, New Delhi, Bangalore, Bombay, Calcutta, 1977. ISBN: 0-85226-758-4.

[217] Amarnath Reddy. *Expert System Based Switched Mode Power Supply Design*. PhD thesis, The Energy Systems Group, Department of Electrical Engineering, The University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh, UK, October 1997.

[218] M. Reformat, E. Kuffel, D. Woodford, and W. Pedrycz. Application of genetic algorithms for control design in power systems. *Institution of Electrical Engineers(IEE) Proceedings. Generation, Transmission & Distribution*, 145(4):345 - 354, July 1998. ISSN: 1350-2360.

[219] Michael M. Richer. Introduction. In Mario Lenz, Brigitte Bartsch-Spörl, Hans-Dieter Burkhard, and Stefan Wess, editors, *Case Based Reasoning Technology:*

from Foundations to Applications, number LNAI 1400 in Lecture Notes in Artificial Intelligence; subseries of Lecture Notes in Computer Science, chapter 1, pages 1 – 15. Springer - Verlag, Berlin, 1998. ISBN: 3-540-64572-1.

[220] C.A. Ritson and F.M. Stansfield. 'Low Cost Computing for Machine Design'. In B.J. Davies, editor, *Proceeding of the Sixth Annual Design Engineering Conference (DES 83)*, pages 339 – 359, Birmingham, UK, 4th – 6th October 1983. IFS (Conferences) Ltd, IFS (Publishing) Ltd and North - Holland Publishing Company. ISBN: 0-444-86817-8.

[221] Horst W.J. Rittel. Some Principles for the Design of an Educational System for Design. *Journal of Architectural Education*, XXVI(1-2):16 – 26, Winter-Spring 1971. This paper was originally printed in the conference proceedings entitled *Education for Technology*, at Washington University, St. Louis, MO, US, 1967.

[222] G.D. Rockefeller. Fault Protection with a Digital Computer. *Institute of Electrical and Electronic Engineers (IEEE) Transactions on Power Apparatus and Systems*, PAS-88(4):438 – 464, April 1969. ISSN: 0885-8977.

[223] Paul A. Rodgers, Avon P. Huxor, and Nicholas H.M. Caldwell. Design Support Using Distributed Web-Based AI Tools. *Research in Engineering Design: Theory, Applications and Concurrent Engineering*, 11(1):31 – 44, 1999. Published by Springer - Verlag. ISSN: 0934-9839.

[224] Walter Rodriguez. *The Modeling of Design Ideas: Graphics and Visualization Techniques for Engineers*. Engineering Drawing Series. McGraw - Hill International Editions, 1992. ISBN: 0-07-053394-6.

[225] W.J.S. Rogers. 'Grid Connection of Embedded CHP Plants'. In *Developments in Combined Heat and Power (CHP) into the Millennium*, number 98/226 in Colloquium Proceedings, pages 4/1 – 4/11. Institution of Electrical Engineers, Savoy Place, London, UK, 20th May 1998. Colloquium was held in the Scottish Engineering Center (Teachers Building), Glasgow, UK.

[226] W.J.S. Rogers. 'Regulatory Regime for Parallel Operation of Generators in Public Electricity Distribution and Supply Networks'. In *System Implications of Embedded Generation and its Protection and Control*, number 98/277 in Colloquium Proceedings, pages 4/1 – 4/7. Institution of Electrical Engineers, Savoy Place, London, UK, 25th February 1999. Colloquium was held in The Midlands Engineering Center, Austin Court, Birmingham, UK.

[227] R. Rong and D.A. Lowther. Adapting Design Using Dimensional Models of Electromagnetic Devices. *Institute of Electrical and Electronic Engineers (IEEE) Transactions on Magnetics*, 32(3):1437 –1440, May 1996. ISSN: 0018-9464.

[228] R.W. Rong and D.A. Lowther. 'Storage and Retrieval of Solutions in the Design of Electromagnetic Devices'. *Institution of Electrical and Electronics Engineers Transactions on Magnetics*, 30(5):3648 – 3651, September 1994. ISSN: 0018-9464.

[229] T.F. Roylance, editor. *Engineering Design*. Pergamon Press Ltd, Symposium Publications Division, Oxford, London, Edinburgh, New York, 1966.

