Design Models for Service-based Software Application

Maria Anjum

School of Engineering and Computing Sciences

Durham University

Doctor of Philosophy

2013

Declaration

The work in this thesis is based on research carried out in the Innovative Computing Group, the School of Engineering and Computing Sciences, University of Durham, UK. No part of this thesis has been submitted elsewhere for any other degree or qualification. All the work presented here is the sole work of the author unless referenced to the contrary in the text.

This research has been documented or is related, in part, within the publications listed below.

- Maria Anjum and David Budgen, A mapping study of the definitions used for service oriented architecture, in *16th International Conference on Evaluation Assessment in Software Engineering* (EASE) May 2012, IET Press & IEEE. Doi: 10.1049/ic.2012.0008
- Maria Anjum and David Budgen, Modelling the design for an SOA system to control a small scale energy zone, in *IEEE 36th Annual Computer Software and Applications Conference Workshops* (COMPSACW) July 2012, IEEE Computer Society. Doi: 10.1109/COMPSACW.2012.100.

Dedicated to

My Mother, Khalida Nasreen Anjum

and in memory of

My Father, Mahmood Ahmad Anjum

Acknowledgements

I would like to praise and thanks to God Almighty for all the success in this thesis in particular and in my life in general.

This research would not have been possible without the financial support of Lahore College for Women University, Pakistan under the scheme of Faculty Development Scholarship from Higher Education Commission Pakistan. Their generous support is highly appreciated.

I would like to express my deepest sense of Gratitude to my supervisor Professor David Budgen for his invaluable advice, continuous guidance and encouragement throughout the course of this thesis. I got opportunity to learn from him and get benefit from his experience.

My sincere gratitude to Professor Philip Taylor, Dr. Neal Wade, and Fawad Khateeb from Energy Group, Durham University. Without their guidance and technical support it would not be possible for me to understand this domain and construct a case in energy engineering.

I would like to express my gratitude to Professor Malcolm Munro from Innovating Computing Group, Guy Hutchinson from Energy Group and Jonathan Berry from Innovating Computing Group, Durham university for taking part in the review sessions. I am highly obliged that they participated as experts in reviews, and for their critical remarks and valuable suggestions.

My special thanks to Professor Thomas Green and Luke Church from Cambridge University for visiting us in Durham and sharing their knowledge on notations.

Thanks to Muhammad Abdallah Al-Tamimi and Khalid Yousif Muhammad from School of Engineering and Computing Sciences, Durham University for being great colleagues and friends.

I greatfully acknowledge the support of my colleagues and friends Dr. Fatima Chami from Chemistry Department, and Muhammad Aurangzeb Mughal from Anthropology Department, Durham University and to Syed Atif Mehdi from Technical University Kaiserslautern, Germany. I would also like to thank Dr. Mariam Rehman from Computer Science Department, Lahore College for Women University, Pakistan for being a wonderful friend.

Thanks to Ustinov College staff particularly Sarah Lee, Trevor Russell, Brian D Taylor and the porters.

Special thanks to my friends from Durham and to my international friends I came to know during my stay in Durham University.

My deepest thanks to my family for their encouragement, love and spiritual support during this time.

Finally, my heartfelt thanks to my mother for her love and constant support throughout my life. I reached to this point because of her.

Abstract

Context: The use of a Service Oriented Architecture (SOA) offers a new and distinct approach to creating software based applications (SBAs) around the idea of integrating distributed autonomous computing resources. A widely available realisation of an SOA exists in the form of web services. However, to date no standard techniques have emerged for developing SBAs. There is also a lack of consistency in describing the concept itself, and the published literature offers little evidence derived from the experience of developing 'real world examples.

Aims: The objective of the work described in this thesis was to conduct a series of studies to explore systematically the concept of what constitutes an SOA by using the published literature, to employ this to construct a proof of concept SOA design model based on a real world problem, and in doing so, to investigate how well existing design notations are able to support this architectural style.

Method: The research described in this thesis has been conducted in an evolutionary manner by employing a range of empirical methods. A mapping study was performed to investigate how the concept of SOA is interpreted by the research community. Based upon this model of SOA, a participant-observer case study was employed to construct an SOA design model and a use case model for an energy engineering application to demonstrate use for a real world problem. Finally, expert knowledge was employed for evaluation of the case study through the use of walkthroughs.

Results: From the mapping study we created an integrated model of what constitutes an SOA for the use with the case study. The case

study outcomes include a design for a renewable energy control system togather with codified experience of constructing and recording the SOA design model. The experience of employing the walkthrough method for evaluation, and the outcomes of the evaluation are also discussed.

Conclusion: From this research we conclude that the SOA research community needs to develop a clearer shared understanding and agreement on the model of what constitutes an SOA and the vocabulary used to describe the SOA concept. This will aid designers to communicate their mental models more effectively and will provide the semantics needed for devising the new notations that this study implies are needed for SBA design. Further, some lessons about SBA design have been derived from the case study experiences.

Contents

List of Figures

List of Tables

Glossary of Terms

ADMD — After Diversity Maximum Demand	Defines peak load for an average customer by calculating maximum demand, per customer as the number of customers connected to the network increases.
CBD — Component based Development	A software development technique that emphasises the use of components for the construction of software applications.
CF — Capacity Factor	Expresses the amount of electricity produced by an electricity generator as a percentage of the maximum theoretical production from that generator.
DFD — Data Flow Diagram	A graphical representation used to illustrate the process of data flow in a system, in terms of its inputs and outputs.
DNO — Distribution Network Operator	Responsible for technical operations of medium and low voltage networks that supply electricity to the customers through their distribution networks.
DSM — Demand Side Management	Used to plan, implement and monitor electricity utility activities that are designed to influence customer usage of electricity in a way that will produce desired changes in the utility load.
EBSE — Evidence Based Software Engineering	A process of identifying, understanding and evaluating findings from research and practice-based experience systematically and objectively gathering and assessing the available evidence.

ESCO — Energy Services Company	The company that govern and manage small scale energy zone (SSEZ) to supply locally generated electricity to their customers in industrial, commercial and domestic sectors.
ESU — Energy Storage Unit	A unit that consists of batteries used for power management.
Islanding	A situation where a distributed generator (DG) continues to maintain the network voltage and frequency to a location, within regulatory limits even after disconnection from the power utility.
SaaS — Software-as-a-Service	A software development technique that separates the possession and ownership of software from its use by presenting software using a service model.
SBA — Service Based Application	A software application constructed through the composition of software services available from the network or by third parties.
SBCS — Service Based Control System	A software system used to control the activities of the SSEZ.
SLR — Systematic Literature Review	A process of identifying, evaluating and interpreting available evidence about a particular topic in an unbiased and objective manner.
SOA — Service Oriented Architecture	An architectural style used to construct service based applications.
SOSE — Service Oriented Software Engineering	Emphasis on the life cycle of service based application development.
SSEG — Small scale Embedded Generator	Micro generation of electricity through a combination of generators designed to operate with a low voltage network.
SSEZ — Small Scale Energy Zone	A controllable low voltage distribution network (LVDN) that consists of a number of small scale embedded generators (SSEGs), distributed energy storage units (ESUs) and units of customer demand.

Chapter 1

Introduction

Service Oriented Architecture (SOA) has emerged in the last decade as considered a new paradigm for developing distributed software applications. The key aspects that differentiate this from previous paradigms are the loose coupling among computational resources, interoperability between heterogeneous applications, negotiation and ownership.

The research described in this thesis investigates some of the ways in which software applications can be designed and constructed around the concept of an SOA. Since, SOA is relatively new area compared to previous technologies (object oriented and component based development), the published literature on this topic is mainly contained within the last decade, and the concepts and the vocabulary that describe an SOA are still evolving. Likewise the techniques to design and develop SOA based applications have not had time to consolidate on some agreed practices.

To conduct research on an emerging area requires a clear model of the concept and needs to establish a chain of evidence that has been constructed in a systematic manner. For this reason, the research described in this thesis has been carried out as a sequence of empirical studies. In this chapter, the nature of an SOA and related concepts are explained briefly to provide a context for the research discussed in the thesis. The chapter also explains its relevance to the challenges and opportunities facing the development of a service based application (SBA).

The next section describes the context, section 1.2 provides the objectives of this research, and section 1.3 summarises the structure of the thesis.

1.1 Context

The emergence of software service technologies and of related concepts such as that of a Service Oriented Architecture (SOA), have provided a new and distinct approach to creating distributed systems around the idea of integrating distributed autonomous computing resources.

The software service model also known as Software-as-a-Service (SaaS) introduced by Pennine Research Group (Brereton et al., 1999) provided a long term vision of developing software using a service model. This forms a demand-led paradigm, whereby a requirement is fulfilled by the assembly of various services, as and when needed. The distinguishing characteristic of this concept is that it separates the possession and ownership of software from its use (Turner et al., 2003; Budgen et al., 2004). More recently, the SaaS model has also been described as a software delivery model by Laplante et al. (2008), where service is delivered on demand over the internet. This provides an opportunity for organisations to share resources in a constantly changing environment and to get paid through either micro- or macro-billing mechanism for providing these services. This facilitates the customers to get a desired service without employing their own resources, which also reduces the cost of software ownership.

1.1.1 Architectural Style and SOA

Shaw and Clements (1997) have defined an architectural style as a set of design rules that aid in identifying the type of components and connectors required to construct a system. It can make use of local or global constraints for composition. The subsystems take the form of components which are distinguished by the functionality they provide. The connectors are the type of interactions that occur among components. In this regard, SOA can be considered as an architectural style that describes components in the form of services, and where interaction among these services can take different forms (such as request and response messages or remote procedure calls).

The services that constitute an SOA are called atomic services. The atomic services provide elements of the required functionality and cannot be subdivided. These services range from performing a simple function such as a calculation,

to those encapsulating a complex business process. They have well-defined interfaces, are self-contained and are independent of the state or context of other services (Papazoglou and Heuvel, 2007). Laplante et al. (2008) have described SOA as a software construction model. This differentiates SOA from the SaaS model. As mentioned by Laplante et al. (2008), SaaS provides services for SOA to use and SOA helps to realise the concepts of SaaS.

The focus of the research described in this thesis is on SOA. Throughout in the thesis, any application constructed around the SOA concept is described as a service based application (SBA).

1.1.2 Component Based Development (CBD) and SOA

The SOA concept has partly evolved from component based development (CBD). However, it provides a more flexible approach towards distributed application development. The similarities between both technologies have become differences in the way they address the problem (Breivold and Larsson, 2007). Tables 1.1 and 1.2 provide a brief comparison of some key differences between CBD and SOA.

1.1.3 SOA as a New Paradigm

Reuse and abstraction are considered important concepts in software development. Development methodologies have encouraged reusability through emphasis upon modularity and on hiding the internal details, such as by making use of the concept of information hiding introduced by Parnas (1972). Abstraction can be used as a technique for making the essential features of a system visible and suppressing others. In this way, it helps with reducing the complexity of the problem (Loy, 1990). An SOA makes use of these concepts by encapsulating the functionality as a service and by providing details in the form of interfaces that are essential to invoke and use the service. This provides a more flexible approach for developing applications and introduces a new layer of abstraction to components. The way SOA based applications are developed can be considered a paradigm shift.

Table 1.1: A side by side view of CBD and SOA

	CBD	SOA
Process (assembly)	In CBD, a software application is developed by integrating components developed by third party, called off-the-shelf components (COTS). The components encapsulate their architecture and implementation details. The application also need 'glue code' to make these components work together (Garlan et al., 1995).	The construction of an SOA based application treats service publication, discovery, selection and composition as key elements. Composition is a type of 'glue code' however, has taken a form of process in SOA and has become an implicit feature.
Specifications (interface, assumptions etc.)	Component are different from each other in terms of their abstraction and complexity. Detailed specifications are required about component interface, behaviour, possible interactions with other components and about the configuration. Also, replacement of component needs details about functional and non-functional features of the component. Further, assumptions made about the application domain, underlying infrastructure and about individual components need to be explicitly specified. This can remove the potential conflicts among components (Garlan et al., 1995; Crnkovic and Larsson, 2000; Geisterfer and Ghosh, 2006).	In SOA, service description includes details of functional, and non-functional features. The interface description makes easy for a consumer to discover and select appropriate services from a pool of services.
Interoperability	The 'glue code' is required to make different components work together. Components are written in different languages and use different platforms, therefore, interoperability has become an issue (Garlan et al., 1995; Brereton and Budgen, 2000).	Interoperability is considered an implicit feature of SOA. The services in an SOA encapsulate the implementation details, and have invokable interfaces that make it platform, protocol and location independent.
Reusability	To develop a reusable component requires three to four times more resources than developing a component that serves a particular case (Szyperski et al., 2002). Further, a change to any component without proper planning requires extra effort.	The services composed in SOA based application may be provided from new services, legacy applications and components wrapped as a service.

Table 1.2: A side by side view of CBD and SOA

	CBD	SOA
Ownership and Choice (selection)	Components are developed by third party, therefore, replacement of old component with the new version makes the consumer to use the component from the same vendor. That limits the selection of a component and makes the choice vendor specific and platform dependent.	In SOA, service consumer has choice to select a different service provided by a different service provider, each time application is executed. This provides more choice to service provider and consumer when compared to CBD.
Versioning and Standardisation	New versions of components are released as new changes are made in the components. That raise the issue of compatibility between different versions of components. Therefore, it becomes consumers responsibility to track these changes and modify application accordingly. Further, it is difficult to standardise components (Crnkovic and Larsson, 2000).	In SOA, service providers keep the ownership of service and therefore responsible of changes made in the service. As services are accessed on demand, therefore, service consumer has choice to change service provider.
Support and Maintenance	For reusable components, support is required to: develop components for different platforms, development of different variants of components for different products; and development and maintenance of different versions of components for different product versions (Crnkovic and Larsson, 2000; Brereton and Budgen, 2000)	In SOA the change of service interface might be an issue if carried out without informing the consumer. As services make use of contracts, this binds the service provider to take the responsibility of any change made in the service.
Evolution	The components in the application are configured at build time by making use of tested and known components. As system evolves and new versions of components are made available there is usually no mechanism to detect and install new components (Crnkovic and Larsson, 2002).	In SOA application, service selection and composition are considered as a process and application make use of this to find and compose different services at runtime.
Cost , quality, and time to market	CBD was adopted due to the reduced cost of development, access to better resources in terms of quality of components and rapid development and deployment of applications. However, lack of reusability in terms of factors discussed above have made the community reluctant to adopt SOA concept. Therefore, CBD and SOA are somehow evolving in parallel.	Reusability of services provides these benefits but with greater flexibility and independence.

In the case of an SOA, a practical realisation of the concepts already exists widely in the form of web services. However, design practices suited to developing service based applications (SBA) are still evolving. Current efforts in SOA research are largely directed towards the construction of software development life cycles for service based applications. This is termed as service oriented software engineering (SOSE). In this thesis we have focused on an important aspect of SBA development, namely the logical and physical design. The reason for selecting design is the gap that exists in the current literature, where there is lack of models for an SOA design process as well as of evidence about any experiences of developing service based applications. Further, design is considered an important part of software development, needed to produce high quality software and to deal with the increasing complexity of software applications.

1.1.4 Software Design and SOA

Software development techniques have their own underlying philosophy and the design methods and notations developed for these techniques aid the designers with representing such system features. This means that designers need vocabulary and a set of supporting representational forms in order to communicate their design ideas. The absence of these, and the lack of consensus, can lead to possible misunderstandings about the concepts, or invites multiple interpretations (Wieringa, 1998). As noted by Shaw and Clements (1997), software designers extensively use descriptive forms to explain their design ideas, and in doing so they make use of a vocabulary that is informal, ambiguous and difficult to communicate to others. Therefore, it is desirable to establish a common vocabulary that represent shared understanding of the concepts and can be used to communicate architectural knowledge. Their work has contributed significantly to define the terminology of software architecture.

In a similar manner, the component community has developed some level of agreement on the definition of a software component, such as the widely used component definitions by Brown and Short (1997); Heineman and Councill (2001); Szyperski et al. (2002). A recent study by Boer and Farenhorst (2008) describes how definitions of architectural knowledge are employed by researchers, and sim-

ilar efforts are now required for the case of SOA. Definitions of SOA do exist in the literature, but the lack of common understanding has caused the research community to borrow terms from implementation technologies and often to use the terms in an ad hoc manner. In this thesis we have sought to explore the concept of SOA as described in the existing literature, and have used this to develop an integrated SOA model as discussed in Chapter 5.

Experimental studies on how designers work indicate that they usually construct 'mental models' of the intended system. During the process of development they may also mentally execute the model in order to observe its behaviour (Adelson and Soloway, 1985; Visser and Hoc, 1990). To record and communicate their mental models, designers make use of different representational forms (such as diagrams). These aid them in transferring their ideas in order to construct design models. These representations or notations provide syntax and semantics to communicate ideas and to help with refinement of a system design. They also make use of the well defined set of concepts and terminologies offered by that particular architectural method, for example, forms such as the class diagram are based on object oriented concepts.

To investigate the issue of SBA design in a systematic way, this study has identified SOA attributes from the published literature through a mapping study and constructed an integrated SOA model; constructed a case study to represent a significant 'real-world' problem; developed an SOA design model; and then evaluated the resulting design by employing expert knowledge with regard to both application domain and software design. The architecture of the thesis is shown in Figure 1.1.

1.2 Research Objectives

The concept of SOA is widely employed in the creation of service based applications (SBA). In the process of exploring design literature on SOA we became aware that the term is used rather vaguely and there are many different interpretations. We also identified that there is no evidence available about the experience of constructing SOA based application design. Further to this, the examples used in literature are apt to be constructed artificially and be narrow

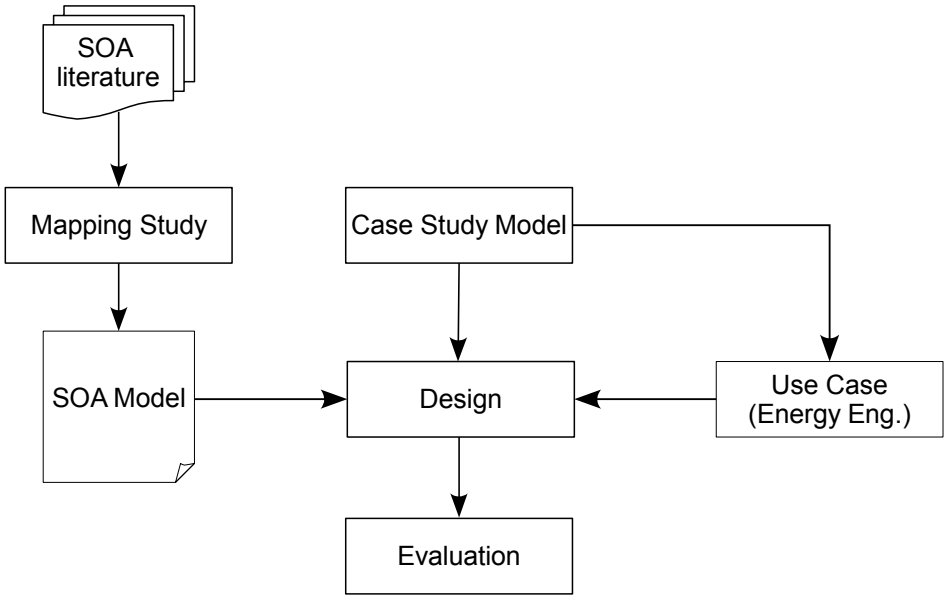


Figure 1.1: Thesis Structure

in their scope. As noted by Lane and Richardson (2011), the process models and development life-cycles available in the literature lack empirical validation and there is no proof of their applicability in real life scenarios. Even the process model used in service-oriented modelling and architecture (SOMA), which they found to be quite mature when compared to other models, lacks evidence about the claim that this model is based on the experience of developing a large number of service based projects. Also we found that there is no evidence that there are any standard practices available for designing applications using an SOA form. This is also observed by Oliveira et al. (2010) who note that there is no consensus about representing service-oriented architectures, and that informal ways are used to explain the model that could be interpreted in different ways.

These findings have encouraged us to fill some of the gaps by conducting research on SOA to systematically explore the concept, and by employing a real-world case study, construct an SOA design model.

The main objectives of this research are therefore:

- To identify the key characteristics of a Service Oriented Architecture (SOA) from the published literature in a systematic manner, and to investigate how

the term SOA is being interpreted by the research community.

- To conduct a case study based on a real-world problem. Within this, we have used a use case model to represent the operational features of the 'case' we have selected from the domain of energy engineering.
- To model SOA attributes through the use of abstract diagrammatical forms to help design the use case, and hence to determine the suitability of existing notations for designing such systems and to identify where new forms of representations are required.
- To evaluate the use case model and the SOA design model by employing expert review.

1.3 Thesis Structure

The thesis is organised as ten chapters. After this introduction chapter, there are two chapters (2 and 3) that contain background information on the research area under investigation.

Chapter 2 provides an overview of Component based development (CBD) and Service oriented architecture (SOA). It explains briefly the evolution and the emerging forms of both paradigms.

Chapter 3 provides details about software design, a survey of on-going research on the SOA based software life cycle and the issue of notations in software design.

Chapter 4 then describes the research methods used to conduct this research. It presents the rationale for choosing these forms and how they are employed.

Chapter 5 explains the process of performing the mapping study. The chapter describes the different phases of the mapping study including data extraction, synthesis, results, and findings.

Chapter 6 is about the case study based upon a small scale energy zone (SSEZ). The case study provides an introduction about SSEZ, the need for a case study approach in this research and the way it has been used. The chapter also describes the use case developed as part of the case study. The description of the use case covers the requirements for the SSEZ control system in detail,

including SSEZ network configurations, network operational features, and SSEZ network data that will be accessed from inside and outside of the SSEZ control system.

Chapter 7 describes the SOA design model constructed as a part of case study. The details from initial design model to its representation in different diagrammatical forms are provided.

Chapter 8 explains the evaluation process through the use of walkthroughs, including how the reviews were conducted, the lessons learned from them, and the outcomes of the evaluation.

Chapter 9 discusses the research conducted in this thesis, and considers the possible threats to validity.

Chapter 10 concludes the thesis, summarises the contributions, and indicates a number of potential areas for conducting further work.

Chapter 2

Literature Review - SOA Models

2.1 Introduction

Reuse is an important concept in software development. This has encouraged software community to develop new techniques for development. To achieve reuse, modularity was introduced which is based on the well known concept of 'separation of concern'. Structured approach achieves this by dividing a problem into a set of functions. However, with the introduction of 'information hiding', the new concept of developing software was evolved in the form of object oriented (OO) paradigm. This changed the way of thinking of software design and development. Instead of thinking a system as a set of functions; the 'Object' is considered the main element to be focused on for developing the system. Thinking about object means considering real world entities that have a set of attributes and associated functions. The object oriented paradigm introduced a new level of abstraction and was considered a paradigm shift.

Reuse has different levels from the reuse of source code to the development of reusable software. Since software industry is evolving rapidly and softwares are becoming more complex, it has become important to reuse the available functionality to develop new softwares. For this reason the concept of 'component' was introduced. In component based development (CBD) a system is developed by integrating existing (or third party) components instead of building it from scratch. CBD requires a new way of software development and reuse a large 'chunk' of

system functionality. The evolved form of CBD is service oriented architecture (SOA) that adds another level of abstraction and provides more flexibility than CBD.

In this chapter we briefly describe CBD, how this is evolved, its limitations that led the software community towards SOA. Further we describe SOA, its forms and its realisation through current technologies.

2.2 Component based Development (CBD)

The objective of component based development (CBD) is to build application systems through the assembly of ready to use software components (i.e. components-off-the shelf). By producing application systems with pre-constructed software pieces, CBD promises the benefits of accelerated software development, reduced costs, higher reusability, and greater flexibility (Szyperski et al., 2002). The technologies that have become standard for component development, integration, and deployment include Enterprise Java Beans (EJB) by Sun Microsystems Inc., Common Object Request Broker Architecture (CORBA) by the Object Management Group and Microsoft Corporation products that include Component Object Model (COM), Distributed Component Object Model (DCOM) and .NET.