[230] James Rumbaugh. The life of an object model: How the object model changes during development. *The Journal of Object-Oriented Programming*, 7(1):24 – 32, March-April 1994. ISSN: 1097-1408.

[231] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, and William Lorenson. *Object-Oriented Modeling and Design*. Prentice-Hall International, Englewood Cliffs, New Jersey, London, UK, 1991. ISBN: 0-13-629841-9.

[232] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. The Addison-Wesley Object Technology Series. Addison-Wesley, Reading, Mass, 1999. ISBN: 0-201-30998-x.

[233] Stewart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall International Inc., 1995. ISBN: 0-13-360124-2.

[234] Michael D. Rychener, editor. *Expert Systems for Engineering Design*. Academic Press. Inc., 24-28 Oval Road, London NW1 7DX, UK, December 1988. ISBN: 0-12-605110-0.

[235] Z. Saad-Saoud, L.M. Craig, and N. Jenkins. Static VAr Compensators for Wind Energy Applications. In J.A. Halliday, editor, *Proceedings of the 17th British Wind Energy Association Conference*, pages 347 – 352, Warwick, UK, 19-21 July 1995. British Wind Energy Association. ISBN: 0-85298-961-X.

[236] Z. Saad-Saoud and N. Jenkins. 'The Application of Advanced Static VAr Compensators to Wind Farms'. In *Power Electronics for Renewable Energy*, number 97/170 in Colloquium Proceedings, pages 6/1 – 6/5. Institution of Electrical Engineers, Savoy Place, London, UK, 16th June 1997.

[237] Mohindar S. Sachdev, Pramod Dhakal, and Tarlochan S. Sidhu. Design Tool Generates Substation Interlock Schemes. *Institute of Electrical and Electronic Engineers (IEEE) Computer Applications in Power*, 13(2):37 – 42, April 2000. ISSN: 0895-0156.

[238] Thoman E. Sandman. 'Induction Support for KBDSS Development: a proposed system design'. In V. Milutinovic, B.D. Shriver, J.F. Nunamaker Jr., and R.H. Sprague Jr., editors, *Proceedings of the Twenty-Fifth Hawaii International Conference on System Sciences*, volume 3, pages 4 – 13, Kauai, HI, USA, 7th – 10th January 1992. Institution of Electrical and Electronics Engineers, Computer Society Press, Los Alamitos, CA, USA. ISBN: 0-8186-2420-5.

[239] Roger C. Schank and Robert P. Abelson. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. The Artificial Intelligence Series. Lawrence Erlbaum Associates, Inc., Hillside, New Jersey, US, 1977. ISBN: 0-470-99033-3.

[240] H.P. Schmidt. 'Application of Artificial Neural Networks to the Dynamic Analysis of the Voltage Stability Problem'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 144(4):371 – 376, July 1997. ISSN: 1350-2360.

[241] Robert Sedgewick. *Algorithms in C*. Addison - Wesley Longman Inc., Harrow, England, second edition, 1988. ISBN: 0-201-31452-5.

[242] James N. Siddall. *Probabilistic Engineering Design : principles and applications.* Mechanical Engineering. Marcel Dekker, Inc., 270 Madison Avenue, New York, New York, 1983. ISBN: 0-8247-7022-6.

[243] Herbert A. Simon. 'The Structure of Ill-structured Problems'. In Nigel Cross, editor, *Developments in Design Methodology*, chapter 2.4, pages 145 – 166. John Wiley : Chichester, 1984. Originally published in *Artifical Intelligence*, volume 4: 181-200, 1973.

[244] Herbert Alexander Simon. *The Sciences of the Artificial.* Karl Taylor Compton lectures, 1968. Cambridge : M.I.T. Press, 1969. pages 55 - 56.

[245] Chee-Kiong Soh and Ai-Kah Soh. 'A Knowledge-based Approach to the Design of Offshore Jacket Structures'. *Computer-Aided Engineering Journal*, 7(1):7 – 11, Feburary 1990. ISSN: 0263-9327.

[246] David A. Solomon. *Inside Windows NT.* Microsoft programming series. Microsoft Press, Redmond, Washington, second edition, March 1998. ISBN: 1-57231-677-2.

[247] Ian Sommerville. *Software Engineering.* Addison - Wesley, Workingham, England, third edition, 1989. ISBN: 0-201-17568-1.

[248] Ian Sommerville. *Software Engineering*, chapter 11: Object Oriented Design, pages 203 – 231. Addison - Wesley, Workingham, England, third edition, 1989. ISBN: 0-201-17568-1.

[249] Y.H. Song and C.S.V. Chou. 'Advanced Engineered-Conditioning Genetic Approach to Power Economic Dispatch'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 144(3):285 – 292, May 1997. ISSN: 1350-2360.

[250] Y.H. Song, H.B. Wan, and A.T. Johns. 'Knhonen neural network based approach to voltage weak buses / ares identification'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 144(3):340 – 344, May 1997. ISSN: 1350-2360.

[251] Y.H. Song, G.S. Wang, A.T. Johns, and P.Y. Wang. 'Distribution network reconfiguration for loss reduction using fuzzy controlled evolutionary programming'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 144(4):345 – 350, July 1997. ISSN: 1350-2360.

[252] Y.H. Song, G.S. Wang, P.Y. Wang, and A.T. Johns. 'Environmental / Economic Dispatch using Fuzzy Logic Controlled Genetic Algorithms'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 144(4):377 – 382, July 1997. ISSN: 1350-2360.

[253] William R. Spillers, editor. *Basic Questions of Design Theory.* North-Holland Publishing Company and American Elsevier Publishing Company, Inc., 1974. ISBN: 0-7204-2818-1 or 0-444-10739-8.

[254] Steven Stapleton and Robin Wallace. 'Reducing Mini-Hydro Generator Outages in Weak Electricity Networks'. In *Small Hydro '98 : 7th International Conference & Exhibition*, volume 1, pages 25 – 34, Astir Palace Hotel, Athens,

Greece, 16th - 18th November 1998. International Water Power & Dam Construction and Wilmington Business Publishing, Wilmington Business Publishing, Wilmington House, Church Hill, Wilmington, Dartford, Kent, DA2 7EF.

[255] Steven A. Stapleton and A. Robin Wallace. Construction of a Hierarchical Rule Base to Aid the Design of Embedded Generation Schemes. In *32nd Universities Power Engineering Conference (UPEC '97)*, volume 2, pages 1114 – 1117, Manchester, United Kingdom, 10th – 12th September 1997. Department of Electrical Engineering and Electronics, University of Manchester Institute of Science and Technology (UMIST). Volume 1: ISBN 0-9523165-3-6, Volume 2: ISBN 0-9523165-4-4, Two Volume Set: 0-9523165-5-2.

[256] Louis I. Steinberg. Design = Top Down Refinement Plus Constraint Propagation Plus What? In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 2, pages 498 – 502, Alexandria, Virginia, October 1987. Institute of Electrical and Electronic Engineers (IEEE). ISSN: 0000-0498.

[257] Mark G. Stewart. Probabilistic risk assessment of quality control and quality assurance measures in structural design. *Institution of Electrical and Electronics Engineers (IEEE) Transactions on Systems, Man and Cybernetics*, 21(5):1000 – 1007, September – October 1991. ISSN: 0018-9472.

[258] Bjarne Stroustrup. *The Design and Evolution of C++*. Addison Wesley Publishing Company, 1994. ISBN: 0-201-54330-3.

[259] Bjarne Stroustrup. *The C++ Programming Language*. Addison Wesley Longman, Inc., third edition, September 1997. ISBN: 0-201-88654-4.

[260] Visual Studio. ©Microsoft Corportation, 2000. Internet homepage: http://www.microsoft.com/.

[261] Nam P. Suh. *The Principles of Design*. Oxford Series on Advanced Manufacturing. Oxford University Press, 200 Madison Avenue, New York, NY, 1990. ISBN: 0-19-504345-6.

[262] Y. Sun, H. Jiang, and D. Wang. 'Fault synthetic recognition for an EHV transmission line using a group of neural networks with a time - space property'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 145(3):265 – 270, May 1998. ISSN: 1350-2360.

[263] Ivan E. Sutherland. *Sketchpad: A Man-Machine Graphical Communication System*. PhD thesis, Massachusetts Institute of Technology (MIT), 1962.