Sharp and Ryan (2010) have described component development and component based system development as two separate processes. And the development phases for the development of a component and a system based on a set of components are different. In CBD, component is a building block and like OO development, it is important to define what a component is. For this reason, different definitions are being evolved within the CBD community. That shows that CBD community has developed a shared understanding of the concept and terminology of component. The definitions that are widely used by CBD community are explained below.

- Brown (1997) contributed an early definition by describing component as "an independently deliverable set of reusable services".
- Heineman and Councill (2001) categorised a component as being "a software element that conforms to a component model and can be independently

deployed and composed without modification according to a composition standard".

- A widely used definition by (Szyperski et al., 2002) describes a component as "A software component is a binary unit of composition with contractually specified interfaces and explicit context dependencies only. Context dependencies are specified by stating the required interfaces and the acceptable execution platform. A software component can be deployed independently and is subject to composition by third parties".

- Hopkins (2000) has provided a definition that describes a component as "a physical packaging of executable software with a well defined and published interface".

- Another definition in use is from (Waguespack and Schiano, 2004) that defines a component as "an artifact of systems development, manufactured explicitly for the purpose of being used in the construction of multiple systems by multiple development groups".

In the definition offered by Brown two main elements of components are discussed: one is reusable services and the other one is independent delivery of components which means components cannot be developed with embedded dependencies on one another, but that a component might have generic dependencies that could be satisfied by different providers (Brereton and Budgen, 2000). In Heineman and Councill, the emphasis is on a component model, independent delivery as mentioned earlier by Brown and standardisation. Szyperski's definition further stresses the need for well defined interfaces and explicitly defines the context specific dependencies for the purpose of component composition. Hopkins and Waguespack definitions more or less emphasise the same concepts discussed in earlier definitions.

Apart from the efforts to define what a component is, there has also been discussion on the type and nature of the components. Brown et.al, (1998) have divided components into two categories: abstract and off-the-shelf. CBD with abstract components requires new methods and tools for design, development

and integration of components. This represents a 'white box' form of components where the design focus is on the collaboration of interfaces to understand system architecture and enables reuse and replacement of implementations that conform to the interface specifications. For systems developed with off-the-shelf components, there is a need for new techniques of application assembly. Such components can be considered as black box as there is limited or no access to their internal design, and functionality.

Carney and Long (2000) has categorised components on two axes: origin and modification. Component origin means how the components were developed at first place and represents the commercial point of view. The five possibilities about component origin are listed below.

- Independent commercial item
- Custom version of a commercial item
- Component produced to order under a specific contract
- Existing component obtained from external sources (for example, a reuse repository)
- Component produced in-house

The modification axis is about the type of 'glue code' that is required for components composition and represents the technical aspect. This includes following options:

- Very little or no modification
- Simple parametrisation
- Necessary tailoring or customisation
- Internal revision to accommodate special platform requirements
- Extensive functional recoding and reworking

Carney and Long (2000) article on COTS not only provides clarity about the acronyms that exist for components, but also their categorisation using *origin* and *modification* helps with understanding the complexity of the component based system design and integration mechanism.

The issues of CBD are discussed in detail by (Brereton and Budgen, 2000). Their focus is more on development issues of component systems without going into the debate of architectural forms of such systems and their implementation. The issues are discussed through a framework that is based on five functional areas (software product, software process, business and people or skill) which are mapped to the viewpoints of three key stakeholders (component providers, component integrators and integrated system customers) that they identified for such systems. The research emphasises the need for further work in the direction of component evaluation, maintenance, integration of technical and commercial factors and composition rules to build component based systems. In addition to this, according to Brereton and Budgen (2000) while moving towards CBD based development, the software development process needs to integrate the new concepts of selection, evaluation, and integration into its development process.

CBD gained popularity because of the concept of reusable components. Further elements that contributed towards the acceptability and adoption of this development style were low cost, less development time, efficient development, enhanced product reliability, reduced maintenance, portability, flexibility and easy access to best resources. CBD research community continues to put effort into finding solutions that will reduce the risks associated with this type of development. The challenges that are still open in CBD are:

Reuse: Reuse principles place high demands on reusable components. The components must be sufficiently general to cover the different aspects of their use. At the same time they must be concrete and simple enough to serve a particular requirement in an efficient way. However, developing a reusable component requires three to four times more resources than developing a component (Szyperski et al., 2002). This is because the requirements of the components are usually incomplete, not well understood (Sommerville and Kotonya, 1998) and bring additional levels of complexity.

Versioning: In a component based system, components are configured at design time with known and tested versions of components. However, problems arise when new versions of a component come. In such case, a system based on components needs to know about the new versions and how to replace previous one with new one. A component can be replaced easily if it is compatible with its previous version. Compatibility issues are relatively simple when changes introduced in the products are in the nature of maintenance and improvement only. Using appropriate test plans, including regression tests, functional compatibility can be tested to a reasonable extent. More complicated problems occur when new changes introduced in a reusable component eliminate the compatibility (Vitharana, 2003). In such a case, additional software, that can manage both versions, are required to be written (Crnkovic and Larsson, 2002).

Component Selection: For all technologies, component definition is usually limited to syntactic specification of interfaces. There is no support for semantic specification of software components (Teiniker et al., 2005). Significant progress in the advancement of the component-based paradigm will probably not occur without successfully addressing selection and reuse. The issues related to replacement of software components (considered as primary driver for CBSE) are also closely related to the issues involved in selection and reuse (Geisterfer and Ghosh, 2006).

Component Assembly: There is no support for nested component composition where components can be composed to construct subsystems and further embedded into another component (Teiniker et al., 2005). Architectural mismatch discussed by Garlan et al. (1995) is another important factor that makes it difficult to write 'glue code' for components.

Portability: All current server component technologies (EJB, COM/DCOM, CORBA etc.) have a strong interdependency between their component model and either the used platform, programming language or middleware technology (Teiniker et al., 2005).

Development Environment: When developing reusable components, several

dimensions of the development process need to be considered that include support for development of components on different platforms; support for development of different variants of components for different products; support for development and maintenance of different versions of components for different product versions. To address these types of problems, development environment support is essential (Crnkovic and Larsson, 2000).

Standardisation: There are a number of models proposed for component standardisation but not enough work has been done in this direction. There is the possibility of interoperation between EJB, COM/DCOM/.NET, and CORBA infrastructures, however, there is no significant progress for the emergence of a united component infrastructure (Duan and Yuan, 2007; Mahmood et al., 2007).

Visual Modelling and design: Despite limited success with extending the Unified Modelling Language (UML) for component modelling, visual modelling of CBD with different component infrastructures remains one of the most difficult and challenging subject in CBD (Duan and Yuan, 2007). The issue of component based system design is still open (Szyperski et al., 2002) and OO techniques are still mainly used to develop components (Vitharana et al., 2003).

Although research on the above issues is still going on, the bottlenecks in CBD have encouraged the software community to devise new development techniques, such as the concept of software services.

2.3 Service Oriented Architecture (SOA)

SOA is an emerging paradigm that is being widely advocated for software development. It can exploit the power of the internet and of grid systems to provide an alternative and distributed approach to the traditional way of designing, developing and implementing monolithic software applications. As an architectural paradigm, it integrates the resources provided by multiple applications that may be owned by others, and that may use quite disparate technologies and platforms.

Compared to traditional distributed object-oriented architectures, SOA is better able to integrate heterogeneous systems, and is potentially more adaptable in a changing environment. It uses software service technology as a fundamental framework for the design and development of applications.

SOA has attracted the attention of industry and academia because of its features of loose-coupling, reusability, and interoperability (Lewis and Smith, 2008). Indeed, one of its attractions for commercial purposes is the potential to re-deploy legacy systems within a 'service wrapper'. The concept of SOA has been implemented in various technologies and applied in several domains. It is considered as representing a paradigm shift from object oriented (OO) and distributed computing (DC) (Cotroneo et al., 2004). It can also be considered as having roots in some earlier forms of distributed technology such as DCOM / CORBA (Chen et al., 2006).

In a service oriented architecture, a system is decomposed into smaller parts (components/modules) that are able to provide the required functionality by employing a number of services.

2.3.1 Software Service Model

In software service model, a system is built through the use of autonomous, distributed computation elements which are self contained and can be combined on demand (Budgen et al., 2004; Prinsloo et al., 2006). This concept is also known as software-as-a-service (SaaS). Brereton et al., (1999) introduced the concept of SaaS in which *services* are composed out of smaller ones (and so on recursively), procured and paid for on demand. The idea they presented was to deliver and consume software as services with a long term vision for software evolution. The concept was elaborated and used in *Web service conceptual architecture* by IBM. IBM and Microsoft focused primarily on technical solutions where as SaaS research was using a market led approach (Zhu et al., 2004).

In the SaaS model, services have the following three properties;

- Being used rather than owned, with no significant processing needing to be performed by a user.

- Conforming to a document-style interface, making a service independent of programming language constructs for interconnection and data exchange:
- Being stateless, in the sense of not preserving end user knowledge across different episodes of use.

SaaS focuses on separating the possession and ownership of software from its use. In this way it would open up new markets, both for relatively small-scale specialist-services providers and for larger organisations that provide more general services (Turner et al., 2003).

Currently, SaaS is generally linked to cloud computing (Cusumano, 2010) and research work on SaaS applications is more focused on single instance multi-tenant models (MTAs) which are best known in terms of cloud computing (Schroeter et al., 2012). Multi-tenancy refers to a principle where a single instance of the software runs on a server, serving multiple client organisations (tenants). With MTA, a software application is designed to virtually partition its data and configuration, so that each client works with a customised virtual application instance. Although all tenants share the same software, they feel like they are the sole user of the software(Tsai et al., 2010).

Both the SOA and SaaS, concepts are based up on demand-led composition of services which we regard is the most important aspect of this new paradigm. According to Laplante (2008) despite their significant differences, SaaS and SOA are closely related architectural models for large-scale information systems. Using SaaS, a vendor can deliver a software system as a service. Using SOA enables the published service to be discovered and adopted as a service component to construct new software systems, which can also be published and delivered as new services. In other words, the two models complement each other: SaaS helps to offer components for SOA to use, and SOA helps to quickly realize SaaS.

2.3.2 SOA Model

The software service model provides specific functionality across a network and an SOA is a structure that combines individual elements of functionality to provide an overall system. Each service is essentially autonomous and so the process of

composition can involve short-term or long-term negotiations for service provision between the 'supplier'and the 'consumer'. Further to this, in SOA, services can be made available by different service providers (meaning that they can have different ownership), and their functionality is published through well-defined interfaces in such a way that they can be discovered and composed to provide functionality to other services, systems or end users.

A general service model used to explain SOA is shown in Figure 2.1. The interaction model consists of three-elements that are: service providers, service requesters, and registries that are used for service discovery (Baresi et al., 2003). The interactions among these entities are through the *publish*, *find* and *bind* operations.

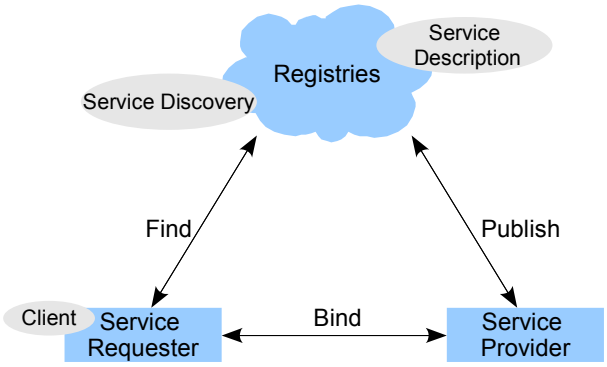


Figure 2.1: Service Oriented Architecture Model (Huhns and Singh, 2005)

The service providers publish their service descriptions in service repositories or registries. Service users search these repositories to find their required services. Once a user finds a particular service, it is possible for them to directly negotiate with the provider to agree on terms and then invoke the service (Schuschel and Weske, 2004). The three basic functions that must therefore be supported in an SOA are:

1. Describe and publish services
2. Discover a service
3. Negotiate with a service provider and consume the service.

The development of SOA based applications involve different activities that are different from tradition development models. Lewis and Smith (2008) have provided a comparison between traditional and service oriented system that is shown in Table 2.1.

Table 2.1: Differences between Traditional and Service-oriented Systems (Lewis and Smith, 2008)

Traditional Systems	Service-oriented Systems
Tight coupling between system components	Loose coupling between service consumers and services
Semantics shared explicitly at design time	Semantics shared without much communication between developers of consumers and services
Known set of users and usage patterns	Potentially unknown set of users and usage patterns
System components owned by the same organisation	Systems components potentially owned by multiple organisations

The most significant advantage of SOA when compared to CBD is that it embodies loose coupling among services. Once a service is discovered, the user is not required to have any detailed knowledge of its location, implementation, implementation language, or execution platform. The only concerns of the user are to renegotiate (or agree) terms of use and to determine how the service can be invoked through a service interface (Chen et al., 2006). The CBD challenges discussed in section 2.2 are discussed here with respect to SOA in Table 2.2.

2.3.3 Service Composition Process

Service composition is a process in which a business process is constructed by integrating a set of atomic services. An atomic service is one that provides some functionality (that can be a simple formula calculation, business function, a search process etc.) and that can not be divided any further into smaller services, meaning that it is the lowest level of service provision. The composition process is similar to the traditional work flow model. Activity, control flow and data flow are the basic elements of composition process. Activities correspond to certain operations carried out by atomic services. Control flows describe the dependency

Table 2.2: CBD Challanges and SOA

CBD Challenges	SOA
Reuse	SOA offers loose coupling among services that provides flexibility and makes it easy to reuse the functionality.
Versioning	There is no need to keep information about different versions as services remain under the ownership of service providers. However, if service interface changes without prior information then consumer has to manage the change. But there is no compatibility issues.
Component selection	In terms of specifications, in SOA, service interfaces are well defined and there is no issue of service replacement as we have in CBD.
Component assembly	In SOA, higher level services can be composed of lower services.
Development environment	SOA is platform independent and provides interoperability. However, tools to develop SOA based applications are still evolving.
Standardisation	It is important for CBD but for SOA it is not critical.
Visual modelling and design	SOA is also facing this challenge.

relations among the activities, that is, the time sequence that the basic services to be carried out. Data flows describe the data transformation between the activities (Qing-Ming et al., 2009).

The process of service composition can be divided into different activities which are discussed below.

- Service description and publication: provides the basis for matching user needs to available services; and describes functionality, interfaces, non-functional characteristics and constraints, as well as describing the parameters within which both the provider and the user are willing to negotiate.

- Service discovery: is used to locate appropriate services, resulting in a list of candidate services and their providers. Service discovery employs matching techniques to select services by comparing their descriptions against user constraints. Service discovery enables suitable services to be located based on functional requirements, non-functional requirements or both. A

service discovery approach is either syntactic-based (where descriptions are represented as a set of strings and "string matching" is used) or semantic-based (where ontological relationships are used to perform mappings between terms of user requests and service descriptions) (Gooneratne and Tari, 2008).

- Service selection: involves identifying, appropriate services from the list of discovered services. If more than one service is providing the same functionality, then QoS and user preferences can be used to select the most suitable service to fulfil the requirements. In the case of a dynamic environment, service selection and re-selection (in case of failure) will need to be performed at runtime.

- Service negotiation: is a process of interaction between the user and one or more service providers, with the aim of agreeing the terms and conditions for the supply of a service.

- Service integration and execution: At this stage, a complete plan is generated that describes how to call atomic services to obtain the overall behaviour of a business function (Agarwal et al., 2008). A process is designed that defines the order of service interaction and execution. By combining and linking services, the process shows the control and data flow from one service to another.

  The process can be constructed statically, or (semi/fully) dynamically. In static composition the process model is created manually and service binding is done at design time. Workflow-based methods (Rao and Su, 2004) come in this category. Workflow based service composition explicitly defines the control and data flows among services. Standards such as BPEL, BPEL4WS, XLANG, WSFL, BPML and WSCI can be applied to define the workflow models. However, these models are based on static composition and do not support dynamic adjustment of the workflow to create a better fit to the requirement (Zhao and Tong, 2007). By adding semantics for service interfaces, usually with the help of OWL-S, more information a can be added to a service description (Fujii and Suda, 2005). This approach is

called semi-dynamic composition. In dynamic composition, process model is created automatically and service binding is done at runtime. For this, AI and agent technology are used.

Many factors are involved at different stages of service composition process. For example, in *service publication* an important consideration can be that of representing service features and defining the way that services can be listed in the registry, so they can be found during the selection process. The selection criteria for candidate services may involve QoS features and user preferences; and an execution process that defines the order in which services are to be executed (either through runtime planning or predefined workflows) according to a business process, by considering data and functional dependencies among services. There are further issues (like time of execution, cost, fault-tolerance, self configuration etc.) that are associated with the composition process, and that depend upon the techniques used.

In SOA based systems, service providers, service brokers and service consumers are the main stakeholders (Gu and Lago, 2007) and thus they have a direct impact on the selection of service composition technique and implementation technology.

Service consumers (either end-user, other system or service, or service provider itself) can affect the way services are published and selected, and at the same time, their agreement or disagreement on the end result in terms of non-functional features (accuracy, time, and cost) can effect the composition process. In addition, the negotiations and contracts with service providers (a company, another system, website etc.) can effect the selection of a particular service and its availability at run time.

In some situations, service level agreements (SLAs) are used by service providers and consumers to provide a contract for a particular level of service quality. These SLAs have to be defined, monitored and enforced so that service functionality and data can be predictably and contractually delivered between the providers and users. For both providers and users it is important to understand and map business drivers to quality attribute requirements and to clearly articulate these in the SLAs. Equally important is to have processes to monitor the quality of

the service provided and to define policies that deal with situations where the contracted level of service is not met (O'Brien Lero et al., 2007).

However, the limitations of current solutions and tools can effect the decision making process and restrict service providers and consumers to making compromise solutions.

2.3.4 Technical Perspective

In terms of technology used to realise SOA, web services are widely considered to be the most suitable technology for implementing an SOA (Harrison and Taylor, 2005; Papazoglou and Heuvel, 2007). According to Baligand and Monfort (2004), although web services are not the only way to model the service paradigm, they are one of the major technologies that can provide both the interoperability and loose coupling required for an SOA. However, SOA based applications can be implemented using other technologies such as message-oriented middleware (MOM) (e.g. IBM Websphere MQ), publish-subscribe technologies (e.g. Java Messaging Service (JMS)), and Common Object Request Broker Architecture (CORBA) (Lewis and Smith, 2008).

Two core themes exist to implement web services: XML based technologies and more recently Representational State Transfer (REST).

- Web services exploit XML and internet technologies to integrate applications. Three key XML-based standards have been defined to support Web service deployment: Simple Object Access Protocol (SOAP), Web Services Description Language (WSDL), and Universal Description, Discovery and Integration (UDDI). WSDL provides a description framework for web services and is primarily aimed at service invocation. UDDI offers a registry service that allows advertisement and discovery of Web services. SOAP provides a standard way to structure messages that can be carried over a variety of transport protocols with HTTP being the most frequently used one (Korotkiy and Top, 2006).

- Representational State Transfer (REST) is another way to provide web services. REST was originally introduced as an architectural style for building

large-scale distributed hypermedia systems. The REST architectural style is based on four principles: identification of a resource through its URI; uniform interface using a fixed set of four operations create, read, update, delete; self-descriptive messages in a variety of formats (e.g., HTML, XML, plain text, PDF, JPEG, etc.) and interaction through stateless messages (Pautasso et al., 2008; Lewis and Smith, 2008)

2.3.5 Business Perspective

To run and manage SOA applications, an organisation needs an SOA infrastructure. An SOA infrastructure consists of several elements that support the main aspects of SOA including security, governance, management, orchestration and resourcing (e.g. virtualisation) (O'Brien et al., 2008). In general, it is recognised that SOA adoption can provide benefits for business agility, adaptability, legacy leverage, and integration with business partners. Given these potential results, an important criterion for making business decisions concerns the amount of investment that is required for SOA adoption and the projected pay-off over a certain period of time. Current efforts have focused on individual case studies and there have not been any rigorous analyses that can be generalised (Kontogiannis et al., 2008).

SOA governance focuses on the smooth adoption and successful operation of an SOA as the enterprise architecture in a company. By providing guidelines, responsibilities, and reference processes, it ensures its integrity and adaptability to business and administration processes (Niemann et al., 2009). It involves the techniques and processes to model policy, risk, and trust, and to ensure that a service acts on requests that comply with claims required by policies (Kontogiannis et al., 2008). A number of organisations such as IBM, AgilePath and Software AG have developed sophisticated models of SOA governance. These models focus mostly on its relationship to corporate enterprise architecture, use of registries, SOA life cycle management, defining and monitoring service level agreements, and defining and analysing metrics on policy enforcement, effectiveness of services and use of services. Most efforts to define and implement governance are still vendor-driven and are focused upon by those governance aspects that can be

automated by their tools (Kontogiannis et al., 2008).

SOA governance includes policies, procedures, roles and responsibilities for design-time governance and runtime governance. Design time governance ensures that services meet the business objectives that they are meant to serve. Runtime governance ensures that services are being provided and consumed in a consistent fashion (Lewis and Smith, 2008).

2.4 Summary

Paradigm shifts introduce change in the way softwares are developed. This change includes the way of thinking towards software development and requires new methods and tools to realise these changes. From functions to objects and from components to services, reuse and information hiding are the main concepts that are causing these paradigm shifts. Service Oriented Architecture (SOA) is evolved from component based development (CBD). For CBD, the idea was to promote reusability, in order to reduce development time and to use 'off the shelf' components to deliver software. However, the challenges in CBD have encouraged researchers to find forms that are flexible enough to handle all these issues, and the concept of the *software service* is essentially one of these.

The concept of component and service share a common development model where (component/service) development and assembly are performed by different actors and can take place at different locations. Since services focus on other aspects of development such as dynamic discovery and negotiation that are generally not explicit considerations for components (Cervantes and Hall, 2005).

Developing a service based system is therefore a different task from that of using previous software development technologies, especially those developed for the object-oriented paradigm. In OO there is usually no issue of ownership, hence no contracts and negotiations.

As noted by Sharp and Ryan (2010) for components, the development of services and the SOA based system are two different things. Both need new techniques and tools for design and implementation.

Chapter 3

Literature Review - SOA Design

3.1 Introduction

In this chapter we explain software design in general and techniques in particular. From literature on software design, we identify that design makes use of two important concepts: abstraction and modularity. Abstraction facilitates to reduce system complexity and modularity allows reusing system functionality. Further, modularity is considered an important feature for achieving reusability. Although choosing the right level of abstraction is non-trivial and involves complex tradeoffs (Wagner and Deissenboeck, 2008). The better a designer can predict what is likely to change in future; the easier it will be to change a module at later time without effecting other modules. The accepted criteria for making design decisions about modularity are derived from the concepts of separating computation from representation, preferring composition over inheritance and reducing coupling (Van der Hoek and Lopez, 2011).

Design theories usually form up around "separation of concerns" mentioned by Dijkstra (1976) and the concept of information hiding defined by Parnas (1972). The focus is to reduce dependency between modules, to hide the complexity and the internal functionality from others. This is particularly import today when software systems are large and more complex.

These concepts have made the software community to make use of different techniques to develop systems such as structured, object oriented, component

and more recently service based application development. These techniques differ from each other because of the concepts we mentioned earlier and also due to the approaches they used to solve the problem. Loy (1990) has mentioned these approaches as a different way of thinking about a problem. In structured development a problem is divided into a set of functions and subroutines and programs are developed to realise this. He named it a 'functional paradigm' also known as top down approach of problem solving. However, in case of object oriented development, the problem is analysed in terms of objects and classes. Further it introduces new level of modularity and provides concepts such as inheritance, and encapsulation. Therefore, the way system is developed using OO technique is different. This is called a bottom up approach.

Apart from knowing about development techniques, software design activity requires a set of tools and notations (text, tables, and diagrams). These aid the designer to represent system features and to communicate his ideas that exist in the form of mental models. That is why, designers make use of sketches and box and line diagrams to express the models constructed in their minds. Software design studies discuss this in terms of design thinking where focus is on the cognitive aspects, how mental models and mental simulations are created (Guindon and Curtis, 1988; Kim and Lerch, 1992). Further the analysis of design activities and design process through observation of expert and novice designers in real situations (Adelson and Soloway, 1985; Guindon, 1990*b*; Visser and Hoc, 1990; Reeves et al., 1995; Pohthong and Budgen, 2001; Petre, 2009).

While designing a software system, a number of factors contribute that include knowledge of the application domain, of software architecture, design methods, experience, and the support for tools and notations to express design solution and design concepts (Guindon, 1990*b*).

In this chapter we will briefly explain software design strategies, design methods such as structured, object oriented and the use of notations in software design largely in terms of diagrammatical forms.

3.2 Software Design Strategies

The theory on software design strategies has largely been derived from observational studies. Visser and Hoc (1990) has divided design strategies into different categories such as top-down and bottom-up, breadth-first and depth-first, and opportunistic. These strategies are discussed below.

- Top-down and bottom-up Strategy: The top-down approach begins from most abstract level down to the lowest and concrete level. This strategy is strictly used only when problem is familiar and a solution is similar to a previous one (Visser and Hoc, 1990). In other words, this strategy is suitable for well-structured problems where designer already knows the correct decomposition (Guindon, 1990*b*). In the bottom-up strategy the problem is approached by concentrating on small chunks of system elements and then they are integrated to construct larger elements (Mayrhauser and Vans, 1995).

- Breadth-first Strategy: The breadth-first strategy is used in combinations with a top-down strategy. The solution takes the form of a tree and at each level; information is maintained at equal level of detail (Visser and Hoc, 1990). Further, it ensures that the information about current state of design at one level of abstraction will be available to the next iteration (Anderson, 1981). Adelson and Soloway (1985) named it as balanced development.

- Depth-first Strategy: Instead of keeping all branches at the same level of details, in depth first strategy, some branches are constructed more in detail and others are handled afterwards (Visser and Hoc, 1990).

- Opportunistic Strategy: In this case, a mixed approach is used based on the previous strategies, on designer's experience and knowledge of design and application domain. Further, in opportunistic design strategy, the decision at given level of abstraction may influence the subsequent decisions at higher or lower level of abstractions (Guindon, 1990*a*).

Design is a creative process, involves decision making, deals with uncertainty and is constrained by a number of factors. The design progresses non- linearly

and involves the exploration of both breadth and depth of the problem. The known design problems are tend to be explored in opportunistic fashion as opposed to entirely novel design problem, which tend to involve the exploration of alternatives (Van der Hoek and Lopez, 2011).

Design strategies adopted by designers vary from domain to domain and from one development technique to another. The studies of software design practiced by designers reveal useful strategies that designers make use of and the characteristics essential to design process. They make use of provisionality and juxtaposition to explore alternatives and maintain awareness about options. For this, they intentionally change paradigms, formalism and representations to change the perspective. Further they avoid tools that impose restrictions and restrict them in expressing their ideas. However, they need tools that support conceptual design and provide conceptual design visualisations (Petre, 2009) .

3.3 Software Design Methods

The software design methods such as structured, object oriented, and services are explained in this section.

3.3.1 OO and Structured Design

New technologies arise the need of new methodologies for software development. Moving from structured development towards object oriented (OO), brought many changes in the way softwares were analysed, designed and developed. The survey like computer survey paper by Wieringa (1998) on the comparison of both methodologies: structured and OO, provides some insight how the need for new concept representation was addressed through new methodologies and the way structured methodologies were revised to fill the gap.

The debate on either previous methodologies are sufficient to address new needs is always there in software community. In their comparison on OO and structured methodologies, Fichman and Kemerer (1992) referred to Yourdon's(1987) categorisation of OO methodologists as revolutionaries and synthesists, where the first group believed that OO was a radical change and needed new methodologies

and the second group considered it a set of principles that could be embedded in existing methodologies. Further to this discussion, Fichman et al. (1992) quoted both Booch as well as Coad and Yourdon's stances on OO:

" Let there be no doubt that object-oriented design is fundamentally different from traditional structured design approaches: it requires a different way of thinking about decomposition, and it produces software architectures that are largely outside the realm of the structured design culture (Booch)."

" We have no doubt that one could arrive at the same results [as Coad and Yourdons OOA methodology produces] using different methods; but it has also been our experience that the thinking process, the discovery process, and the communication between user and analyst are fundamentally different with OOA than with structured analysis (Coad and Yourdon)."

The comparison of structured and OO methodologies by Fichman et al., (1992) is in two parts. One deals with the comparison of six (three conventional and three OO) analysis methodologies, and other with five (two conventional and three OO) design methodologies. The comparison uses 11 modelling dimensions. The design methodologies used in the study include:

- Yourdon and Constantine structured design
- Martin information engineering design
- Wasserman et al. object-oriented
- Booch object-oriented design, and
- Wirfs-Brock et al., responsibility driven design.

Fichman et al., concluded that object oriented design (OOD) is a radical change from both process and data oriented methodologies. The dimensions that need to be addressed by OO community and that are not supported in structured design are the detailed definition of classes and inheritance, class and object relationships, encapsulated operations and message protocols. Further to their analysis, the method of decomposition of modules in both methodologies is different. In structured analysis the view is function-oriented and modules such

as programs, sub routines and functions only contain procedural code. Whereas in OO, object that bundles methods and data, is a primary unit of modularity.

Iivari (1995) compared six OO analysis methods on three modelling perspectives: structural, functional and behavioural. The analytical framework is referred in Table 3.1.

Table 3.1: The analytical framework

	individual	object community
Structure	object, object class/type, Attribute	Relationship, Inheritance, Composition/aggregation, Subsystem
Function	Method/operation/service	
Behaviour	State transition	message/request

The study identified the strength and weaknesses of these methods along with the identification of the problem areas. The study found that the methods were similar with respect to structural modelling but quite different in their functional and behavioural modelling. Table 3.2 provides a brief summary of the findings.

The study also concludes that the OO methods are weak in guidelines to partition the system into subsystems. It argues that the mechanisms for modelling desired functional and behavioural capabilities at the level of the whole system are essential for OO.

(Wieringa, 1998) described system properties in terms of functions, communications and behaviour. These properties were used to conduct a survey of structured and OO software specification methods. The techniques were classified using the specification of external interaction and internal decomposition. The external interaction techniques where then further subdivided into functions, communications and behaviour. The study stresses on the need of simplicity in diagram techniques and the use of formal semantics to define them. The properties used to analyse specification techniques were:

- Functional specification techniques
- Behaviour specification techniques

Table 3.2: Strengths and weaknesses of OOA methods (Iivari, 1995)

Method	Strength	Weakness and problems
Coad and Yourdon	Conceptual simplicity A streamlined process	Little attention to the functionality and behaviour of objects. The concept of 'subject' as a substitute for 'subsystem'.
Jacobson et al.,	'Use cases' as functional and behavioural abstractions, Object categories	Little attention to the functionality and behaviour of individual objects. An unconventional way of modelling relationships Insufficient instructions. (e.g. integration of use cases)
Martin and Odell	A rich set of concepts for behaviour modelling	Modelling attributes Poor instructions (especially on how to proceed from behavioural modelling to structural modelling)
Rumbaugh et al.,	Balanced attention to the three perspectives (structure, function, behaviour) Object modelling Generally good instructions	The perspectives are not clearly integrated. Little attention to 'subsystems'. No enforcement of encapsulation in the sense of data hiding.
Shlaer and Melior	Balanced attention to the three perspectives 'Domain' analysis and 'sub-system' analysis	Information model based on the relational model. No support for complex objects. No enforcement of encapsulation in the sense of data hiding. Somewhat insufficient instructions (e.g. domain integration and object lifecycle derivation)
Wirfs-Brock et al.,	Early attention to the behaviour of object communities (collaborations) The concept of 'subsystem'	No explicit concepts for attributes, relationships and complex objects. Unclear boundary between OOA and OOD. Neglect of the internal behaviour of objects. Complexity of the process.

- Communication specification technique
- Decomposition specification technique

3.3.2 Service Oriented Software Engineering (SOSE)

In the literature, the software development life cycle (SDLC) is discussed for both structured and OO development, the same effort is now needed for the development of SOA based applications which is termed as Service Oriented Software Engineering (SOSE).

SOSE (Tsai, 2005) is a term introduced to bring the new features of SOA, such as identifying, discovering and composing services, into traditional software engineering activities (like coding, testing and deployment) (Gu and Lago, 2011). The basic engineering principles remain the same, but the way that they are applied are different in SBA development. Specifically, in SBA, most engineering tasks require to be performed at runtime in a collaborative manner. Because systems are composed at runtime using existing services, many engineering tasks need to be performed without complete information, which makes SOSE different from traditional software engineering (Tsai, 2005). In the literature, different SOSE methodologies have been proposed, for which the focus is on different aspects of SBA development. Some of these techniques are listed below:

- SOSE methodology (Karhunen et al., 2005) aims at developing methods and tools to improve quality and profitability of software development. The framework is based on

  1. business, service-oriented and component-based development features;
  2. it focuses that the first activity in developing SBA applications should be the creation of business case to justify project implementation;
  3. creation of a design model by focusing on communication and integration of service components.

  By combining component-based and service-oriented development, the SOSE component model provides three level of granularities: system level component (SLC), business service component (BSC), and component level.

- (Erradi et al., 2006) have developed Service Oriented Architecture Framework (SOAF), an architecture-centric framework. The proposed framework is business process-centric and is comprised of a set of structured activities grouped in five phases (information elicitation, service identification, service definition, service realisation and road map & planning). It incorporates a range of techniques and guidelines for systematically identifying services, deciding service granularity and modelling services while integrating existing operational/legacy systems.

- The method developed by (Papazoglou and Heuvel, 2006) for service-oriented analysis and design is based on the rational unified process (RUP), component-based development (CBD), and business process modelling. It concentrates on the levels of the web services development life cycle. It follows an iterative and incremental approach that consists of eight phases including planning, analysis and design (A&D), construction and testing, provisioning, deployment, execution and monitoring. The methodology stresses reliance on reference models, and considers several service realisation scenarios (including green field development, outsourcing and legacy wrapping).

- Gu and Lago (2007) have proposed a stakeholder-driven service life cycle model for SOA. The model represents the activities associated with stakeholders and the interaction among them. The study compares different SOA life cycles (2 academic and 6 vendor-specific) from the stakeholder's point of view. In Gu's model, life cycle activities are divided into design time, run time and change time. Design time refers to the life cycle of a service before it is available for use. During the runtime stage, services are put into production and the implementations start to work. The change time stage comes after runtime. It focuses on the life cycle of a service when adjustments have to be made when business requirements change. The activities are then associated with stake holders (service provider, application provider (service consumer) and service broker) and service life cycle stages.

- Chang (2007) has proposed a service oriented analysis and design (SOAD) methodology for adaptable services. The process consists of six phases

(Defining Target Services, Defining Unit Services, Planning Service Components Acquisition, Acquiring Service Components, and Composing Services.) and the result of each phase refers to one or more deliverables. The criteria for designing SOAD methodology involves:

  - Modelling the service variability among different clients and different contexts
  - Modeling the mismatch between published services and expected services
  - Designing adaptation mechanisms into service components
  - Enabling dynamic composition of services

  For Service Variability, three types of variation points are considered in service design (Workflow, Service Composition, and Logic). For service mismatch, three types identified are: interface mismatch, functional mismatch, and non-functional mismatch.

- SOMA by Arsanjani et. al.,(2008) is a software development life-cycle methodology invented and initially developed in IBM for designing and building SBA solutions. It consists of seven phases (Business modelling and transformation, Identification, Specification, Realisation, Implementation build/assembly, Deployment, monitoring, and management, Solution management) and provides support for the two main aspects of SOA governance: design-time and runtime governance.

- Offermann and Bub(2009) have proposed SOAM, that consists of six phases: company analysis (which covers business process aspects), service operation discovery, legacy system analysis, consolidation, service design and process preparation. Existing notations and models (UML, BPMN etc.) are used for modelling in SOAM.

Gu Lago (2011), have compared the SOSE methodologies both from vendors (SOAD, SOMA, SOUP by IBM, OASIS model, SO from CBDI) and academia,

on the bases of their general characteristics and those that are specific to service-orientation (service provision, service consumption etc.). Their evaluation framework is based on the characteristics identified through feature analysis. The focus of this study is not on what is practised, rather, its aim is to gain insight into the common features they share and the specific features they individually hold. Also they aim to differentiate the methodologies that are truly service-oriented from those that deal little with service aspects. However, it did not provide any priority to any methodology.

The study found that many service life cycle activities are not well supported. Mostly the design and analysis phases are covered and there is not sufficient detail available about construction, delivery and management phases. The development roles and responsibilities are not properly addressed. Further, the support for service consumption from both consumer and provider side is not fully available.

Finally study has proposed an evaluation framework for SOSE methodologies to facilitate organisations that want to adopt SOA.

3.4 Notations and Diagrammatical Representations

Designers make use of 'mental imagery' in constructing on abstract solution to the problem which can be externalised. The externalisation of images is used to share ideas and to communicate design decisions about the proposed solution. These images are discussed among designers to assess its adequacy in terms of how it solves the problem and what insight it offers about the particular issues. By doing this a shared set of semantics is developed among the team. Therefore, design process can be considered as a 'dialogue' between designers and artefacts, and among designers themselves. To aid this dialogue, designers make use of sketches (notations) that aid them in transition of 'mental image' to 'external representations' (Petre, 2009). Also these notations support them to uncover missing information and to ensure completeness of the problem (Guindon, 1990*b*).

Design notations plays an important role in producing the design. They express the design solution and are vehicle for developing the design solutions.

Software design activities are apt to involve different forms of notations. They range from informal conventions that are established on-the-fly (such as sketches or box and line diagrams) by a group of designers engaged in a design exercise to precise formalisms that are standards for the field. Two primary concerns in the formulation of notations are expressiveness and usability. Expressiveness concerns what aspects of a design can be captured in the notation; usability concerns the fluidity with which designers can work with the notation. Though both factors are equally important, the primary driving force behind the development of most new notations has been expressiveness adding modelling capabilities, often for a particular analysis purpose (Taylor and der Hoek, 2007).

Further, notations have both a syntax (structure) and semantics (associated meanings), and these need to be expressed correctly to ensure that the notation meets its purpose (Wieringa, 1998). For this reason notations need to have the quality that they could be easily produced and help the designers to explore their ideas about design and communicate those ideas to others (Budgen, 2003). In this regard, Green and Blackwell (1998) have proposed a framework called *cognitive dimensions*. The focus of which is upon notations (information representation) and by doing this they have provided a set of discussion tool for evaluating quality concepts.

Notation design in itself is a science that needs a proper theory for design and evaluation. Designing cognitively effective visual notations can, therefore, be seen as a problem of optimising them for processing by the human mind, in the same way that software systems are optimised for particular hardware (Moody, 2009).

In software engineering, notations exist in multiple visual forms such as data flow diagram (DFD) and ER modelling. They exist in multiple visual forms: DFD exists in two semantically equivalent forms: the De Marco style, consisting of circular "bubbles" and curved arrows and the Gane and Sarson, consisting of rounded rectangles and right-angled lines. ER modelling also exists in a variety of visual dialects. Despite the fact that these notations have been used in practice for over 30 years, there is still no consensus on which is best. Also why SE visual notations look so similar to one another and change so little over time. Without sound principles for evaluating and comparison of visual notations, there is no

reliable way to resolve such debates. For visual notation design to progress from a craft to a design discipline (a self-conscious design culture), there is a need to define explicit principles for evaluating, comparing and constructing visual notations (Moody, 2009).

The UML provides a detailed set of notations. However, while design activity is in progress, many details of such notations are usually ignored and simple and basic notations are used to express the concept. UML modelling philosophy is based on OO concepts and considered a de facto standard for OO modelling. Thus the notations semantics and syntax both are developed to support the design of OO systems.

Budgen et. al., (2011) conducted a survey on UML notations in order to determine the extent to which the forms and characteristics of the UML have been studied empirically. They found that apart from class diagram where its forms have been compared with other notations, not much evidence exists about the use of other UML notations. They also identified that there is a lack of evidence about the adaptation of UML notations in the field as compared to laboratory experiments.

In case of CBD, a component diagram is introduced in the set of UML notations. However, how far these extensions are effective for the design of CBD systems has not yet been evaluated.

3.5 Summary

Software design is an essential element of software development. In software design, two features abstraction, and modularity are considered important because they aid reusability. In this chapter, we have explained different design strategies used by expert and novice designers, reported through observational studies. Further, an overview of various design methods used for software development is presented. Finally, notations and diagrammatical representations are discussed as part of the software design activity.

Chapter 4

Research Method

4.1 Introduction

The chapter describes the research process employed in this thesis. The research strategy adopted is a multi-method one, in which different research methods that are appropriate to each of the sequence of the research questions are employed. As noted by Wood et al. (1999), a multi-method approach is used to investigate a phenomenon by employing a combination of empirical research methods, with the intention that the strength of the different methods complement each other. It is considered that this approach potentially provides benefits in terms of more robust conclusions, development and investigation of research hypothesis in an evolutionary manner, and increases the understanding of research results. A research study such as the one described here is not a single, discrete event, rather a process that proceeds through a number of phases that pose different tasks and problems for the researcher. In this way, particular research methods tend to be more useful in relation to some phases than others, hence combining them has a beneficial effect. Even where methods do perform similar functions, combining a range of approaches may well yield a better result (Mingers, 2001). The use of a multi-method approach in the area of information systems (IS) and more recently in software engineering is discussed by Wood et al. (1999); Mingers (2001, 2003); Petter and Gallivan (2004); Mandić et al. (2009).

The overarching research process developed for this thesis is shown in Figure

4.1. The process begins by posing a research question about the attributes of an SOA. To answer this question a *mapping study* technique which is a form of systematic literature review (SLR), has been employed.

- **Objective:** The objective was to collect evidence from existing literature about the way that SOA terminology was used by the research community.
- **Outcome:** The outcome of this study was an SOA model that we constructed through thematic analysis of SOA literature. The study also identified the issues related to SOA based application development. The issues that we considered for further research include: the need for a real world case instead of using 'toy' examples; and the construction of an SOA design model. We selected design because SOA realisations already exist predominantly in the form of web services and are discussed widely in the SOA literature. However, there is no standard design technique available for SOA-based application development. These factors motivated us to adopt a case study approach in order to pursue design issues in greater depth.

The case study is a research method used *to understand a contemporary phenomena in its real setting* (Yin, 2008). Here, The case study is used to address a real-world phenomena in an SOA context.

- **Objective:** To develop a use case from the energy engineering domain that describes the control system for a small scale energy zone (SSEZ). The use case provides an operational scenario to construct an SOA design model, and in doing this existing design notations have been used to describe the different features of the SSEZ control system.
- **Outcome:** The outcome of the case study is the use case model and the SOA design model for the SSEZ.

The outcomes of the case study have been evaluated by employing expert reviews (i.e. walkthroughs). The reason for using this are both the interdisciplinary nature of this research and also the unavailability of similar studies for comparison. Therefore, it was felt appropriate to involve experts from both domains (energy engineering and computer science) to evaluate the case study outcomes.

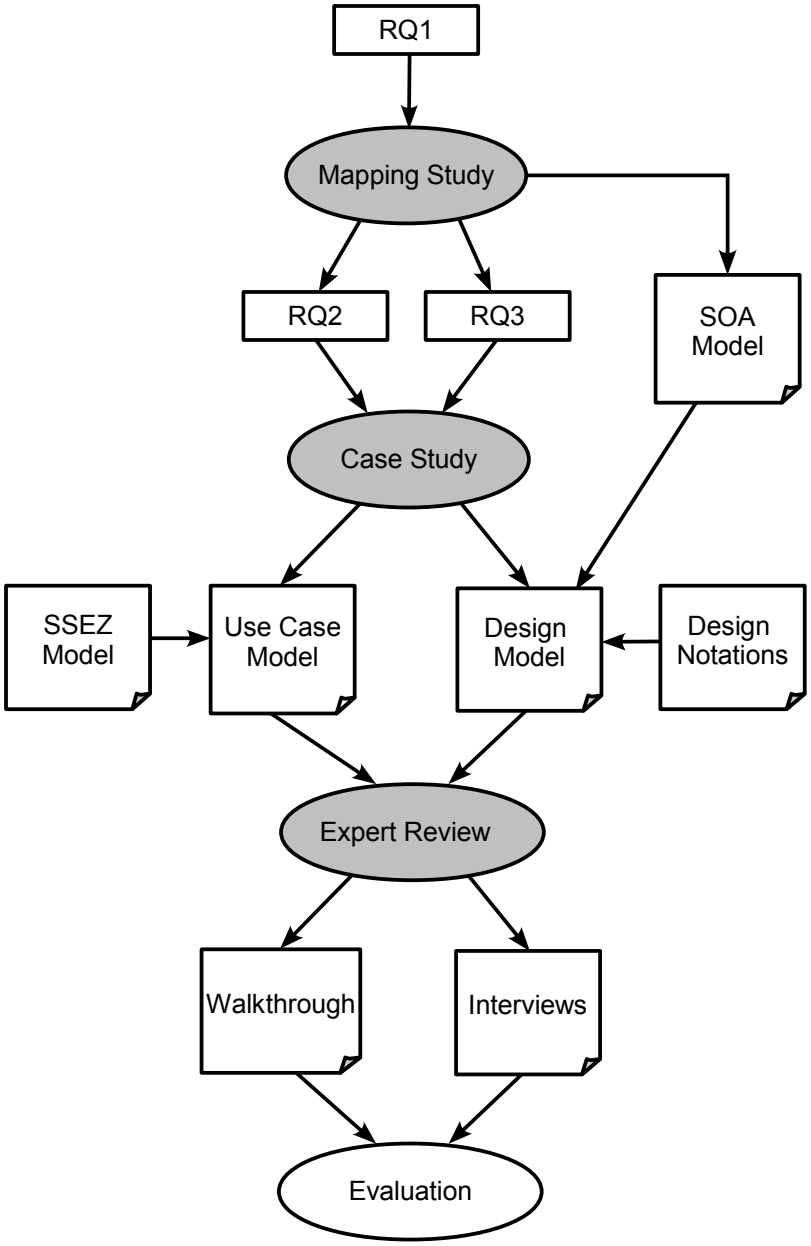


Figure 4.1: Research Process for this thesis

- **Objective:** Identify gaps in the use case description and SOA design model by employing expert knowledge from both domains. Also, to identify the effectiveness of using a walkthrough for evaluation in an academic context.
- **Outcomes:** The outcomes include the lessons learned from the walkthrough experience and the issues identified related to use case description and the SOA design model.

The evaluation process explains the activities carried out as part of the walkthrough and the results of these activities.

The form of each research method adopted is discussed briefly in the following sections. Fuller details are then provided in the later chapters.

4.2 The Mapping Study

Evidence based software engineering (EBSE) places emphasis upon adopting a 'systematic' approach of collecting evidence from research as a mechanism to be used by researchers and practitioners to find best evidence about the area of research (Dybå et al., 2005). In EBSE, systematic reviews (SRs) are a key tool for creating evidence based on synthesising data from individual studies. These reviews use explicit and rigorous methods to identify, critically appraise and synthesise relevant studies on a particular topic. Further they could be used to identify areas where the available evidence is insufficient and further studies are required (Dybå et al., 2007).

A systematic review, or as used in EBSE, the term systematic literature review (SLR), is defined by (Kitchenham and Charters, 2007) as a way " to identify, evaluate and interpret all available research relevant to a particular research question, or topic area, or phenomenon of interest in an objective and unbiased way".