[264] Jim Tapper. The making of the European energy network. *Institution of Electrical Engineers (IEE) Review*, 46(3):9 – 13, May 2000.

[265] G.N. Taranto and D.M. Falc ao. Robust Decentralised Control Design using Genetic Algorithms in Power System Damping Control. *Institute of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 145(1):1 – 6, January 1998. ISSN: 1350-2360.

[266] Ernest R. Tello. *Object Oriented Programming for Artificial Intelligence: A guide to tools and system design*. Addison-Wesley Publishing Company, 1989. ISBN: 0-201-09228-X.

[267] John C. Thomas and John M. Carroll. 'The Psychological Study of Design'. In Nigel Cross, editor, *Developments in Design Methodology*, chapter 3.4, pages 221 – 235. John Wiley : Chichester, 1984.

[268] P.G. Thomas and J. Welsh. 'A REC's Experience of Embedded Generation within its Networks'. In *The Impact of Embedded Generation on Distribution Networks*, number 96/191 in Colloquium Proceedings, pages 5/1 – 5/15. Institution of Electrical Engineers, Savoy Place, London, UK, 15th October 1996.

[269] Hamid A. Toliyat, Javad Sadeh, and Reza Ghazi. Design of Augmented Fuzzy Logic Power System Stabilizers to Enhance Power Systems Stability. *The Institute of Electrical and Electronic Engineers (IEEE) Transactions on Energy Conversion*, 11(1):97 – 103, March 1996. ISSN: 0885-8969.

[270] C.T. Tse, K.W. Wang, C.Y. Chung, and K.M. Tsang. Parameter optimisation of robust power system stabilisers by probabilistic approach. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 147(2):69 – 75, March 2000. ISSN: 1350-2360.

[271] S.K. Tso, T.X. Zhu, Q.Y. Zeng, and K.L. Lo. 'Evaluation of load shedding to prevent dynamic voltage instability based on extended fuzzy reasoning'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution*, 144(2):81 – 86, March 1997. ISSN: 1350-2360.

[272] H.W. Turner and C. Turner. Fuses. In The Electricity Council, editor, *Power System Protection: Part 1 - Principles and Components*, chapter 5, pages 307 – 338. Peter Peregrinus Ltd, Stevenage, UK and New York, 1981. ISBN: 0-906048-47-8.

[273] John Twidell. 'Renewable Energy: Implementation and Benefits'. In *APSCOM - 93. 2nd International Conference on Advances in Power System Control, Operation and Management*, number 388 in IEE Conference Proceedings, pages 418 - 424, Hong Kong, 7th – 10th December 1993. Institution of Electrical Engineers, Savoy Place, London, UK. ISBN: 0-85296-569-9.

[274] Yasushi Umeda and Tetsuo Tomiyama. Functional Reasoning in Design. *The Institute of Electrical and Electronic Engineers (IEEE) Expert*, 12(2):42 – 48, March/April 1997. ISSN: 0885-9000.

[275] A. Robin Wallace. Electrical engineering - power systems. Third year undergraduate lecture notes, Department of Electronics and Electrical Engineering, University of Edinburgh, Kings Buildings, Mayfield Road, Edinburgh, EH9 3JL, UK. Telephone: 0131 650 1000, Fax: 0113 650 6554, 1995.

[276] Alexander Robert Swan Wallace. *Small Scale Hydro Power Generation*. PhD thesis, Department of Electronic and Electrical Engineering, Faculty of Science and Engineering, University of Edinburgh, Energy Systems Group, Department of Electronic and Electrical Engineering, The University of Edinburgh, Kings Buildings, Mayfield Road, Edinburgh, Scotland, UK, EH9 3JL, October 1990.

[277] Robin Wallace. Personal discussions, April - November 1996. Discussions held in a series of meetings with Dr. Robin Wallace in the Department of Electronics and Electrical Engineering, University of Edinburgh.

[278] D.A. Waterman. *A Guide to Expert Systems.* The Teknowledge Series in Knowledge Engineering. Addison - Wesley Publishing Company, 1986. ISBN: 0-201-08313-2.