The aim of an SLR is to ensure that the literature review is objective, unbiased, rigorous and repeatable. SLRs are used to answer specific research questions, however there are situations where the topic under investigation is relatively new and little or no evidence is available in the literature. In such situations a broader research question is developed. To handle such situations, a mapping

study or scoping review is used (Petticrew and Roberts, 2006; Kitchenham and Charters, 2007).

A mapping study is "a form of SLR that is used to identify gaps in the set of primary studies, to determine where new or better primary studies are required, and also to find clusters where there may be scope for more complete SLRs to be undertaken"(Kitchenham et al., 2011). Such a study addresses a broader topic than an SLR, and is designed to provide an initial indication of the size and location of the literature relating to a particular topic. It provides a comprehensive review of the topic and establishes how a particular term is used in what literature, by whom, and for what purpose (Cruzes and Dybå, 2011). The stages of a mapping study are generally similar to those of a SLR, although the research question itself is likely to be much broader and the searching may be less rigorous.

Motivation: The concept of SOA has evolved in the past decade and the literature available on this topic is almost entirely published after 2000. Hence to explore the concept, a mapping study is employed to collect evidence from literature. As explained in (Kitchenham et al., 2011), the goal of a mapping study is the classification and thematic analysis of literature, therefore, we considered it appropriate to conduct a mapping study to examine the concept of an SOA.

Research Question (RQ 1): The research question we choose for the mapping study was therefore: ***"What are the key characteristics of a Service Oriented Architecture?".***

Process: The structure and the procedure of conducting mapping study is discussed in detail in Chapter 5.

Result: The result of this study is an SOA model that emerged through the use of a thematic analysis and the synthesis of the SOA literature. This model has provided the terminology for describing SOA attributes and related concepts in the rest of the thesis.

4.3 The Case Study

A case study is an empirical research method that can be used to investigate a certain phenomenon in depth. Runeson and Höst (2009) noted that case studies that have appeared in the software engineering (SE) literature have addressed topics ranging from well defined and thoroughly performed studies to 'toy' examples. This may be because empirical research in SE has a strong focus on experimental forms of research. Further, the use of the case study as a research method is relatively new in SE when compared to information systems (IS) research (Runeson and Höst, 2009).

There are three types of case study, depending upon the research perspective, namely positivist, critical and interpretive. In software engineering (SE), case studies tend to use a positivist perspective and therefore, Yin's (Yin, 2008) definition is widely used, where this describes a case study as:

"an empirical inquiry that investigates a contemporary phenomenon in depth and within its real-life context especially when the boundaries between phenomenon and context are not clearly evident."

The key characteristics that make case study research appropriate for SE are:

- "it is of flexible type, coping with the complex and dynamic characteristics of real world phenomena, like software engineering,
- its conclusions are based on a clear chain of evidence, whether qualitative or quantitative, collected from multiple sources in a planned and consistent manner, and
- it adds to existing knowledge by being based on previously established theory, if such exist, or by building theory" (Runeson and Höst, 2009; Runeson et al., 2012).

Case study research is considered appropriate for SE because SE is a multidisciplinary field and involves areas where case studies are normally conducted such as 'field' studies. Also research in SE is usually aimed at investigating how development, operation and maintenance is carried out by software engineers and other stakeholders under different conditions (Runeson and Höst, 2009).

Motivation: The objective of employing a case study is to investigate a real world problem in detail and use this to model an SOA design. During the process of conducting the mapping study, it was observed that the cases discussed in the literature are artificially constructed and are narrow in scope. Also SOA applications are largely discussed from an implementation point of view. Further we found no studies that discussed the design of SOA based applications independent of technology, or the need for new notations. Therefore, to fill this gap, the resources within the School have been utilised to construct a case study.

Research Question (RQ 2 & RQ 3): The research question we have sought to answer through the use of a case study is: ***"Can the characteristics of an SSEZ control system be successfully modelled through the construction of a use case model?"***

The other two questions associated with the case study are: ***"How can SOA attributes be modelled using abstract diagrammatical forms?"*** and ***"How can such abstract models be developed?"***

Process: The case study process is shown in Figure 4.2. The case study domain is that of energy engineering, and the 'case' (or unit of analysis) is a *Small Scale Energy Zone* (SSEZ) control system. The case study design is categorised as a single-case design. The rationale for using single-case is the complexity and wide scope of the selected problem. The other reason was access to resources. Also, this was considered an opportunity to understand the phenomena in depth when resources are available within school and so could be used as a 'representative' case for SOA modelling. The operational model of the SSEZ control system has been explained by constructing a 'use case'. A protocol for the case study as suggested in (Brereton et al., 2008) was produced and is available in Appendix B. An SOA design model was also developed as part of the case study. This has made use of existing notations. For the evaluation of the use case and design, an expert review technique (walkthroughs) has been used.

Result: The outcomes of the case study include the use case model discussed in

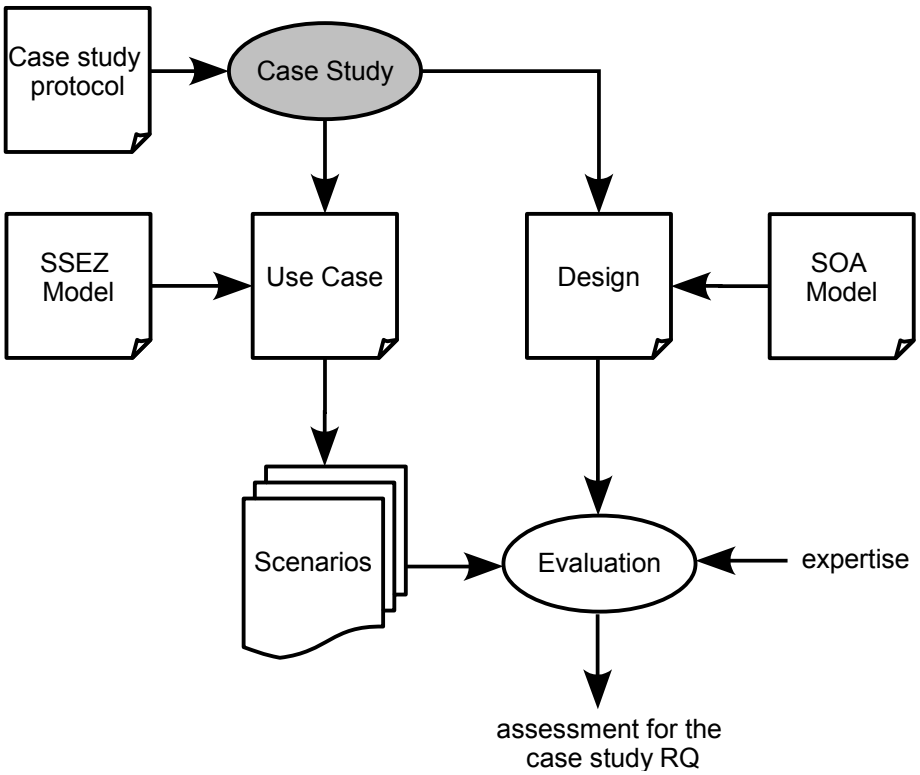


Figure 4.2: Case study process

Chapter 6, and the SOA design model discussed in Chapter 7. The evaluation process that investigates the outcomes of the case study is described in Chapter 8

4.4 Expert Review / Walkthrough

Reviews are commonly used in software engineering to evaluate the quality of software work products (code, design, or requirement specification etc.). The purpose of conducting a review is to identify defects in the work product so that problems created in one phase will not invalidate the next. Reviews are performed at different stages of development and can have a significant impact on the cost, quality, and development time of the software (Kemerer and Paulk, 2009).

In the literature, different review procedures exist, with the choice of form to use depending upon the nature of the problem. For example, inspections (Fagan,

1976) are used for code reviews, and make extensive use of checklists. However, structured walkthroughs can be used to verify both the overall approach and the outcomes (Weinberg and Freedman, 1984).

Motivation: The research discussed in this thesis is interdisciplinary and exploratory. Evaluating the use case and the SOA design model requires knowledge about energy engineering, and software design (more specifically about SOA). Therefore, a qualitative approach is used for evaluation, in the form of a walkthrough. As explained in (Budgen, 2003), a walkthrough is a useful technique for assessing the structural and behavioural aspects of a design. Through proper planning and organisation, it can bring together the people with expertise related to the domain and those with technical knowledge, in order to make realistic projections about the behaviour of the design.

Research Question: The overarching research question for walkthrough is: ***"Are the design, and the notations used, appropriate for the construction of an SOA model for the SSEZ control system?"***

Process: To conduct the walkthrough, a protocol was developed, as reported in Appendix F. The two walkthrough sessions were each followed by interviews with the participants. A cyclic approach of plan-act-reflect was adopted for this as used in action research (Oates, 2005). Experts from both domains (energy engineering, and computer sciences) were involved.

Result: The walkthrough identified gaps in the design. The data collected through the interviews with the walkthrough team addressed the issues related to the walkthrough process. As we mentioned earlier, the walkthrough can be used for academic purpose, therefore, lessons learned from this experience are also discussed. The walkthrough technique is used as part of the case study evaluation, and therefore, discussed in Chapter 8.

4.5 Summary

The research discussed in the thesis is exploratory and interdisciplinary. Therefore, a multi-method research approach as discussed in the beginning of this chapter has been used. The research methods employed include a mapping study, a case study and a series of walkthroughs. The research process consists of identification of SOA characteristics; modelling the characteristics of an SSEZ control system through a use case; construction of an SOA design and evaluation of the use case and design by means of expert reviews.

Chapter 5

The Mapping Study

5.1 Introduction

Service Oriented Architecture (SOA) is a paradigm that has emerged and evolved over the past decade, and hence the research community has published widely about SOA. As with every new paradigm, the concepts and terminologies that define this architecture are still evolving. The lack of agreement on common terms, a lack of awareness that other terms exists and a lack of an agreed model of what SOA is has led to potentially inconsistent use of terms. Hence individual researchers have interpreted the term in different ways, depending upon their needs and purposes. So, while the term SOA is widely used, interpretations of its meaning appear to vary according to who uses it and what the perceived benefits of an SOA are (Harrison and Taylor, 2005).

To design an SOA based application, designers need an architectural model that provides a relatively established set of common concepts, together with notations that embody a shared understanding of how the semantics will be interpreted for the eventual implementation. Therefore, to find out systematically how consistently the term SOA is used in literature, and what characteristics such a system should possess, we have carried out a mapping study that sought to answer the question *"What are the key characteristics of an SOA?"*. In this chapter the process of conducting a mapping study is explained and the outcomes are discussed in detail.

5.2 The form of a Mapping Study

A mapping study is a form of SLR (Kitchenham and Charters, 2007) that is designed to provide a systematic and unbiased overview of a research area, to establish if evidence exists on a topic and to provide an indication of the quantity of the evidence. The early stages of a mapping study are generally very similar to those of a systematic literature review, although the research question itself is likely to be much broader, in order to address the wider scope of such a study adequately. The three stages involved in conducting a mapping study, as defined in (Budgen et al., 2008), are:

1. identification of primary studies that may contain relevant research results *(searching)*;
2. selection of the appropriate primary studies from the results of (a), after further examination *(inclusion/ exclusion)*;
3. where appropriate, performing a quality assessment of the selected studies *(bias / validity)*.

5.2.1 Identification of Relevant Studies

Mapping studies and SLRs commonly use the PICO model to break down their research question and organise their searching process (Petticrew and Roberts, 2006). For this study, we interpreted this as follows:

Population of interest Papers that explicitly identified how they interpreted the term SOA.

Intervention Whether the definition used was explicit, or by reference.

Comparison Since we were using a mapping study to identify definitions rather than the more conventional use to find empirical studies, this element has no direct interpretation.

Outcomes These were the set of characteristics included in the definitions adopted for a given paper.

Based on this, we then performed a search to identify relevant papers.

5.2.1.1 Search String:

A set of three search strings were used to search relevant studies. The software engineering guidelines suggest that identifying suitable search strings might need some element of prototyping (Kitchenham and Charters, 2007), and as at times the initial choice of string was not present in the title of papers, or was only used in the keywords in abbreviated form, we gradually adjusted the strings to include these situations and so find the maximum number of papers.

Due to the popularity of service based systems we also identified many related terms in the search results such as service oriented programming, service oriented computing, service orientation etc. The following strings were the ones that we eventually used.

- "SOA"
- "Service oriented architecture"
- "Service-oriented architecture"

The interfaces of the electronic libraries such as those provided by IEEE, ACM and Science Direct are organised using different forms for specifying the details of a search, and so these strings were mapped on to the interface for each source in a manner that could bring the relevant results (details in Appendix A).

5.2.1.2 Selection of Time Period:

We considered it appropriate to start our search from the year 2000 since this was when the first standard for SOAP became available. The search was conducted during 2010 and so the complete time period used for our search was 2000-2009.

5.2.1.3 Choice of Electronic Databases:

The electronic electronic databases used were: ACM, IEEE Xplore and Science Direct. These cover major publishers of conference proceedings, since much of the literature on SOA is still being published in this form. Experience from other mapping studies that we have undertaken suggests that restricting the search to

this set of electronic databases will access the bulk of the relevant literature for an emerging topic. This could then use snowballing if appropriate.

5.2.2 Selection of Primary Studies

The selection process was performed in four steps.The decisions about inclusion and exclusion at each step were made according to the following criteria:

- Papers should be published in journals or conferences.
- They should be written in English.
- Papers should not come under the category of abstract, workshop, tutorial, and keynote (these were treated as being 'grey' literature and excluded).

The study topic was non-empirical, therefore inclusion was not restricted to studies using any specific research method, type of intervention, or outcome measure. Papers authored by both academic and industry researchers were included. The study selection process is shown in Figure 5.1. The steps involved in selection process are described below.

5.2.2.1 Step 1: Searching:

We identified 921 candidate studies that contained matches to the search strings, after excluding duplicate studies. During the search process, especially in the ACM database, articles found in the search results contained similar terms like Service oriented programming, Service oriented systems, Service oriented applications, Service oriented design and Service oriented software engineering. All of these were excluded. We strictly followed the criteria that the search string *must* appear in the title or abstract or keywords of the paper. The search results from the IEEE database using the search string *SOA* brought rather different results. After 100 records, articles on semiconductor optical amplifier (SOA) began to appear in the search results. The search on Science direct brought 205 studies out of which 135 were selected and included in the set of 921.

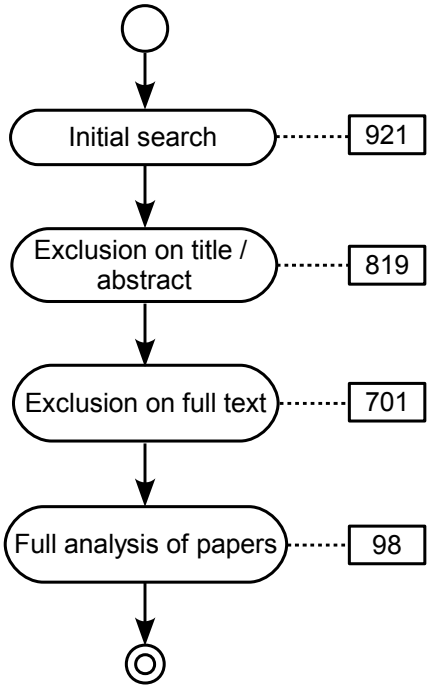


Figure 5.1: Studies selection process

5.2.2.2 Step 2: Exclusion on title / abstract:

The titles and abstracts of papers found in the first phase were then analysed to determine their relevance. We excluded 102 studies that did not meet the criteria such as short papers, keynotes, and tutorials. Here, we faced the same problem discussed in (Budgen and Zhang, 2009), where the low quality of abstracts, use of inappropriate titles, and provision of irrelevant keywords created difficulty to make such judgements.

5.2.2.3 Step 3: Exclusion on full text:

In this round, we examined the full text of the remaining 819 studies and tried to find out whether they discussed SOA any further in the text, other than in the abstract and titles. As a result, 118 of these papers were excluded, along with those that provided the abstract in English but with rest of the content being in another language. Finally 701 studies were left for data extraction.

5.2.2.4 Step 4: Inclusion on definitions:

We carefully analysed how each paper has described SOA concepts and what sources and definitions were used to explain these concepts. Out of the 701 studies, these were 98 papers that specified the definitions of SOA used, and explicitly referenced the sources of these definitions. It was quite interesting and also problematic to find that there were studies that use:

1. a single definition with one reference;
2. a single definition with two or more references;
3. two separate definitions with separate references;
4. the term SOA with one or more citations; and
5. definitions without any references.

Here, we had to decide whether or not to include those papers that fall into categories (4) and (5). In Table 5.5 we have given the count of studies that come under case (4) with certain conditions. In the case of (5), (Papazoglou, 2003) and (Jammes et al., 2005) and (Komoda, 2006) gave their own description of SOA and these were used by other studies: 7, 2, 1 times respectively. Talaei-Khoei et al. (2005) explicitly mentioned what they mean by SOA, however, their definition was similar to the one used to define SOA model (through operations of publish find and bind), without a reference and not cited in any other study. For further analysis, only those studies under cases (1), (2), (3) and (5) have been considered, although case (5) is a bit restricted.

Table 5.1 provides summary of the selection process. The table contains the details about the studies identified through each database, along with the count of studies stating definitions explicitly.

5.2.3 Data Extraction

In data extraction phase, we examined the 98 studies that had been rigorously selected on the basis of the following criteria:

Table 5.1: Summary of Selection Process

Databases	Studies Found	Studies Selected	Studies with Definitions
IEEE Xplore	326	278	60
ACM	390	288	23
Science Direct	205	135	15
Total	921	701	98

- The paper specifically describes SOA.
- The description in the paper contains reference(s) to definitions of SOA.

For each paper the definitions in the text were extracted, together with the details of the references used. During the extraction process, there were some cases where the authors used two different definitions to support their views. So the definitions were listed as two separate definitions although they appeared in the same paper. Also, in some papers a single definition had more than one reference. Studies such as (Casola et al., 2008; Gasikanti et al., 2007; Zhang et al., 2006*a*; Schroh et al., 2009; Choi et al., 2007; Mykkanen et al., 2007; Panahi et al., 2009) came under this category. In this case, the references were verified and only those definitions were selected that contained the actual text. The three cases identified during data extraction are shown in Figure 5.2

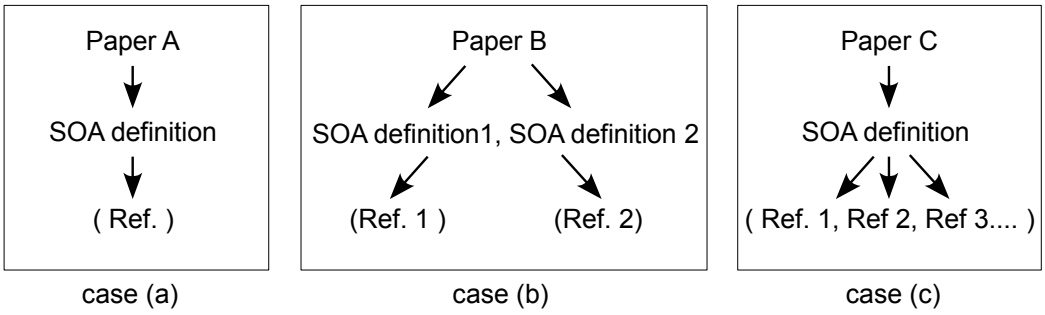


Figure 5.2: Cases identified during data extraction

- Case (a) represent the situation when a paper has used one SOA definition and cited a single reference for that.

- Case (b) demonstrate the situation when two definitions are stated in one paper with two corresponding references. In this case definition count is considered two.
- Case (c) is discussed earlier where a paper provides a single definition but use multiple references.

The data extracted from papers included article information, definitions, source of definitions and their references. This was recorded in a spreadsheet where further information is added that includes the verification of references (where possible) and any inconsistencies between the source and reported text.

The verification of references proved a troublesome process as many references were taken from the grey literature, often being provided on web sites. As a consequence, many were either not available or the URLs have changed.

During the data extraction process, it was also observed that some researchers used the explanation of the SOA model given in Figure 5.3 as a definition. So, the definitions such as those in references (Baresi et al., 2003) and (Massuthe and Schmidt, 2005) come under this category.

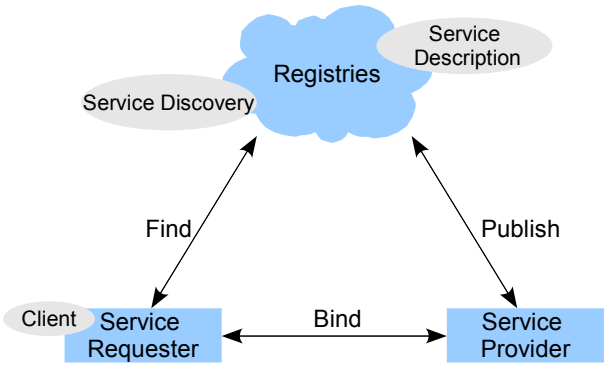


Figure 5.3: Service oriented Architecture Model

From the 98 studies that fulfilled the selection criteria, the total number of "different" definitions found was 96. Both selection and data extraction were undertaken by the author. While the guidelines suggest that two analysts should normally undertake data extraction – we felt it unnecessary to do so for a non-empirical topic such as this one, since no interpretation or quality assessment is involved and any decisions about exclusion are therefore largely objective.

5.3 Analysis

The synthesis process contained two sub-processes: extraction and analysis of terms used in the definitions (process a) and analysis of sources used for definitions (process b). The process is shown in Figure5.4.

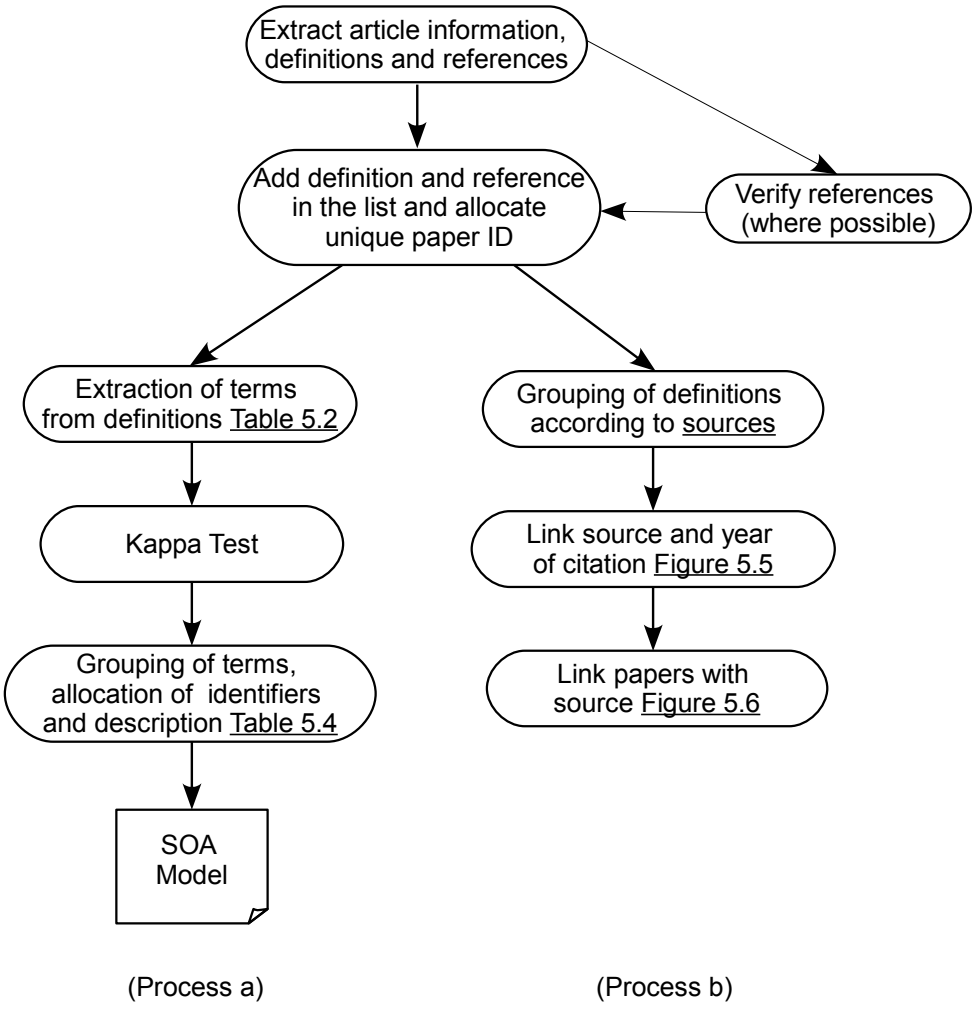


Figure 5.4: Synthesis Process

A thematic analysis (Cruzes and Dybå, 2011) was carried out on the definitions extracted from the papers. The sources of definitions were analysed thoroughly. Thematic analysis is a method for identifying, analysing and reporting patterns (themes) within data (Braun and Clarke, 2006). It minimally organises

and describes the data set in rich detail and frequently interprets various aspects of the research topic (Cruzes and Dybå, 2011). The strength of this method is that it provides flexible procedures for reviewers, copes well with diverse evidence types and can be used for theory-building (Cruzes and Dybå, 2010).