[279] Ian D. Watson. 'An Introduction to Case - Based Reasoning'. In Ian D. Watson, editor, *Progress in Case - Based Reasoning, First United Kingdom Workshop,* number LNAI 1020 in Lecture Notes in Artificial Intelligence; subseries of Lecture Notes in Computer Science, pages 3 – 16, Salford, UK, 12th January 1995. AI-CBR and British Computer Society Specialist Group on Expert Systems, Springer - Verlag, Berlin. ISBN: 3-540-60654-8.

[280] Peter Watson. Personnel interview, 15th May 1997. Discussions held with Peter Watson from Scottish Power, Control Operations, Head Office, Cathcart Business Park, Spean Street, Glasgow in the Department of Electronics and Electrical Engineering, University of Edinburgh. Personnel present where Dr. D.E. Macpherson, Paul McCabe and Steven Stapleton.

[281] S. Websper, R.W. Dunn, R.K. Aggarwal, A.T. Johns, and A. Bennett. 'Feature extraction methods for neural network – based transmission line fault discrimination'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution,* 146(3):209 – 216, May 1999. ISSN: 1350-2352.

[282] Ann L. Winblad, Samuel D. Edwards, and David R. King. *Object - Oriented Software.* Addison - Wesley Publishing Company, Wokingham, England, 1990. ISBN: 0-201-50736-6.

[283] Patrick Henery Winston and Berthold Klaus Paul Horn. *LISP.* Addison - Wesley Publishing Company, third edition, October 1989. ISBN: 0-201-08319-1.

[284] Thies Wittig, editor. *ARCHON: An Architecture for Multi-agent Systems.* Ellis Horwood Series in Artificial Intelligence. Ellis Horwood, Market Cross House, Cooper Street, Chichester, England PO19 1EB, 1992. ISBN: 0-13-044462-6.

[285] Andrzej J. Wnuk. 'Some Topics on a Model for the Design Process'. In *Proceedings on Artificial Intelligence, Simulation and Planning in High Autonomy Systemms,* pages 149 – 159, Tucson, AZ, USA, 26th - 27th March 1990. Institution of Electrical and Electronics Engineers, Computer Society Press, Los Alamitos, CA, USA. ISBN: 0-8186-2043-9.

[286] K.P. Wong, A. Li, and M.Y. Law. 'Development of constrained - genetic - algorithm load - flow method'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution,* 144(2):91 – 99, March 1997. ISSN: 1350-2360.

[287] H.C. Wu and C.N. Lu. 'Automatic fuzzy model identification for short – term load forecast'. *Institution of Electrical Engineers (IEE) Proceedings: Generation, Transmission & Distribution,* 146(5):477 – 482, September 1999. ISSN: 1350-2352.

[288] E.-C Yeh, S.S. Venkata, and Z. Sumic. 'Improved Distribution System Planning using Computational Evolution'. *Institution of Electrical and Electronics Engineers (IEEE) Transactions on Power Systems,* 11(2):668 – 674, May 1996. ISSN: 0885-8950.

[289] Erh-Chun Yeh, Ying Sun, S.S. Venkata, and Z. Sumic. 'Design By Expectation : a framework for engineering design optimization'. In *Proceedings of the Sixth International Conference on Tools with Artificial Intelligence*, pages 105 – 111, New Orleans, LA, USA, 6th – 9th November 1994. Institution of Electrical and Electronics Engineers, Computer Society Press, Los Alamitos, CA, USA. ISBN: 0-8186-6785-0.

[290] D.J. Young, K.L. Lo, J.R. McDonald, R. Howard, and J. Rye. 'Development of a Practical Expert System for Alarm Processing'. *The Institute of Electrical Engineers (IEE) Proceedings C*, 139(5):437 – 447, 1992.

[291] Lotfi A. Zadeh. Fuzzy Sets. *Information and Control*, 8:338 – 353, 1965.

[292] Lotfi A. Zadeh. Making Computers Think Like People. *Institute of Electrical and Electronic Engineers (IEEE) Spectrum*, pages 26 – 32, August 1984. ISSN: 0018-9235.

[293] Ibrahim Zeid. *CAD/CAM Theory and Practice.* Computer Science Series. McGraw-Hill Inc., 1991. ISBN: 0-07-072857-7.