Each sub process presented in Figure 5.4 is discussed in next sections.

5.3.1 Definition Terms and their Classification

In the SOA literature, the early definitions describe the form of an SOA as being a relationship between three actors: service provider, service requester and registry, using *publish, find* and *bind* operations as in Figure 5.3. The later definitions, however, contain additional concepts that seems to address specific requirements introduced by different communities. Table 5.2 provides a list of the terms that have appeared in different definitions over the period of time covered by this study.

The extraction of these terms from the reported definitions was done by both the author and supervisor, working independently. The final agreed list of terms took three iterations, in which we both classified them according to the existing set, and also tried to identify any missing ones.

To check our results we used the Kappa test, which is a statistical measure used to calculate the degree of agreement among experts / raters for qualitative items. The value of κ is given by:

$\kappa = \frac{Pr(a)-Pr(e)}{1-Pr(e)}$

where

Pr(a) = probability or proportion expected by chance;

Pr(e) = probability or proportion observed.

The interpretation of the value of κ is shown in Table 5.3.

The Kappa scores obtained are related to the final agreed set of extracted terms and were calculated for each term over all 98 definitions. The result shows an average agreement of 95% with a maximum value of 1 and a minimum of 0.65, which can be considered as a high level of agreement among raters.

After resolving any differences, the terms were grouped, new identifiers were

Table 5.2: SOA: Terms used and year first used in a definition

Features	2003	2004	2005	2006	2007	2008	2009
Software Architecture	+	+	+	+	+	+	+
Component model	+	+	+	+	+	+	+
Service Provider	+	+	+	+	+	+	+
Service Requester	+	+	+	+	+	+	+
Service Discovery	+	+	+	+	+	+	+
Service Negotiation		+	+	+	+	+	+
Service Publication		+	+	+	+	+	+
Service Registry		+	+	+	+	+	+
Interoperability		+	+	+	+	+	+
Service Invocation		+	+	+	+	+	+
Network Environment		+	+	+	+	+	+
Distributed system architecture		+	+	+	+	+	+
Encapsulation		+	+	+	+	+	+
Interfaces		+	+	+	+	+	+
self-containment		+	+	+	+	+	+
Service Composition / Integration		+	+	+	+	+	+
Broker			+	+	+	+	+
Dynamic binding			+	+	+	+	+
Agility			+	+	+	+	+
Flexibility			+	+	+	+	+
Granularity			+	+	+	+	+
Platform independence			+	+	+	+	+
Framework			+	+	+	+	+
Service Interaction			+	+	+	+	+
Loose coupling			+	+	+	+	+
Heterogeneous			+	+	+	+	+
Architectural style			+	+	+	+	+
Connection Technology			+	+	+	+	+
Business Functions / Processes				+	+	+	+
Reusability				+	+	+	+
Application architecture				+	+	+	+
Architectural paradigm				+	+	+	+
Service description				+	+	+	+
Language independent					+	+	+
Hardware independent					+	+	+
Service contracts					+	+	+
Service Independence					+	+	+
Service bus					+	+	+
Service Consumer					+	+	+
Reuse					+	+	+
Communication					+	+	+
Messaging protocols					+	+	+
Orchestration						+	+
Different Ownership						+	+
Measurable Predictions						+	+
Web services						+	+
On demand						+	+
Choreography							+
Resource Management							+
Uniforming / standards							+

Table 5.3: Interpretation of the value of κ

Value of κ	Strength of agreement
≤ 0.20	Poor
0.21 - 0.40	Fair
0.41 - 0.60	Moderate
0.61 - 0.80	Good
0.81 - 1.00	Very Good

allocated and new descriptions for the groupings were created (these were not extracted from existing ones). In reviewing the terms, it is important to note that not every definition states new concepts, but rather that the same concepts are apt to be expressed in different ways and used in different combinations in the definitions. Table 5.4 provides detail about the extracted terms that have been grouped and each grouping is given an identifier (one word where possible, assuming that the word 'service' is implicit throughout), a brief description of what we mean by the term in terms of SOA characteristics (italicising what we see as keywords), and then a list of the terms that we think are synonymous with that (or partly so).

One common thing that appeared in all definitions is that they provide an interpretation of the underpinning concept of service through its various characteristics. Almost all definitions in one way or the other are focused on explaining the concept of a *service* and its underlying benefits. This is a key factor of SOA popularity, which is being used increasingly by the research community and industry, despite lacking a single agreed definition. Recent studies also discuss SOA implementations; problems of legacy systems; highlight SOA benefits; explain limitations of current technology, address their own solutions (frameworks, models, approaches etc.) and provide experiences, but little effort has been made to explore the term itself. The model in Figure 5.3 often used to explain SOA is the same that is also used to explain the concept of web services.

5.3.2 Definition Sources

For the purpose of analysing definition sources and to find out how frequently they have been referenced, the set of references that formed some sort of cluster

Table 5.4: Grouping of Terms for SOA

Identifier	Description of Characteristic	List of related terms
Architecture	Describes the overall *organization* of a system built from services as the elements, interacting through the use of mechanisms such as SOAP.	application architecture, architectural paradigm, architectural style, software architecture
Binding	The *time* at which a particular service (and provider) is chosen. In an SOA, this can be at the time of use through dynamic binding.	agility, dynamic binding, flexibility, loose coupling, on demand
Capability	The *purpose* of an SOA as viewed from an end-user perspective	business functions, resource management
Composition	The process by which a given set of services are assembled in order to provide a single overall service that meets an end-user need.	choreography, integration, orchestration, service composition
Contracts	The mechanisms for *agreeing* upon the terms and conditions under which a service will be delivered.	service contracts, service negotiation
Delivery	The process that follows composition, whereby service *functionality* is supplied by the service providers to meet end-user needs.	service interaction, service invocation, service provider, service consumer
Distributed Sources	An SOA is implicitly capable of being created using services that are *delivered across a network* and hence that are not necessarily owned or controlled by the end-user or their agents.	different ownership, distributed system architecture, network environment, network
Identity	The characteristics that *describe* a particular service and the means by which these may be accessed.	broker, service discovery, service publication, service registry, service requester, service description
Interoperability	The mechanisms that make it possible to *deploy* services without any knowledge of their location or the means by which they are supplied.	connection technology, framework, hardware independent, interfaces, language independent, platform independence, standards, communication, messaging protocols
Packaging	The characteristics of service implementation that enable it to be treated as a unique and distinct identity.	component model, encapsulation, granularity, reuse, reusability, self-containment, web services
Unclassified		measurable predictions, service bus

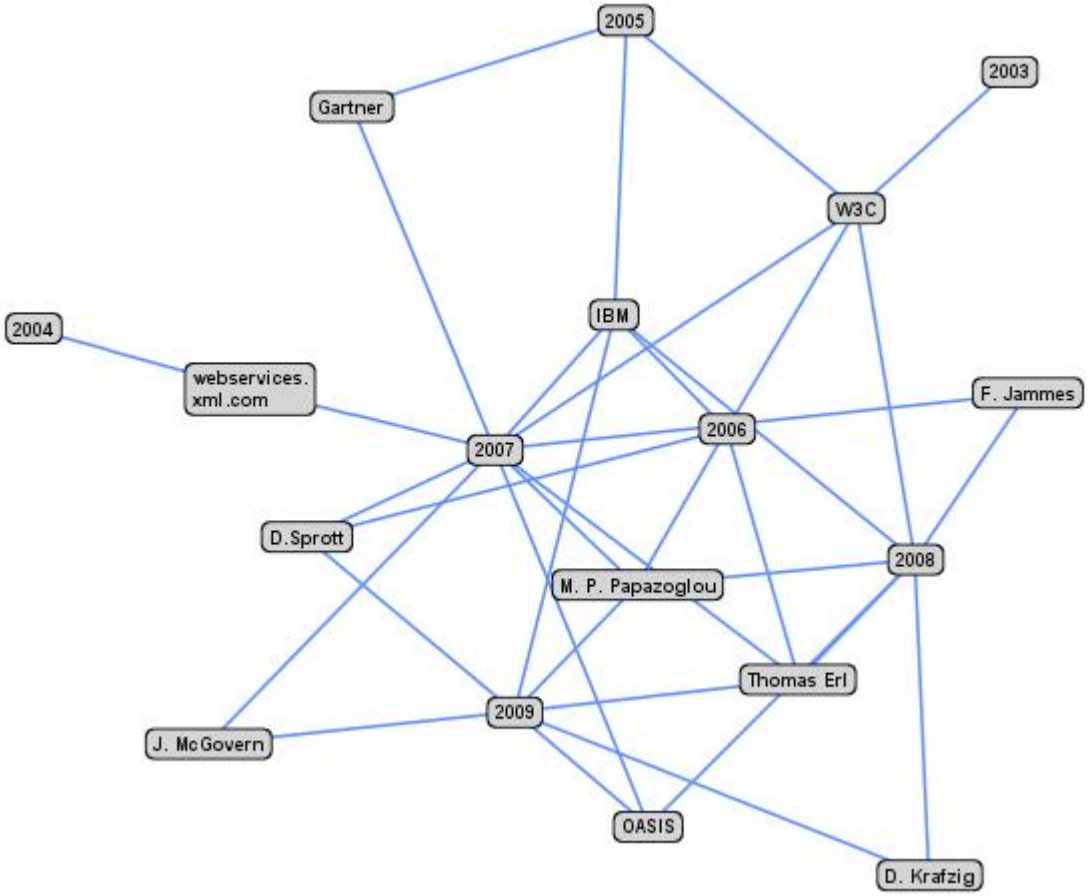


Figure 5.5: Definition Source and reference year

around a source were selected. In doing so, it was decided to include only those definitions that were referenced in more than one paper. This reduced the number of studies from 98 to 65. In Figure 5.5, definition sources cited by these papers, together with their year of publication are shown.

Table 5.5 provides a summary of the frequency with which the definition from each of these sources was re-stated in those papers. We have separated those studies that used the definition without actually re-stating the words. For example, OASIS is referenced in research studies published in year 2007, 2008 and 2009 and there are also five studies that used this source but did not explicitly state the definition.

The definitions, grouped according to their sources of their references to form

Table 5.5: Sources of definition, year published and citation

Source	03	04	05	06	07	08	09	Cited by	Others†
D. Krafzig						+++	+	(Sun and Chen, 2008),(Chmielewski et al., 2008)*,(Nestler, 2008), (Schroh et al., 2009)*	2
D.Sprott				+	+		+	(Allison et al., 2009),(Hoel, 2006),(Gao and Tang, 2007)	-
F. Jammes					+	+		(Mendes et al., 2008),(Barata et al., 2007)	-
Gartner			+		+++			(Zhu and Zheng, 2005),(Henningsson et al., 2007), (Locola, 2007), (Liu and Deters, 2007)	1
IBM			+	+++	+++ +++	+	+	(Massuthe and Schmidt, 2005), (Erradi et al., 2006), (Wong-Bushby et al., 2006), (Zhang et al., 2006*a*)*, (Kim et al., 2007), (Gasikanti et al., 2007), (Griffin and Pesch, 2007), (Kumaran et al., 2007), (Liang and Chung, 2007), (Luthria et al., 2007), (Dimitrov, 2008),(Duan, 2009)	9
J. McGovern					+		+	(Khoshnevis et al., 2009),(Bierhoff et al., 2007)	-
M. P. Papazoglou				+	+++	++	+	(Prinsloo et al., 2006), (Yue et al., 2007), (Kumar et al., 2007), (Luthria et al., 2007), (Tewoldeberhan and Janssen, 2008), (Gu and van Vliet, 2009), (Zhang and Gracanin, 2008)	14
OASIS					++	++++	+++	(Howerton, 2007), (Hrastnik and Winiwarter, 2007), (Demirkan et al., 2008), (Sabucedo et al., 2009),(Canfora et al., 2008), (Hourdin et al., 2008),(Papagianni et al., 2008),(Bakker and Iacob, 2009),(Valipour et al., 2009)	5
Thomas Erl				+++	++++	++++	++	(Briscoe and Wilde, 2006), (Zhang et al., 2006*a*)*, (Fornasa et al., 2006), (Wang et al., 2007), (Sward, 2007), (Ricci et al., 2007),(Choi et al., 2007),(Roach et al., 2008),(Chmielewski et al., 2008)*, (Gogouvitis et al., 2008), (Schepers et al., 2008)*,(Candido et al., 2009),(Schroh et al., 2009)*	22
W3C	+		++	+++	++++	+		(Baresi et al., 2003), (Baresi et al., 2005), (Jørstad et al., 2005), (Jardim-Goncalves et al., 2006), (Dan et al., 2006), (Braun and Winter, 2007),(Huang and Fan, 2007),(Yau and Liu, 2007*b*),(Schepers et al., 2008)*,(Yau and Liu, 2007*a*), (Bocchi and Ciancarini, 2006)	5
webservices. xml.com		+			+			(Nakamura et al., 2004), (Pichitlamken et al., 2007)	1

[]* : papers that have two references or two definitions with different source.
†others : count of papers that cite but did not explicitly state the definition.

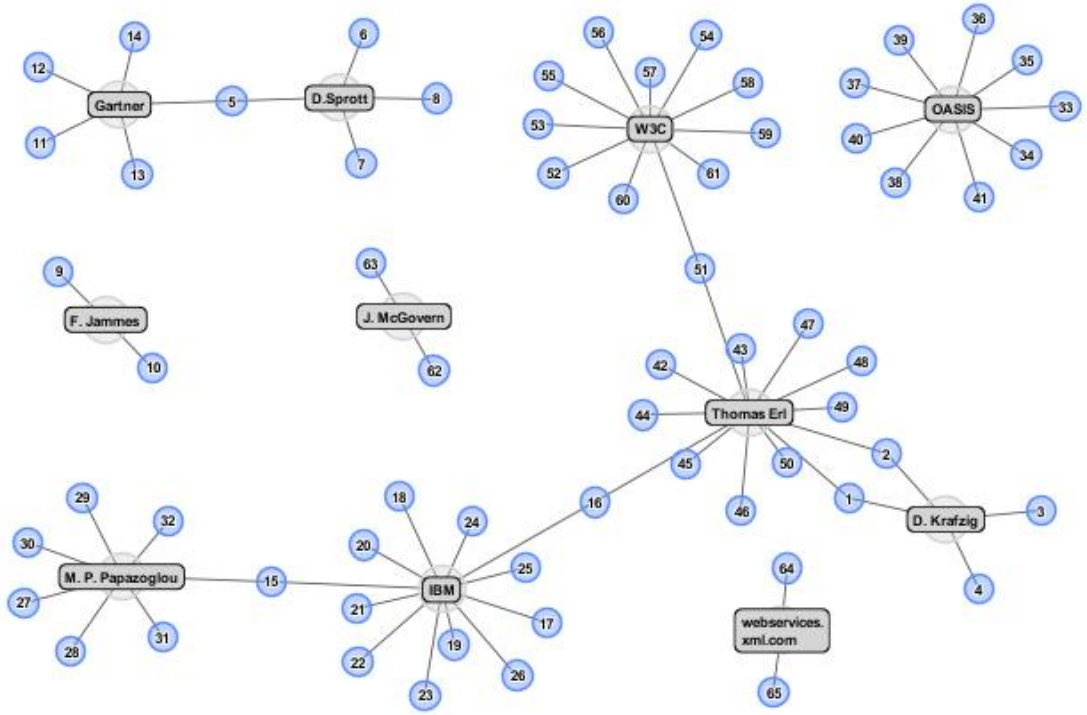


Figure 5.6: Clusters of papers around definitions of SOA

clusters, are shown in Figure 5.6. In this Figure, the central box in each cluster represents the source of a definition. The small circles linked with the central ones represent the papers that reference these sources. The numbers written in these circles represent unique paper ID that was given to each paper during data extraction.

The definition sources should not be taken to imply that all definitions linked with one source are taken from one web site or article, in fact, they appeared in different years (e.g http://www.w3.org /TR /2002 /WD-ws-arch-20021114/ (Baresi et al., 2003) & http://www.w3.org /TR/ 2004/ NOTEws-arch-20040211 (Huang and Fan, 2007)) and also in different books by the same author such as (Erl, 2004) and (Erl, 2005) that were cited by (Wang et al., 2007) and (Briscoe and Wilde, 2006) respectively.

These clusters indicate that W3C and IBM are the main sources from where researchers have adopted definitions, regardless whether they are published on the web sites or in technical articles. The interesting difference for the definitions

provided by OASIS, W3C, IBM and Thomas Erl is that only the OASIS definition is consistent over time, whereas, the other three sources have provided different and evolving versions of their definitions over time.

5.4 Findings

The key research findings from analysis are listed below.

- The various definitions that have appeared in the literature are not contradictory, but they differ in their level of abstraction and also in their assumed context (perspective), with a pattern that is similar to the analysis given by (Budgen, 2003) in his discussion on different (and evolving) definitions of CBSE. From the perspective of a consumer, the service interface characteristics are the only point of interest, whereas for service providers, service implementation is an important issue. For service developers, service composition and service discovery form a challenging task for which they need a solution that is independent of any technological dependencies.

- The community has used a number of different definitions of SOA, but predominantly, those are from W3C, OASIS and IBM (at least, in those papers that actually referenced definitions). The reason why the research community is employing definitions from these sources is very clear. The most dominant and preferred technology that was used to implement SOA over the period covered by the study is web services as defined by W3C. The business solutions in turn are mainly focused on IBM-defined frameworks. In studies (2008 and 2009), the OASIS definition has also been used by researchers, as it explicitly states that services have different 'ownership'. Figure 5.6 provides information that industry defined definitions are increasingly cited by researchers. Therefore influenced by these vendor-oriented definitions, the SOA term itself has become ambiguous. The problem associated with this trend is that the definition changes with change in product features.

- Our analysis includes those papers that have explicitly stated SOA defini-

tion. The great bulk of publications that discussed SOA made no explicit reference to any definition of SOA, suggesting either a lack of awareness of the variety in use or, more charitably, an assumption that they were writing for a community based around one definition.

- There is little or no discussion of the need to clarify SOA, at least in the published literature. (We found only one study in the software architecture domain where the authors analysed the definitions of software architectural knowledge (Boer and Farenhorst, 2008).)

Based on our analysis of different definitions, we find ourselves in a situation similar to that encountered by Shaw when defining a vocabulary for software architectures (Shaw and Clements, 1997). They observed that the designers make use of extensive descriptive vocabulary to explain their system which is informal, casual and ambiguous. Therefore, they considered it necessary to establish a common vocabulary for architecture styles so the communication about styles become more effective.

5.5 Discussion

5.5.1 Related Work

The studies reported in (Gu and Lago, 2009) and (Oliveira et al., 2010) have presented the outcomes from systematic literature reviews of service-oriented system engineering(SOSE) and SOA respectively. In the first study, the review aims to identify the challenges faced by SOSE as discussed in studies published during the time period of 2000 to 2008. The paper classifies these challenges to identify research trends in SOSE and based on this, establishes a future research agenda for SOSE. The second study has analysed the establishment and use of reference models and architectures that have been proposed to support service-oriented systems, their domains and new research lines. This study concludes that while SOA is receiving more attention from both academia and industry, there is no consensus about how to present SOA reference models and architectures. Apart

from these studies and the one described in the next paragraph we have not been able to identify any attempts to discuss the concept and vocabulary of SOA.

A recent publication on SOA from the SEI, in the form of white paper (Lewis, 2010), has provided a discussion of SOA terminology. The report aims to establish a baseline of terms for service-oriented systems and describes SOA as "a way of designing, developing, deploying and managing systems.." with SOA features that are almost the same as those that we found in different definitions. However, from a technical point of view, the author has described SOA as "an architectural style or design paradigm" and separated it from a system architecture or a complete system.

5.5.2 Answering the Research Question

The research question asked for the mapping study at the start of this chapter was "What are the key characteristics of an SOA?", and our analysis of the literature suggests that the list presented in Table 5.4 effectively captures the current thinking about SOA characteristics.

Two practical questions that this study raises is whether it would be helpful either to:

- adopt one widely accepted definition of SOA within the community; or
- require authors of papers to make clear which definition(s) they are using.

While the first is an attractive idea, it is not clear that this is necessary, although arguably the lack of such a clearly shared understanding has possibly impeded the early stages of other developments in software engineering, such as component based development. The second is probably more practical and could easily be adopted by workshops and conferences.

Returning to the motivation for an original research question, there is no indication that lack of a common definition has so far hindered the developers of SOA, perhaps because the elements that differ between definitions are not in conflict. However, for the purposes of modelling and developing SOA-based systems, a common understanding of the elements of an SOA are essential. So, one of the questions that this study raises is how can the characteristics identified

in Table 5.4 be modelled? For example, an important element in determining how a system will execute is that of *contracts*. A contract can potentially take many forms, and while overall system functionality may not be affected by the choice of the form of contract, it may well influence non-functional issues such as performance, quality of output, and cost, since it determines which service provider will be bound when the service is required. Such issues do not arise in more 'conventional' statically-bound forms and so cannot readily be modelled using existing notations.

5.6 Conclusions

This secondary study is different from most previously published ones, both in the evidence-based literature, such as those listed in (Kitchenham et al., 2009), as well as the services literature, since the focus here was not to find empirical forms of evidence, but rather to find evidence about how the concept of SOA is defined and used.

We have tried to answer the question "*What are the key characteristics of an SOA?*". What has been demonstrated by the mapping study is that definitions of SOA do certainly exist, although there is no one universally adopted one, and that different communities do not even seem to be aware that other definitions exist. Table 5.4 summarises the characteristics of an SOA, as extracted from the literature. Characteristics such as interoperability and reuse, with or without dynamic binding, have much to offer. They also resonate strongly with the emergence of the semantic web, allowing resources to be physically distributed and moving away from a centralised and static model of resource management. However, as we have demonstrated, SOA is still an emerging concept. There are some substantial challenges to overcome in order to achieve its full potential, not least identifying practical forms of design model, based upon the characteristics, and this is more likely to occur if the SOA community converges on a shared set of concepts for the meaning of SOA.

5.7 Summary

The research community is publishing widely on SOA, however, while performing our research into service design issues, we became aware that the term SOA was apt to be used in rather different ways with no one clear definition. So, to identify what characteristics are generally considered to constitute an SOA (the research question), a mapping study is conducted to collect evidence about how the concept of SOA is defined and what the key characteristics of an SOA are considered to be by the research community.

Through the mapping study, SOA features are identified, grouped and interpreted to be used to explain the key concepts. It was also identified that there is no one single clear definition exist for SOA, rather the same concepts with slight variations are used by industry and within academia. As no agreement exists on SOA definitions, the problem for creating design models to represent SOA characteristics will remain an issue.

Chapter 6

Use Case - A Control System for a Small Scale Energy Zone

6.1 Introduction

In this chapter we describe the use case model that was developed as part of the case study. The case study process is shown in Figure 6.1. The grey boxes represent the elements of the process that are covered in this chapter. The purpose of employing a case study was to apply ideas about SOA design to a problem from the 'real world' instead of using 'toy' examples. From the experience of the mapping study on SOA, we have observed that while the examples used in the SOA literature do illustrate the proposed methods, they are artificially constructed, lack originality and are narrow in scope. The other issue with these examples is that while they were represented as case studies, the forms of these are not consistent with case study research as discussed in (Runeson and Höst, 2009; Yin, 2008).

This issue is addressed by other researchers in SOA community such as a recent study by Espinha et al. (2012). While searching a case study for their research, they identified that there is a lack of suitable case study that can be used by the researchers in order to develop applications, assess their research ideas, and use that for comparison and also for benchmarking. Therefore, they conducted a literature survey on the case studies used in SOA research. They reported

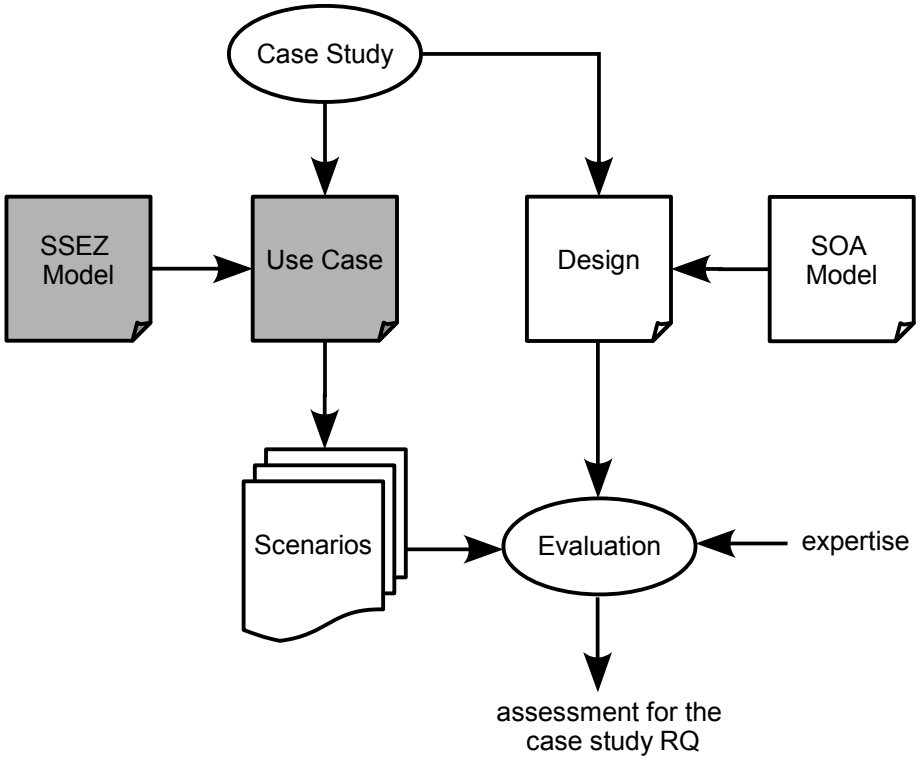


Figure 6.1: Case Study Design

fourteen case studies published in CSMR, ICSE and ESEC/FSE conferences and also from European S-Cube project on service based applications (SBA). They identified that there are case studies created by authors which are quite small to be representative of real service based system such as (Bianculli and Ghezzi, 2007) and (Ardissono et al., 2006). And there are some that include more services such as (Baresi et al., 2004), although their details are not available. The same is the case with industry based research publications. The researchers mention that their approach is applied to real applications but details are not available. Therefore these studies cannot be compared or replicated. Espinha et al. (2012) has also suggested a case study named as 'Spicy Stonehenge' constructed from an existing open source system.

To conduct case study we have developed a case study protocol attached as Appendix B. In this chapter we explain the case study process and discuss the use case model in detail.

6.2 Case Study

The elements considered in the design of the case study are explained in detail below.

Rationale: We undertook case study in response to the questions raised from performing mapping study on SOA. From the analysis of literature published on SOA we became aware of the fact that one of the challenges for the development of SBA is the design of such applications. In particular, the need of the detailed description of the real world problem and the experience of designing a new service based application. Therefore, primary rationale for the case study was to provide deep understanding of the problem and use this along with the outcomes of previous study to construct a proof-of-the concept design.

Case Study Domain: The case study has been taken from the energy engineering domain. The reason for selecting this domain is mainly one of having access to the resources within the School. In addition to this the concept of smart grids, which is similar to that of a small scale energy zone (SSEZ), is considered an important idea in future power systems. The concept involves distributed generation independent of the grid. This helps supplying power when there are blackouts in case of extreme weather conditions such as 'Hurrican Sandy'. On this occasion, the distributed generators helped the universities, hospitals and business to keep their power supply independent of the grid (Venables, 2012). Further to this, these power systems integrated with prediction can help to remove loads from grid and to provide power to the consumers by estimating future power needs in the area. In Muller (2012), a future picture of power systems is presented in the form of distributed energy management systems that combine weather data, real-power, prices, and consumer demand to produce forecast and operating schedules for power plants.

Case Study Type: This is being conducted as a single-case study, due to the breadth and complexity of the domain.

Unit of Analysis: The 'case' (or unit of analysis) is the design for a small scale energy zone (SSEZ) control system. This is a real time system that is run by an Energy Services Company (ESCO) to manage the electrical network and fulfil energy needs in an SSEZ.

Use Case Model: We have constructed an operational model of the SSEZ control system in the form of use case. This contains electrical network information, operational goals, key factors related to control system and the data involved.

Characteristics of Use Case: The characteristics that were considered to be represented in the use case were the needs for adaptability; multiple distributed sources of information; and negotiation. The SSEZ is a distributed system that requires information from different sources (network and service providers). The involvement of service providers fulfils the requirement of negotiation to be present in the case study. Being a real time system, adaptability is also an important attribute of this case. The match between the characteristics of the use case and the service oriented architecture (SOA) are discussed in (Anjum and Budgen, 2012*b*).

The characteristics of an SOA, as mapped on to the use case are listed below.

- **Architecture:** The SSEZ control requires information to be collected by a centralised element. Which means the composition takes the form of a 'tree' where lower level services (network information) are composed into higher level services such as control service.
- **Binding:** The state of the SSEZ needs to be reviewed at regular intervals based on current values and forecasts for demand, provision and weather. Each review may involve a different set of information sources, especially as generation and storage are added and removed from the current profile of the SSEZ. A model of late (runtime) binding as provided in an SOA is therefore particularly well-matched to this need.

- **Capability:** The meaning of this is the same for both an SOA and for the system as a whole, and is concerned with the overall functionality. For an SSEZ, it is related to the ability of the software to use available information to perform the necessary resource management, by modelling demand and provision for the next period of time and then plan any changes accordingly. That in turn is related to the set of algorithmic models used for prediction, the state of the system at any time, and the forecasts for the next period.
- **Composition:** Composition involves bringing together multiple sources of information to facilitate a decision. An example of this in an SSEZ might be the use of information from a wind farm about its current output, together with forecasts of likely demand and a weather forecast, to decide whether to increase or decrease provision from other sources. Composition is a core feature of a software service model, which also provides the means to determine the source of a service (e.g. weather forecast) when the request is issued (late binding), and hence this characteristic makes a service solution well matched to needs of an SSEZ.
- **Contract:** The contracts are rules of engagement with other services or between service provider and consumer. In the case of an SSEZ, contracts for different types of forecasting services may involve terms that address the granularity of data and service availability. If the rules of engagement change, then re-negotiation may result in a change of service provider.
- **Distributed Sources:** The SSEZ is well matched with this implicit feature of an SOA. The data coming from generation, demand and weather is already coming from distributed sources with different ownership, and hence the elements in the overall system will be interacting over the network.
- **Identity:** The characteristics of demand, generation, weather and their forecasts are different and are accessed through different means. These characteristics will help the generation forecast service to select

the most suitable service provider for its processing.

- **Interoperability:** For an SSEZ this is an important feature as its elements need to be able to communicate information with each other in a consistent framework. To ensure interoperability, the electrical power industry is working on two standards: Common Information Model (CIM) and IEC 61850. In an SOA, interoperability is provided by a combination of XML-based messaging forms and an ontological model that provides the necessary semantics. This is clearly highly consistent with the above.

Data Collection: The case study data has been collected through the use of both method triangulation (Oates, 2005) which is shown in Figure 6.1 and also by employing triangulation of multiple data sources. Table 6.1 provides details about this. These techniques are used to increase the validity and consistency of the data. In the case study, the confounding factors that may impact the result are not entirely known or cannot be controlled. This is because in a case study the researcher does not have the same control as is in an experiment. Therefore, Yin (2008) has suggested two ways to handle this problem.

- By conducting multiple case studies or use of multiple cases in a single case study.
- Use of triangulation to gather evidence in a single study.

Time Period: The case study time period was longer than defined in the protocol. This was due to the iterative approach of collecting data and analysing it to identify gaps in the information. The second reason was that of need for the domain knowledge. A significant amount of time was spent on understanding the terminologies, and in collecting the relevant data. The research is interdisciplinary, therefore, vocabulary played an important role in data collection and representation, and was needed to ensure that the documents produced could be understood by both disciplines.

Table 6.1: Data Collection

Method Triangulation	Data Triangulation
Use Case	The data about use case was collected through interviews with domain experts and the study of supporting documents that include research papers, technical reports, and thesis. The information collected in one interview session was used in the next with some additional documents to collect feedback. This was important to identify inconsistency in the collected data and to make sure that the domain information has been understood correctly. The interview sessions (formal and informal) with domain experts helped with understanding the domain and the supporting documents provided sources for its vocabulary. This was mainly a process of requirements elicitation. In Appendix C documents from discussion sessions are attached.
Design	The SOA design model constructed as the part of case study provides details about the service and functional components created from requirements. The design elements are presented through abstract diagrammatical forms.
Evaluation: The walkthroughs and interviews were conducted as part of case study for the validity of use case and the design.	The data for evaluation was collected both from walkthrough sessions and also from interviews with participants. The data about walkthrough sessions contains details about the review. The data is recorded in audio and video files. The feedback about walkthrough process, and design presentation is collected through interviews. The data is collected through by recording interviews.

The use case constructed for use in the case study is discussed in the next sections. The use case was constructed in such a way that it could be understood by both domain experts and the software engineers. There is domain specific information for example the electrical network configurations. This information is provided because the case study we have taken is the part of an ongoing research programme in renewable energy. To get information from energy engineers you have to provide them with the scope of the network as a reference point in order to discuss related issues.

The details in the use case provide a wider scope of the problem. This is useful, particularly when a domain is new and unfamiliar. This also helps in the development of the application from scratch. In this research we are not only constructing the case study to represent a real world problem but also using it as a tool to explore and represent the SOA design problem.

In the next section we discuss the use case model that was developed as the part of the case study.

6.3 Use Case

The use case describes the situation where an Energy Services Company (ESCO) is maintaining the specified electrical network by generating electricity through renewable energy resources[1] and trying to avoid the use of conventional power[2] where possible.

The main objective for the ESCO is to provide electricity to its customers in an efficient, reliable and cost effective way. To achieve this target, the ESCO needs to be able to predict demand and generation for its electrical network; to take decisions, where required, about the buying and selling of energy, as well

[1]According to the Environmental Protection Agency (EPA), renewable energy includes resources that rely on fuel sources that restore themselves over short periods of time and do not diminish. Such fuel sources include the sun, wind, moving water, organic plant and waste material (biomass), and the earth's heat (geothermal). http://www.epa.gov/greenpower/gpmarket/index.htm

[2]Conventional power includes fossil fuels (coal, natural gas, and oil) and the nuclear fission of uranium. Fossil fuels have environmental costs from mining, drilling, or extraction, and emit greenhouse gases and air pollution during combustion. Although nuclear power generation emits no greenhouse gases during power generation, it does require mining, extraction, and long-term radioactive waste storage. http://www.epa.gov/greenpower/gpmarket/index.htm

as when to adopt an islanding mode and to take decisions about demand side management (DSM). For this, the ESCO has to gather and process network and commercial data from different sources and use this to make real-time decisions.

The rest of this chapter consists of a requirements specification based upon a particular instance of the case study.

6.4 The SSEZ Network

The details about the example small scale energy zone (SSEZ) electrical network resources forming the use case, and about their usage are provided below.

6.4.1 Network Configuration

Our example SSEZ comprises of a HV/MV 33kV three-phase network with three 11kV feeders: one for residential use, one for industrial and one for commercial. The residential feeder serves 500 customers with 1MW demand. Photovoltaics (PVs) are attached with each house with a capacity of 2kW. Demand for individual customers can vary from 400W to 15 kW. The average peak demand for domestic customers after taking diversity (ADMD[1]) into consideration is 2-3kW. The minimum load can be considered to be 30% of the peak demand.

The commercial feeder covers a 1 MW demand for a supermarket, as shown in Figure 6.2. The industrial feeder has a 2MW demand for a factory. A wind farm near the factory is connected to the industrial feeder with the capacity of 3MW, with each wind turbine having the capacity to generate 500 kW. An energy storage unit (ESU) is installed next to the wind farm which has a storage capacity of 2 MWh. Figure 6.3, shows the configuration of the SSEZ governed by the ESCO.

Table 6.2 provides a summary of electrical network resources. The load type with its maximum and minimum limits is discussed in Table 6.3

In Table 6.4, details about distributed generators (DGs) are given in detail with actual capacity and after applying capacity factor (CF)[2]

[1]After diversity maximum demand

[2]Capacity factor (or load factor) expresses the amount of electricity produced by an electric-

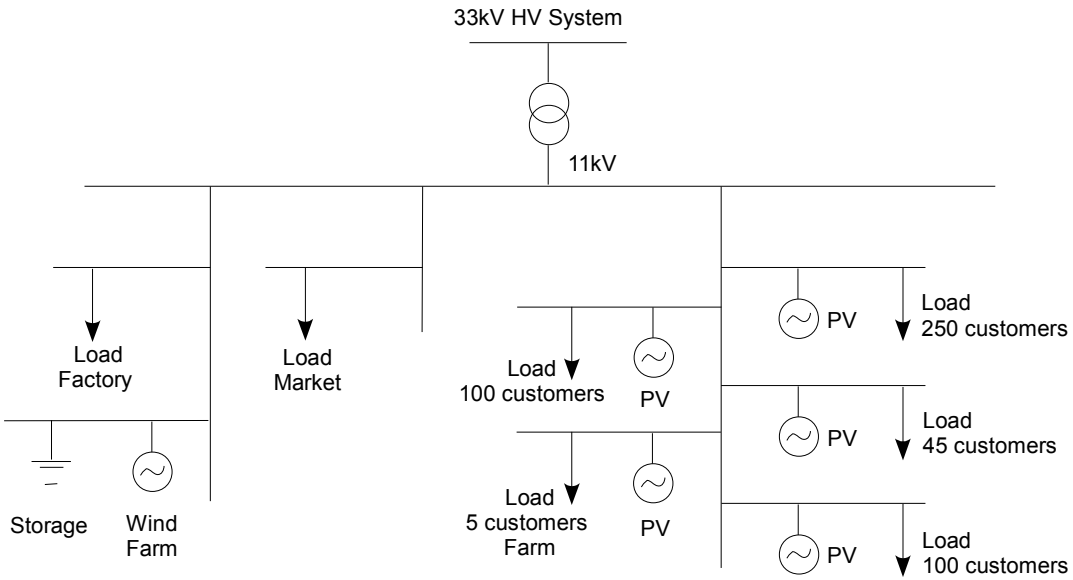


Figure 6.2: Electricity Network

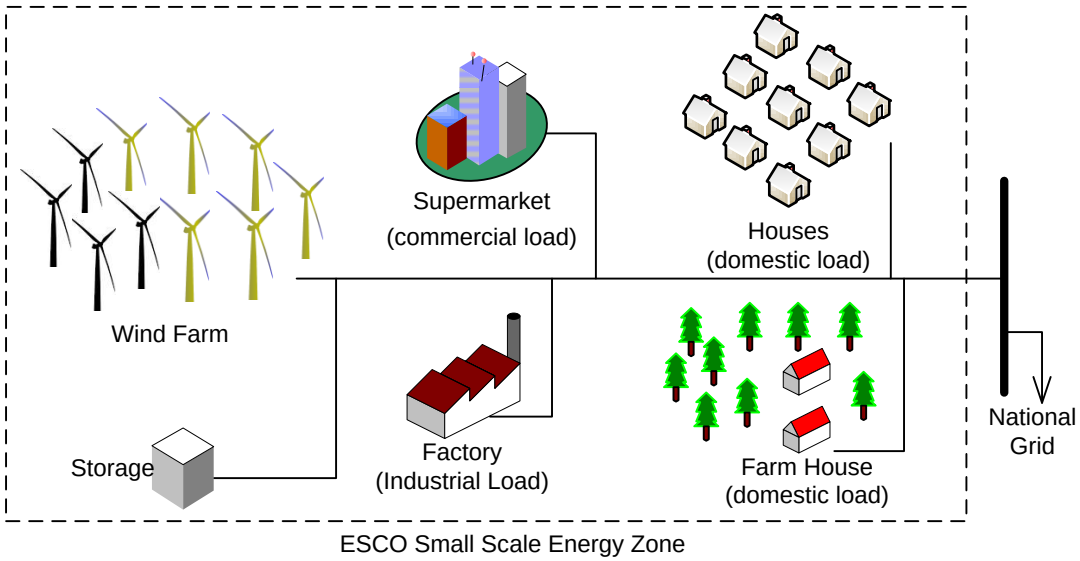


Figure 6.3: ESCO Small Scale Energy Zone

Table 6.2: Summary of Electrical Network

Feeders	Load type	Load	DG
Feeder1:domestic	500 customers	1MW	PV
Feeder2:commercial	one market	1MW	Use power coming from Wind farm, PV or storage device
Feeder3:industrial	one factory & storage unit	3MW	Wind farm & Storage device

Table 6.3: SSEZ Demand

Load Type	Min Load		Max Load	Notes
	Summer	Winter		
Domestic	300 kW (30%)	700 kW (70%)	1MW	seasonal
Commercial	500 kW (50%)	600 kW (60%)	1 MW	Time 9-5 constant throughout the year
Industrial	200 kW (10%)	200 kW (10%)	2 MW	option1: 9-5 shift, option2: 24h shift
Storage	-	-	1 MW	

Table 6.4: Power from Distributed Generators

Generators	Capacity / generator	Quantity	Min Power	Max Power (CF)	Max Energy (CF)	CF
PV	2 kW	500	0	0.2 kW	1752 kWh	10%
Wind Turbine	500 kW	33	0	150 kW	1314 MWh	30%
Storage	1 MW	1	0	1 MW	2 MWh	-

6.4.2 Network Operational Goals

Depending upon the network conditions, five operational goals that need to be met in order to run an SSEZ have been identified in (Trichakis et al., 2008). These goals are summarized below (but an important point to be considered here is that not all goals can be met at the same time).

1. Zero power export: If local generation capacity is less than peak local demand, the goal could be to maintain a zero power export position to the distribution network.

2. Zero power import: If local generation capacity exceeds peak local demand, the SSEZ could aim for zero power import.

3. Zero power import and export (self-sufficient): If there is a close match between peak local demand and local generation capacity, the SSEZ could attempt to operate self-sufficiently, with no power exchange with the distribution network.

4. Constant power import: this involves operating with a fixed power demand, by having a constant level of power import from the distribution network.

5. Dispatchable power export[1]: involves providing dispatchable power to the distribution network over a specified time period.

ity generator as a percentage of the maximum theocratical production from generator. Wind generators typically operate below rated capacity for around 90% of all hours(site dependent); this mode of operation results in an annual average capacity factor for wind turbines substantially below that typically achievable for conventional generators. For UK a 30% capacity factor is generally representative of the current level of wind power development (Sinden, 2007). In Table 6.4, maximum energy with CF is calculated for the period of one year (power capacity(.5 MW) $*$ 24(hours/day) $*$ 365(day) $*$ 0.30(CF) $\equiv$ 1314 MW

[1]Electricity is provided on the request of grid operators or by distribution network operators (DNO) using a short term contract. Where an energy system can be expected to provide a continuous output (given normal conditions) it is termed as "dispatchable", thus offering the ability to furnish power on demand to meet changing loads; e.g., hydrocarbon-based or nuclear power plants are dispatchable, but solar and wind power are not (EnergyLibrary, 2011). An integrated energy system where wind and solar energy sources are coupled with storage devices can be used as dispatchable during peak demand, thereby enabling their broader use (Garrison and Webber, 2011).

For power dispatch, the link between control system and market operators (or local DNO) needs to be established to allow the control system to determine if it is capable of delivering the specified power to the grid and to decide how best to achieve this.

6.4.3 Network Considerations

The factors that are important and need to be considered about network resources are described below.

- The capacity factor for each wind turbine is 10-35% and for the PV it is 10%.
- The time for demand and generation patterns varies from 30 minutes to days and weeks.
- The weather forecast is required from 30 minutes to 1h, 2h,5h or 3 days ahead in order to run renewable energy generators and for prediction of demand and generation in the zone.
- Storage can be used as a form of demand, therefore if there is a need to increase demand (bringing the loads forward), the storage can be charged.
- Because of finite capacity of the storage device, it is necessary to consider its state of charge (SOC) and voltage limits when using it to increase demand.
- Storage may be used to take advantage of price and cost differences by charging up with surplus electricity at low cost times and discharging in peak cost periods (known as arbitrage).
- An important consideration about storage limits could be to stop discharging if the SOC is $\leq 50\%$. Once storage is fully drained it takes more time to fill it to full capacity.
- The percentage of demand that can be deferred to flatten the peak demand according to Ofgem[1] is between 5% to 10%, which is considered a reasonable assumption.

[1]Ofgem is the Office of the Gas and Electricity Markets http://www.ofgem.gov.uk

- In the event of an overvoltage or violation of the thermal limits in the reverse direction, the generation could be curtailed to resolve the issues.
- Power deficit or surplus inside the zone could cause technical problems or commercial opportunities or both or none.
- The important control factor is that the power balance must be met at all times as the national grid is a 'slack-bus[1]'.
- The SSEZ is a 5MW network and at any time power in the network should not exceed this figure.
- The wind turbine pitch control (blade angle) can be used to decrease power output.
- Wind turbines cannot run at full capacity all the time, due to variable wind speed. If wind speed is very low no electricity can be produced and if it is too high, then wind turbines have to shut down to avoid damage. The wind speed is categorised as cut-in speed (i-e 4-5 m/s), rated speed (i-e 10-15 m/s) and survival speed (i-e 25 m/s).
- The system technical constraints mean that the network could not afford more power due to its assets limits[2].

[1]The load-flow problem requires that total generated power matches with the total demand along with transmission losses. However, such losses cannot be determined beforehand, therefore, total generation needed to supply a known demand cannot be exactly specified a priori. In consequence, it is necessary to have at least one bus (the slack bus) whose real power generation can be rescheduled to supply the difference between total system load plus losses and the sum of active powers specified at generation buses (Expsito et al., 2004).

[2]Network constraints are determined by statutory regulations and also by equipment rating. Equipment or component rating varies according to the weather condition such as wind speed or solar radiation. Component temperature is not a constant value and depends on the energy balance between the heat produced inside the component and the heat exchanged on its surface. The dynamic thermal rating (DTR) concept consists of estimating or measuring component temperature or real current carrying capacity, in order to allow the power system component utilisation to be safely increased. DTR is a part of active management technique where power flow can be increased in a safe manner in the certain sections of the network in a cost effective way (Roberts et al., 2008; Michiorri and Taylor, 2009). The LV distributed network constraints include voltage rise limits, voltage unbalance limits, thermal limits and reverse power flows limits (Trichakis et al., 2008).

- Electricity price can be considered as a commercial constraint, and in the case of a system, the ESCO has to decide whether or not it should sell electricity to the grid (export electricity from the network).

6.4.4 Key Factors

The key factors that must be considered while making decisions are:

1. Technical: Network constraints have the highest priority because secure system operation is the single most important requirement of the SSEZ control system (Trichakis et al., 2008).

2. Demand: Keeping essential supplies to customers is essential.

3. Storage: The decision about when to charge and discharge storage is linked to the fact that a storage unit is best used when it is fully charged or is discharged steadily over a longer period of time. The longest time period for discharge considered here, in this use case, is from 30 minutes.

4. DSM: Demand side management (DSM) is important especially in cases of power deficit. DSM is best suited to a short term situation (15 minutes in this use case), when a rapid response is required. Demand cannot be deferred for a long time due to Ofgem policies and the risk of customer relationships.

5. Export: It is better to export surplus energy before planning to turn off power generation. Energy can be exported when energy market prices are high and when there is surplus energy in the zone.

6. Import: Importing energy from the grid is necessary when the green energy supply is not enough to fulfil the demand, or when energy market prices are very low.

6.4.5 Assumptions

- For decision making purposes, we need to define what is meant by short, medium and long term. In this use case, the short term is classified as

around 15 minutes, medium is 30 minutes and long term is 60 minutes.

- Different assumptions can be made about the ESU's state of charge (SOC) such as: 25%, 50%, 75%, 100%.
- It is assumed that the network is in a normal operating condition and that no faults are present.
- The initial assumption is that the system parameters need to be revised at half hour intervals, in accordance with UK and most European electricity market procedures.

6.5 SSEZ Network Data

To manage the SSEZ, the ESCO needs to collect data from its electrical network resources, which are considered as internal elements, and from weather and energy market services which are categorised as external elements. Figure6.4 below provides an abstract view of the ESCO and the sources that it interacts with.

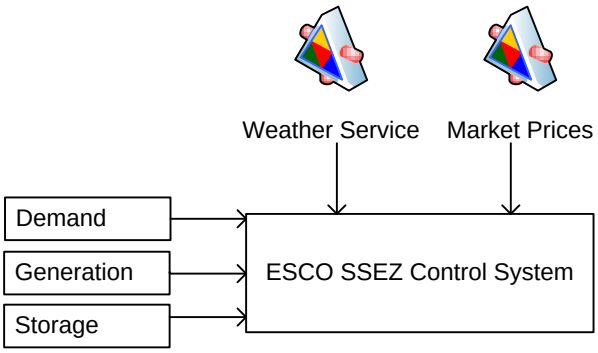


Figure 6.4: ESCO Data Sources

The SSEZ operational data consists of energy consumption data (demand), generation outputs, status of energy storage unit (ESU), weather forecast and energy market prices. The ESCO will communicate with weather and market services through the internet and it is assumed that they will be available in the form of a web service. The details of data and its sources are discussed below:

Table 6.5: Network Operational Data

Category	Source	Information Type	Data
Weather	Webservice (Metoffice or any other weather service provider)	Current & Forecast	Time of day, Wind Speed, Wind Direction, Temperature, Ice, Rain-Soil resistivity, Solar Irradiance, Position of Sun, Cloud Conditions, Ambient Temperature
Current Demand	Smart meters[1]	Current data	Load data, Energy Consumption, Date & Time
Historical demand	Webservice		Load data, Energy Consumption, Date & Time
Generation	Wind farm & PVs	Present States	Power output, Energy, time of day
Storage	Storage device	Present State	State of charge, Voltage
Energy Market	Webservice (National grid)	Current & Forecast price	Settlement price, System buy price, System sell price

Weather Service: Weather forecast data is primarily used for two purposes; generation forecasts and demand forecasts. In the case of renewable generation from wind and solar farms, weather forecasting plays an important role by providing information about expected wind speed and direction. In the same way, weather has an impact on energy demand - for example increased use of air conditioning on hot days and increased heating needs on cold days. Both free and commercial weather services are available via the web, and these can be used by the ESCO to determine what may happen in the short term with regard to wind speed, solar insulation and ambient temperature.

A basic level of weather service is available on the internet for free, but to get accurate and refined data, a higher cost option needs to be used.

Demand: The demand data provides information about a customer's electricity consumption. Customer demand profiles are required to analyse the

[1] A smart meter is an advanced meter that measures energy consumption like conventional meters but have the additional functionality of communicating information to the central system for monitoring and billing. Smart meters transfer real-time energy consumption information and have the ability of bidirectional data communication (Depuru et al., 2011).

electricity usage in the network and also to predict future demand. Smart meters have the ability to provide real-time data through their open network interfaces.

For demand prediction, it is required to know about how much power is currently being drawn, how much energy has been used in last half hour, what is the weather forecast, what is demand now, as well as the time of the day, date and historical demand profile. Other important factors that need to be known for demand prediction (apart from day-to-day variations) are the seasonal demand variations, annual events like Christmas, and any major sports occasions, social events etc. when electrical appliances are likely to be in use.

Historical Demand Profile: Historical demand profiles provide information about previous profiles of energy use that show past energy consumption on a specific day, time and year. This information can help with analysing likely demand patterns for a specific time of the year or a specific occasion.

Generation: The generation data consists of output from different generation sources like wind turbines, solar panels etc. in order to analyse current available energy in the network. The energy generators can be owned by the ESCO itself or they can be provided by different generation plant owners.

To make decisions, the ESCO needs to calculate generation cost, and the amount of energy and power to be generated. If the ESCO is focusing on balancing and metering then it needs power and energy as net amounts. But if it is considering network issues, where wind turbines and solar panels are connected to different parts of the network, then power and energy information need to be collected on each wind turbine and solar panel. In this case, network constraints might affect both forecasting and delivery.

The wind speeds at which wind turbines operate are categorised as cut-in speed (i-e 4-5 m/s), rated speed (i-e 10-15 m/s) and survival speed (i-e 25 m/s). These speed limits put constraints on the wind farm generation plans. There is a relationship between wind speed and pitch angle, that can be used to reduce the output as discussed in Appendix D.

In this use case we are assuming continous provision of the following information from the wind farm:

- Wind Turbine/ farm Power output
- Wind Speed (m/s)
- Time Stamp

The information about power produced from photovoltaics (PVs) that is required is listed below:

- Power output
- Solar irradiance
- Time Stamp

Storage: The two main pieces of information that need to be considered about the ESU are: state of charge (SOC) and current state of the ESU. In some cases voltage condition is also considered (detail in Appendix D). Table 6.6, shows the SOC and its state relationship.

Table 6.6: SOC, current and future states

SOC(%)	Current State	Future State
100	charge	stop / discharge
	stop	stop / discharge
70	charge	charge / stop / discharge
	stop	charge / stop / discharge
	discharge	charge / stop / discharge
50 or less	no discharge allowed	stop/ charge

Energy Markets: The Grid is the backbone source for energy provision. Depending upon market conditions, an ESCO can decide to buy or sell electricity to the Grid.

To obtain up-to-date wholesale buy and sell prices for grid electricity, an ESCO needs to communicate with the energy market. The electricity market price will help the ESCO to determine when to export/import electricity

to the grid and when to charge and discharge its storage units. More detail about the balancing market is given in the Appendix D.

Current and predicted market price is required whenever there is a need for import or export of energy. Energy is imported from the grid only when renewable generation is not enough and when there is a risk of a loss of supply to customers. In this case, brown energy is used, which has environmental effects, and this is the situation which an ESCO will usually try to avoid. Energy export occurs when the current output of generators is more than required in the zone. If a full load is being exploited, the storage has reached maximum capacity and there is still excess energy in the zone, then to avoid energy wastage it needs to be exported to the grid. It is a key part of an ESCO's strategy to decide when to import and export energy as the price to export energy to the grid is only approximately 2p/kW-hr, whereas the energy purchase price from the grid is around 7-13p/kW-hr. Market prices include current system buy price (SBP) and system sell price (SSP). The details about SBP and SSP are given in the Appendix D.

System History: There is need to maintain record of the data about the SSEZ network. For this the decisions made by control system should be logged along with the information collected from network and the data obtained from outside the network such as weather data and energy market prices. This information will help to maintain SSEZ energy profile. Which further can help in making prediction about the energy condition in the zone.

6.6 Summary of Functional and Non-Functional Requirements

The functional and non- functional requirements identified from the use case are listed below:

Functional Requirements:

- Maintain the power balance in the SSEZ
- Access and evaluate data coming from the electrical network resources

- Import power from the Grid as necessary
- Export power to the Grid as necessary
- Access Weather Services
- Access Energy Market Service
- Predict demand for the SSEZ
- Predict generation for the SSEZ
- Maintain a system history by logging key data

Non-functional Requirements:

- **Time:** The control system evaluates its network resources on 30 minutes intervals. Estimation to access data from each resource (either inside or outside the network), estimation of time to process data coming from different resources to take final decision and time-out for the messages sent to services.
- **Cost:** This factor needs to be considered when buying from and selling electricity to the grid. Also, when accessing data from a third party such as weather service.
- **Reliability:** The correctness of data coming from different resources that are owned by third parties.
- **Availability:**
  - The availability of data from different resources owned by third parties, or such as the weather service and the demand forecast.
  - The availability of data from ESCO owned electrical network.
- **Performance:** is associated with the time required for completing tasks as discussed above.
- **Operating policy of ESCO:** The priorities and policies set by ESCO that need to be used for making different decisions about use of the resources owned by third parties.

6.7 Summary

The chapter explains the case study process and provides details about the use case that is constructed as a part of the case study. The use case represents an operational model of an energy control system for a small scale energy zone (SSEZ). The SSEZ electrical network configurations, operational goals and network data sources are discussed in detail.

Chapter 7

SOA Design

7.1 Introduction

The chapter describes the high level design created to model the small scale energy zone (SSEZ) control system that was discussed in detail in Chapter 6. Figure 7.1 highlights the part of the overall process of the case study that is covered in this chapter.

Conducting a design involves integrating case study information, the SOA model and software design knowledge. The design process consists of the tasks that need to be performed to construct the design. In this chapter, the term 'System' is used in a general way. While describing the design for the use case, we have also used the phrase '*service-based control system (SBCS)*'; which means '*the software system that is going to control the activities of the SSEZ*'.

7.2 Design Process

From a cognitive perspective "a software design method is a problem solving approach that focuses designer's attention on certain aspects of the design problem and attempts to facilitate the process of transforming problem requirements into a software solution (Kim and Lerch, 1992)".

A software design 'method' typically consists of three main components: representions, processes and a set of heuristics (Budgen, 2003). The representational

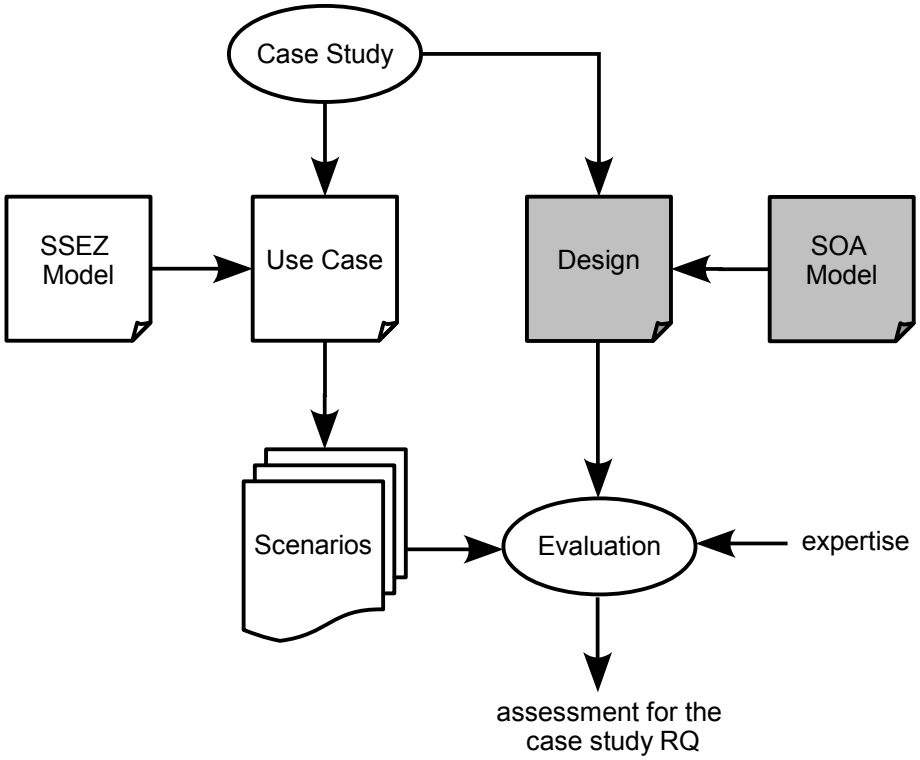


Figure 7.1: Case Study Design

part describes the design model through a set of notations (text, diagrams etc.), by making use of one or more *viewpoints*. Viewpoints are used to represent both the static and dynamic aspects of a system. The process part deals with the procedures and strategies are adopted by the designer to construct the design solution. The third part consists of guidelines that encapsulate knowledge of past experience and information related to particular problems or design techniques.

The design has been constructed by performing a set of activities, where these include both *decompositional*(a top-down-identification of functions) and *compositional* (a bottom-up-identification of entities) approaches (Budgen, 2003).

While designing, a designer has to transform an incomplete and ambiguous requirement specification into a high level system design which is expressed in formal or semi-formal notations (Guindon, 1990*b*). Therefore, process of design is regarded as a creative task, and there is no real systematic way of doing this (Visser and Hoc, 1990). Each new problem has its own level of complexity and

novelty, that may be a combination of new requirements in a familiar type of system or an unfamiliar type of system in an unfamiliar domain (Guindon, 1990*b*).

Usually, the design produced at the end is the result of multiple iterations of this process, as this is not possible to identify all of the features in the first round. Also the identification of one aspect may lead to the discovery of the need for a new service or change same features of the previous one. Therefore, there is no strict order for performing the activities discussed below and they are usually interleaved to some degree.

(a) **Identification of functional components** (functional decomposition of the system) - We partition the system into main functions through which the overall system functionality can be realised. This is based on the well established concept of 'separation of concerns' which is achieved through decomposition of system functionality. In structured design, functional decomposition is used to divide a complex problem into a number of sub-problems. This is categorised as a top down approach in which the problem is viewed from functional perspective rather thinking in terms of *noun* properties as we do in object oriented (OO).

In the SOA context, we consider a system as being a set of services that togather perform the system functionality. By considering these services as functional components we can initially assume them to be black boxes each taking certain input(s) and provides output(s). Expanding these as a white box, we can identify what functionality it provides and whether that can be sub-divided into sub-functions. These sub-functions have there own inputs and output. This makes the hierarchy of function and sub-functions which is called a functional refinement tree by Wieringa (1998). A functional refinement tree divide the function until we reach to the atomic functions that are transactions as shown in Figure7.2. By taking the example of the SBCS, we were able to identify that the system needs network states, it need to assess power in the zone, and evaluates different options to make final decisions.

As Wieringa (1998) has suggested, we can use this functional refinement tree with a Data Flow diagram to show the hierarchical decomposition. In the case of the SBCS, Figure 7.6 and Figure 7.8 correspond to the functional

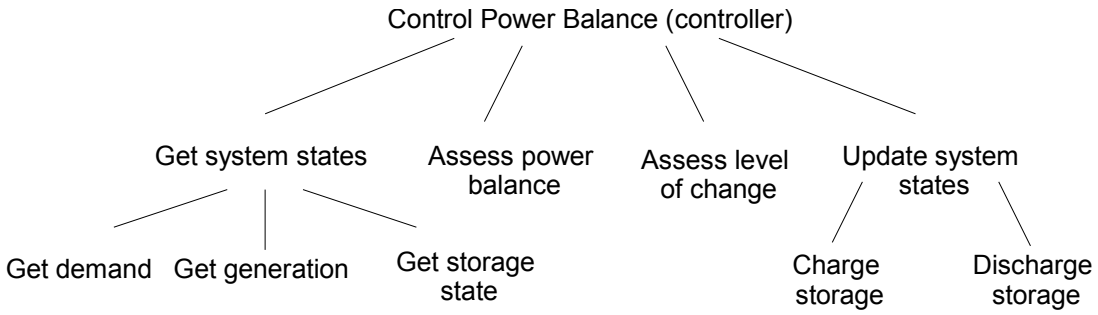


Figure 7.2: Part of functional refinement tree for SBCS

refinement tree. Functional refinement tree show system behaviour an increasing level of detail and can lead towards detailed design which is not in the scope of our discussion.

The other important criteria that need to be considered while identifying and defining functional components is *coupling* and *cohesion* which are defined by Yourdon and Constantine (1979). These are important because reusability is defined as being an intrinsic feature of an SOA. While identifying functional components and forming them into potential services these criteria need to be considered.

The goal of this activity is to identify the main functions performed by the SBCS. The result of this activity is a list of operations through which the overall system functionality can be realised.

(b) **Identification of potential services** - Identification of services is associated with the identification of functional components discussed in (a). A service may provide one or more functions, and may contain a logical grouping of functions within itself. This will also depend on the role of the service in the system. The inputs and outputs will determine the dependency of services on each other. This will help in composition of services and in the construction of workflow. Therefore the main goals of this activity are:

- Identification of services (the functionality that is reusable and will need to be obtained from a third party or that provided as service to others)
- Identification of service roles (what functionality the service will provide)

- Identification of inputs and outputs of these services (which will lead to identifying of dependencies and developing a composition flow)

These goals are similar to those used in object oriented design which is often considered a bottom up approach. However, in OOD the relationship between classes may also need to represent inheritance, which is not the case here.

The goals achieved in this phase will help to develop the possible interfaces for the services which can be realised through a class diagram. They will further aid to define the parameters needed for service invocation.

The output of this activity is a list of identified potential services, including their roles with possible inputs and outputs.

(c) **Functional traceability** - The functional components have been allocated to the services by means of a 'traceability table'. This allows mapping between functional components and identified potential services. The purpose of using functional traceability is to find out which function is provided by which service. This is represented in Table 7.5. This helps to identify service granularity and to determine how cohesive they are. Further, to identify which service is contributing to realise a particular functional component we have used a functional realisation table. The purpose of using these two tables is to:

(a) track which service is providing what functionality and which functionality will be available locally.

(b) identify services that are providing input to realise the functional component, where this is not part of their operations.

The result of this activity is the mapping of identified functional components and identified potential services which is represented by making use of tables.

(d) **Service interactions** - This describes how services interact with each other. The interaction pattern is described by using a table structure where services are ordered on horizontal and vertical axis, in the same order. The service interaction can be to get data or to provide a functional feature to other

services. For example, as interaction with an intermediary service can be to discover a particular service from the registry. In SBCS we have created a service 'get system states'. This intermediary service (or logical service) is responsible to collect system states from other services and to send a response to the controller. As this service is locally available means that it is a logical service not associated with any service provider.

The service interaction table makes the service interactions and dependencies visible. For example, to predict power generation, current generation and the information about the weather forecast is required, which will be used by prediction service. Further prediction service will be called by the controller, as mentioned in Table 7.7.

These interactions further help with identifying the type of messages the services will exchange. The representation of message passaging can be realised through the use of a sequence diagram as discussed in section 1.4.5. The interaction of services can be synchronous and asynchronous. This is an important design choice and depends on the requirement of the system. A combination of both can also be used, however, both have their own benefits and limitations. This choice will effect the way services are implemented. The important factor is that services are considered stateless and the scalability is considered an important factor in SOA design. However, the application domain has its own constraints that also effect the design choices.

The result of this activity is a table structure that represents the service interaction pattern.

(e) **Modelling static and dynamic behaviour through design representations**- As part of the design process, properties that describe the static and dynamic features of the software need to be captured and represented in order to formulate and explore the design model (Budgen, 2003). The viewpoint classification is taken from that of described in (Budgen, 2003). This classification consists of functional, behavioural, constructional and data modelling.

To represent the viewpoints we have made use of existing representational forms where possible, as there are no standards for SOA design. In the

representational forms we have tried to keep the level of abstraction the same. However, each representation has its own purpose and adds further level of information in the design.

In Table 7.1, the design activities are linked with the appropriate representational forms. The purpose is to analyse at what level each representation is useful.

Table 7.1: Summary of design activity

Activities	Form	Supporting Representations
Identification of functional components	List of functions	Data Flow diagram
Identification of services	Table	class diagram, component diagram
Functional traceability	Tables	-
Service interactions	Tables	Sequence diagram, activity diagram

In Table 7.2, we have mapped services features to different representational forms.

Table 7.2: SOA and Representations

Features	Representations
Service Interfaces	class diagram, component diagram
Service Dependencies	class diagram, activity diagram
Message Interactions	sequence diagram
Service ownerships and choices	component diagram
Process, control and data flow	DFD, activity diagram and flow chart

The viewpoints provide a classification of system features. We have used them for the selection of diagrammatical representations such as to represent static features of SBCS class diagram is used that comes under constructional viewpoint. Also we have discussed service interactions and now need to analyse how they will interact in the overall system. Further we need to decide where sequencing and parallel tasks will be required and how the control and data flow of the system will be organised.

In addition to the above discussed points, in the case of service based applications we have to identify:

- Whether services will be locally or remotely sourced. They will be logical or physical. By logical we mean that a service will not be associated with service provider. By physical service we mean a service that will be provided by a third party.
- Identify service providers and the type of contract used with them. There can be long or short term contracts with service providers. This will help to define the binding type (static binding or dynamic (Bianco et al., 2007)) and the use of the registry.
- The decision about the use of registry and its availability. There are different approaches to this. The registry can itself be a third party service, or it can be at the consumers' location. This decision effect the design of the system. By this we mean that if registry is located at consumer side then who will be responsible to update information about services and service providers. If registry is owned by a third party then consumer might need to pay for this service and the owner of the registry will be responsible to manage the information. These choices also depend on the application domain. In case of SBCS, the registry will be preferred to be at consumer end due to time constraints and the type of contracts made by ESCO (contracts that are long term such as six months or so).

We consider these decisions important enough to be mentioned explicitly, and as the detailed design is developed so these features need to be addressed. At a high level, we have represented them through the use of diagrams and discussed them accordingly.

7.3 Service-based Control System (SBCS) Design

The SBCS consists of different components and Figure 7.3 presents a brainstorming diagram that represents the main elements. The SBCS collects information about its network resources that is recorded as system states. It needs market buy and sell prices to import or export energy to the national grid. For the prediction model, it needs demand and generation prediction values. To maintain system history and to be used to add further functionality like asset conditions, the SBCS requires weather data. The Controller is responsible for executing the overall process of the SSEZ control system, and uses priorities (the preferences discussed in Chapter 6) as part of the operating policy of the system.

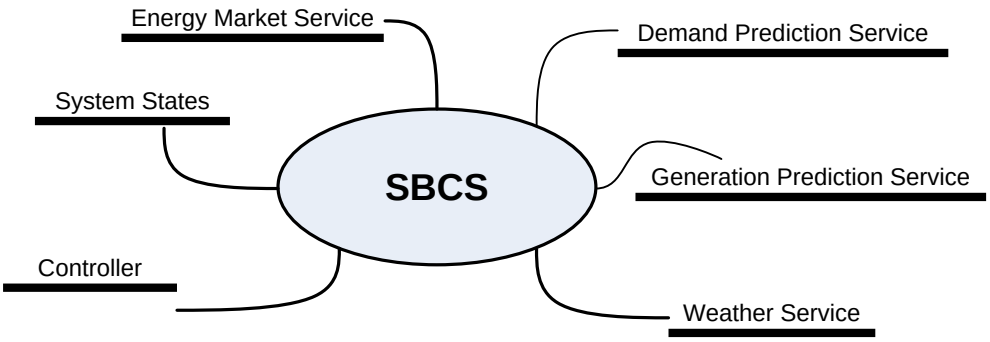


Figure 7.3: Abstract view of SBCS

Figure 7.4, presents an abstract system architecture for the SBCS. The SSEZ network data is recorded as part of system states. The information about current and predicted weather condition in the zone is also collected. This information is used by the component that assesses system states and the prediction model. The component responsible for assessing states makes use of constraints and priorities along with the information about current market buy and sell prices. The current situation for power generation and demand in the zone is assessed. Further, information coming from prediction model is used to assess future conditions in the zone. The information about the current and future condition along with the decisions made by the controller are also recorded as part of system history.

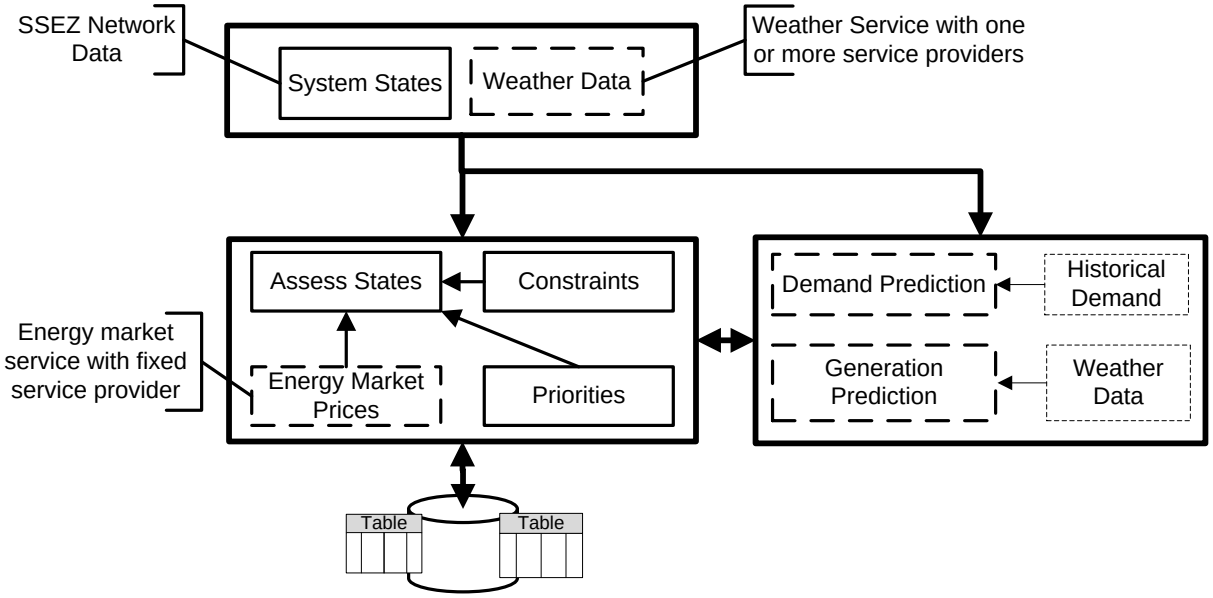


Figure 7.4: System Architecture Abstract View

In the next section we discuss the SBCS design according to the activities defined in section 7.2.

7.3.1 Identification of functional components

The SBCS consists of nine main functional components as shown in Table 7.3. These functions are at the first level and they contain sub-functions. We have kept them at this level here as we have to identify what functionality will be local to the system and which will be accessed through third party.

In the next iteration we can divide these functions into sub functions, for example, F1 contains three informations sources (generation, demand and storage). These sub-functions can also take the form of services, depending on their use. Similarly F4 and F5 provide a high level of abstraction. Dividing them into sub-functions will include further level of details. Here, we can make use of functional refinement tree.

Table 7.3: Functional Components (modules)

Functional Components	Roles
Get system states (F1)	SBCS states from three different sources (generation, demand and storage).
Assess power balance (F2)	To check balance in current demand and generation.
Get weather forecast (F3)	Get weather forecast for SBCS.
Predict demand (F4)	Predict demand based on current and historical data.
Predict generation (F5)	Predict generation based on weather forecast and current generation status.
Assess level of change (F6)	Check different options to assess the type of change required to maintain energy balance.
Get market price (F7)	Get energy market price for SBCS.
Update system states (F8)	Take action by changing system states.
Update system log (F9)	Update data in the SBCS system database.

7.3.2 Identification of potential services

The services that have been identified as being able to provide the functionality identified in Table7.3 are shown in Table 7.4. The role of each service with possible inputs and outputs is also described.

The controller that executes the process is not considered as a service because it is not offering a service outside of its own system. Instead, it is consuming the services listed in Table 7.4. Therefore, F2 and F6 are part of the controller and are not represented as a service. Further, service (S_{HD}) responsible of providing historical demand is identified as part of demand prediction service.

This table is helpful in identifying service interfaces. The class diagram that can help with providing interface information is discussed in section 7.3.7.

7.3.3 Functional traceability

The traceability table, Table 7.5, represents the role that each service plays in providing the functionality listed in Table 7.3. A cross indicates the involvement of a service in a function and its absence means that the service plays no role in the realisation of that function. The functional components, F2 and F6 are not

Table 7.4: Service Role, inputs and outputs

Services	Roles	Inputs	Outputs
Service to get generation output (S_G)	Provides data from wind turbines.	-	generation output, wind speed
Service to get demand (S_D)	Responsible for providing current demand data.	-	energy consumption data
Service to get historical demand (S_{HD})	Responsible for providing historical demand data.	-	energy consumption data
Service to get storage status (S_S)	Provides current state of charge (SOC) and storage status (SS) for storage unit.	-	SOC, SS
Service to predict demand (S_{PD})	Provides demand prediction.	current demand, historical demand, weather data	demand prediction data
Service to predict generation (S_{PG})	Responsible for providing generation predictions.	current generation, weather data, location	generation prediction, wind speed
Service to get weather forecast (S_W)	Provides weather data (current and forecast).	location	wind speed, temperature
Service to provide energy market price (S_M)	Responsible for providing the energy market price.	-	buy and sell price
Service to maintains system log (S_L)	Updates data in the SBCS database.	system states, weather data, market price	-

provided directly by services and are part of the controller, therefore no cross is included in these two columns.

Table 7.5: Functional Traceability

	F1	F2	F3	F4	F5	F6	F7	F8	F9
S_G	×								
S_D	×								
S_S	×								
S_{PD}				×					
S_{PG}					×				
S_W			×						
S_M							×		
S_L									×

In Table 7.6, the services are mapped to functional components in order to represent how the functionality of these functions is realised. These functions are not part of a service, rather they use information from the services to perform their tasks.

Table 7.6: Functional Realisation

	F1	F2	F3	F4	F5	F6	F7	F8	F9
S_G		×				×			
S_D		×				×			
S_S		×				×			
S_{PD}		×				×			
S_{PG}		×				×			
S_W				×	×				
S_M						×			
S_L									

7.3.4 Service Interactions

In Table 7.7, services are listed along the x-axis and the y-axis in the same order, and the table shows where services interact with each other, including the controller (represented by C). The cross indicates service interactions and absence means no direct communication between services. We found this representation

useful in terms of identifying the dependency and interaction of services with each other. This dependency is further shown in the class diagram in section 7.3.7. The interactions in the form of data and control flow are discussed in section 7.3.9 and messages among services is represented in sequence diagram in section 7.3.10.

Table 7.7: Service Interactions

	S_G	S_D	S_S	C	S_{HD}	S_{PD}	S_{PG}	S_W	S_M	S_L
S_G				×						
S_D				×						
S_S				×						
C	×	×	×			×	×	×	×	×
S_{HD}						×				
S_{PD}		×		×	×					
S_{PG}	×			×			×	×		
S_W				×		×	×			
S_M				×						
S_L				×						

7.3.5 Modelling static and dynamic behaviour through design representations

Here, the viewpoints described in the design process section are discussed in more detail. Table 7.8 provides information about the purpose of using a particular representation, and the viewpoint that it is intended to provide.

- Both DFDs and activity diagrams are used to show the functional aspects of the SBCS. A DFD provides a 'big picture' of the SBCS and is used to aid functional decomposition of the system. The activity diagram provides more detail about the processes.
- An activity diagram is particularly useful for representing how services interact with each other in order to perform a specific task or to realise a business process. During service composition, workflow is generated that describes the sequence in which services will be assembled and executed. An activity diagram can be used to describe this.

Table 7.8: Purpose, Representational forms and Viewpoints

Purpose	Representation	Viewpoint
Problem oriented view of system with its inputs, outputs and processing elements	Data flow Diagram	Functional
Service operations and relationships	Class Diagram	Constructional
System components and Service interfaces, relationships, and service providers	Component Diagram	Constructional
System flow with functional components internal and external to the system, sequencing and ordering of activities	Activity Diagram	Behavioural/ Functional
Interaction and order of interaction among services over time	Sequence Diagram	Behavioural
Overall system behaviour, interaction with services and decisions by the control	Flow Chart	Behavioural

- A sequence diagram is selected to represent SBCS behaviour through message passing.
- The class diagram serves two purposes. It can provide the conceptual decomposition of the system into services. And can also be used for service interface information and their relationship with each other.
- The component diagram is used to show how system components and services are interacting with each other. The interfaces that are provided and required by them. This also helps to present the physical and logical view of services.
- A flow chart is used to represent system behaviour. The decisions made by the controller and the interactions with services in a workflow are shown through scenarios.

Notations used in these diagrams are available in Appendix H.

The use of each representational form for the SBCS is discussed in the following sections.

7.3.6 Data flow diagram (DFD)

A DFD introduced by DeMarco (1979) provides an abstraction of the overall system and is considered to be useful for analysis. In structured analysis, the DFD is a well established way of representing processes, external entities, data flow and data stores of the system under development. It provides the concepts and notations in a simple manner which can be considered a major element of its acceptability. Also it provides modularisation in top-down manner, support hierarchical structure and the symbols used are distinct that help in managing the diagrammatical complexity (Moody, 2009).

In Figure 7.5, a DFD has been constructed to represent an abstract view of the SBCS by presenting its processes, the entities it communicates with, and the data it stores. The oblong box is used to represent external entities which are the distributed resources of the SBCS that take the form of service providing inputs. The circle represent the SBCS process. The arrows show the flow of data from the entities to the central process. The output from this process is stored in the data store, represented by two parallel lines.

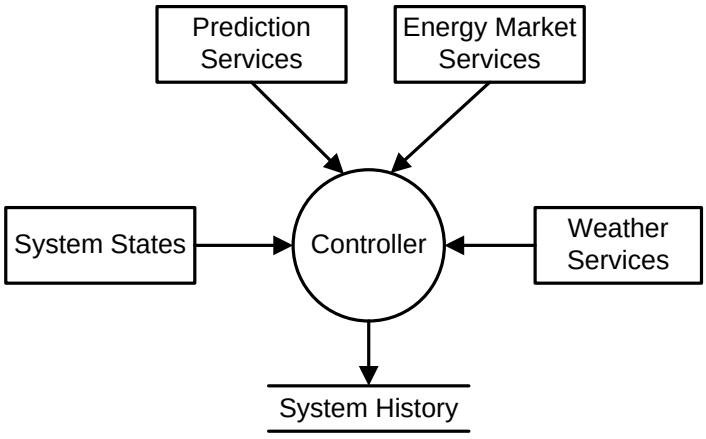


Figure 7.5: Data Flow Diagram (DFD) showing an abstract view of the SBCS

In Figure 7.6 the processes that interact with external entities are shown separately. For example, the arrow pointing towards the *change system states* entity represents decisions made by *Controller*.

Figure 7.7, provides further details about the interactions of processes with entities, togather with some values. Three system states are shown in this di-

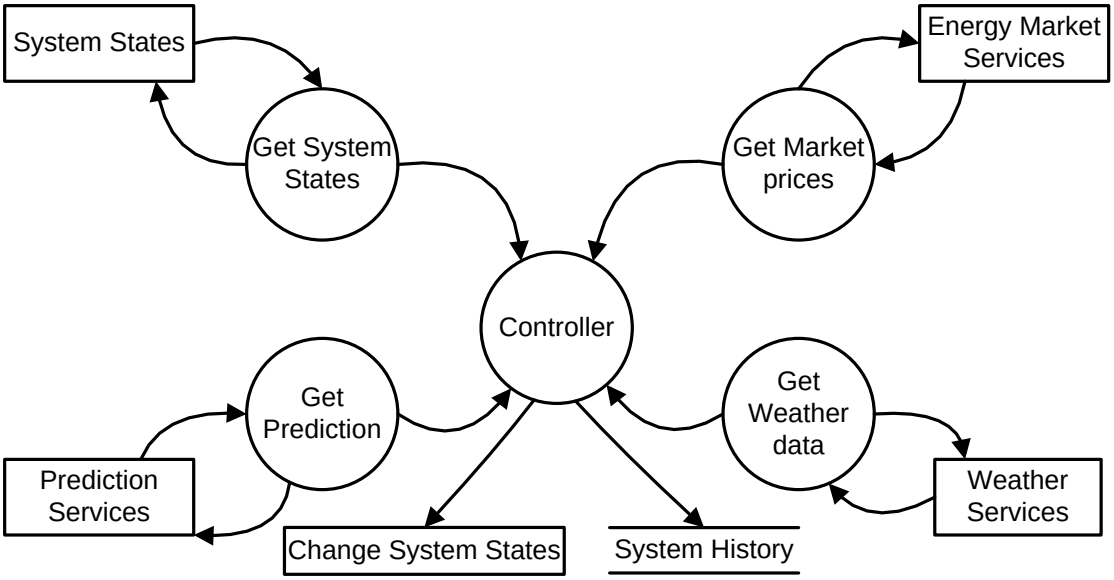


Figure 7.6: Data Flow Diagram (DFD) of SBCS with further entities and sub processes

agram that model energy consumption, energy generated and energy stored in the zone. Current and predicted buy and sell prices are taken from *Market Service*. The *Weather service* provides current and future weather data. *Demand* and *Generation prediction services* provide the prediction values to evaluate the likely future condition of the SSEZ.

Figure 7.8 provides detail about the *Controller* sub-processes. It collects data from different resources including system states and data from different services. In the next process, it evaluates power condition in the zone by comparing demand and generation values. Based on the current power situation, it considers different options available to handle the situation, for example importing or exporting energy to the grid in the case of energy deficit /surplus. Finally it updates the system states, depending on the decisions it has made, for example charging or discharging the storage.

7.3.7 Class Diagram

The purpose of using a class diagram is to represent the static features of the SBCS. Each element in a class diagram is usually partitioned into three sections

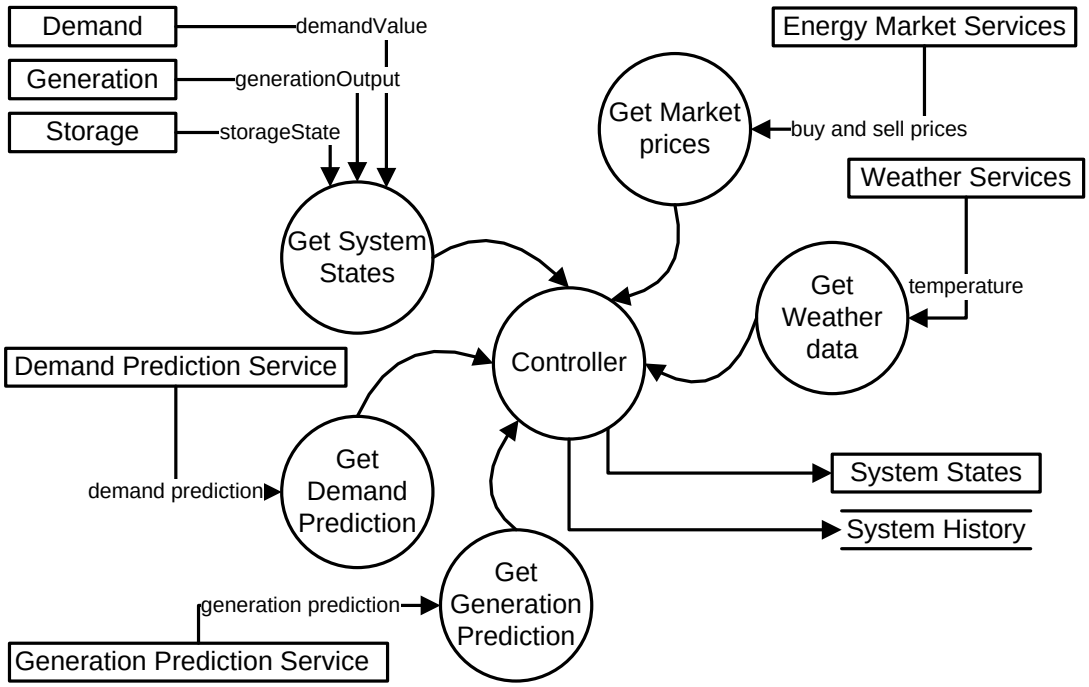


Figure 7.7: Data Flow Diagram (DFD) with details about entities and data

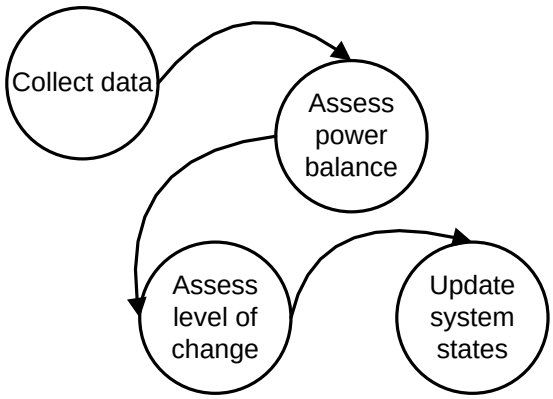


Figure 7.8: Data Flow Diagram (DFD) of Controller sub-processes

that contain the class name, its attributes and its operations. In Figure 7.9, a service model for the SBCS is represented using class diagram. The operations that services offer and perform are listed in the rectangular boxes. The interactions among services which show their dependencies on each other are represented through doted lines. Stereotypes have been used to give more meanings to service model.

In the *Controller* the operations are performed internally, while other services offer interfaces designed to be invoked. For example, the *GenerationPrediction Service* takes data from the *Weather Service* and *Generation Service* and provides predictions to the *Controller*. The *DemandPrediction Service*, invokes the *Demand Service* and provides demand prediction to the *Controller*. The *System-Log Service* is in turn the service that is dependent on the *Controller*, which it accesses to get the data needed to maintain the system history.

7.3.8 Component Diagram

A component diagram has been used to represent the interactions of services, their interfaces, and any dependency between system components. This also provides information about the interfaces that are offered by services and the one required by system components.

We have used a component diagram to represent:

- dependency among services and SBCS components,
- the interfaces provided by service and used by components, and
- to represent information about service providers. This also represent where choice for more than one service provider is available. This information is helpful when we have a mix of fixed and multiple service providers. It also helps to decide that either there is a need to use the registry in the case of the availability of fixed service provider or whether it should be treated as static binding. We consider it an important design decision as this will add or remove the processing of the searching service from the registry.

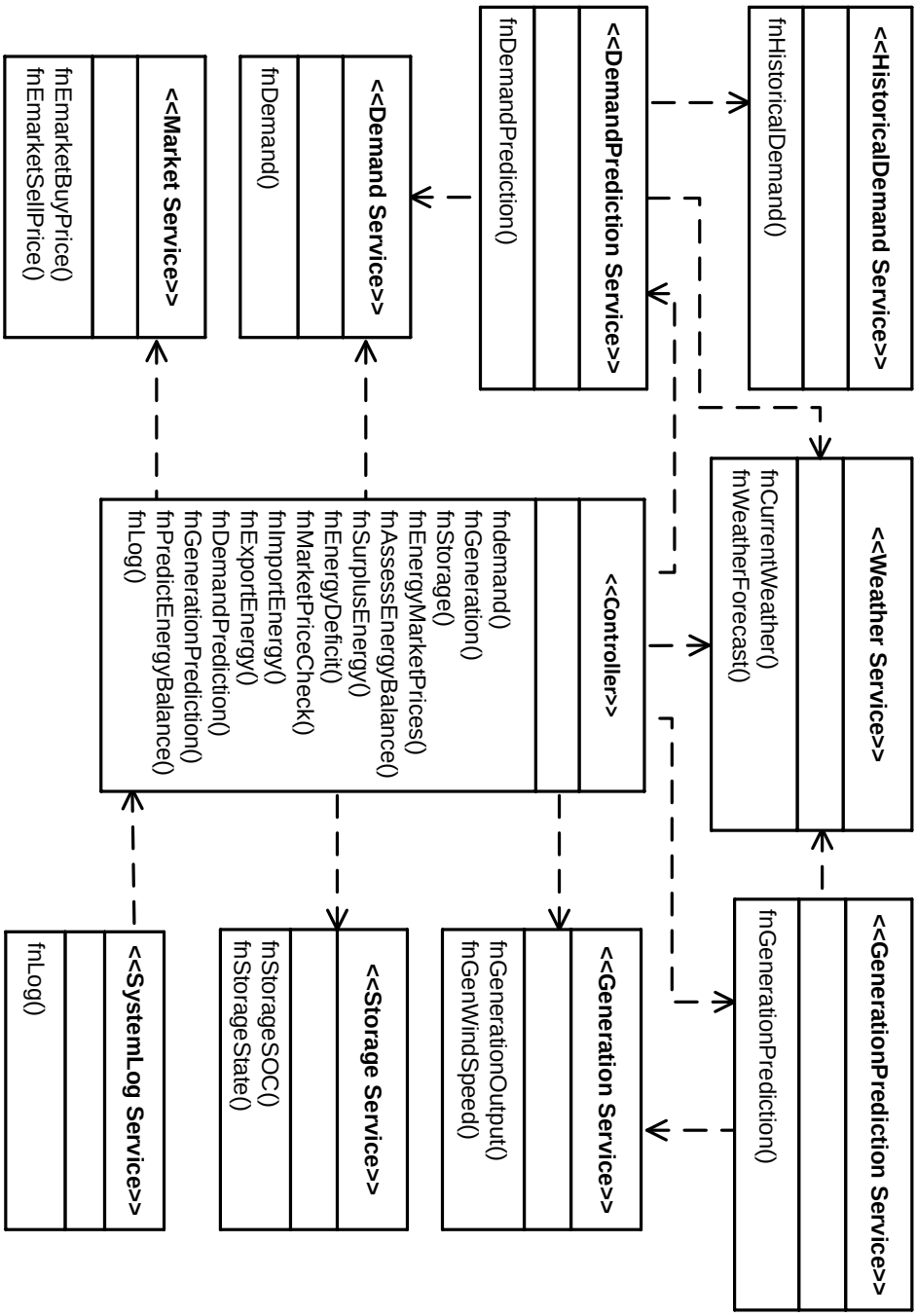


Figure 7.9: Class Diagram for service dependencies and operations

In Figure 7.10 we have divided the SBCS into two main components: the controller and the prediction model. These two components are internal to the SBCS and use the other services.

The *Energy Market Service* has a fixed service provider. This service interacts with controller to provide current market buy and sell price. The *Weather Service* that interacts with controller has two possible service providers. This means that the weather service can be obtained from multiple service providers. For this reason we need to add registry to the SBCS functional components. The registry component will maintain the information about the weather service providers and the non-functional information that includes the time and cost of using these service. Further the weather service used here is also providing current weather condition to the controller. Each time the controller accesses this service there is the possibility that a different service provider is selected.

The services interacting with the prediction model also have multiple service providers. It is possible that the *Weather Service* that provides current weather to the *Controller* will be different to the one that is used by the *Generation Prediction Service* and the *Demand Prediction Service*. This means that each time the *Controller* executes its process, a different set of services might be selected. The dotted arrow going from the prediction model to the controller represents the decision that this component is executed as part of the controller component.

7.3.9 Activity Diagram

An activity diagram provides a workflow-oriented view of a problem. It represents both functional and behavioural aspects. Different activities are organised so as to show system flow. This can also be used to represent a business process. The diagram helps in presenting the decisions, functions, parallel and sequential activities involved in a system. It can also be used to represent control and data flow among different activities. This is also useful in presenting the different scenarios.

We have used an activity diagram to represent the overall flow in SBCS, and for different scenarios that show behavioural aspects. The rectangles used in the diagram are for functions, a diamond is for a decision, the bar representing join

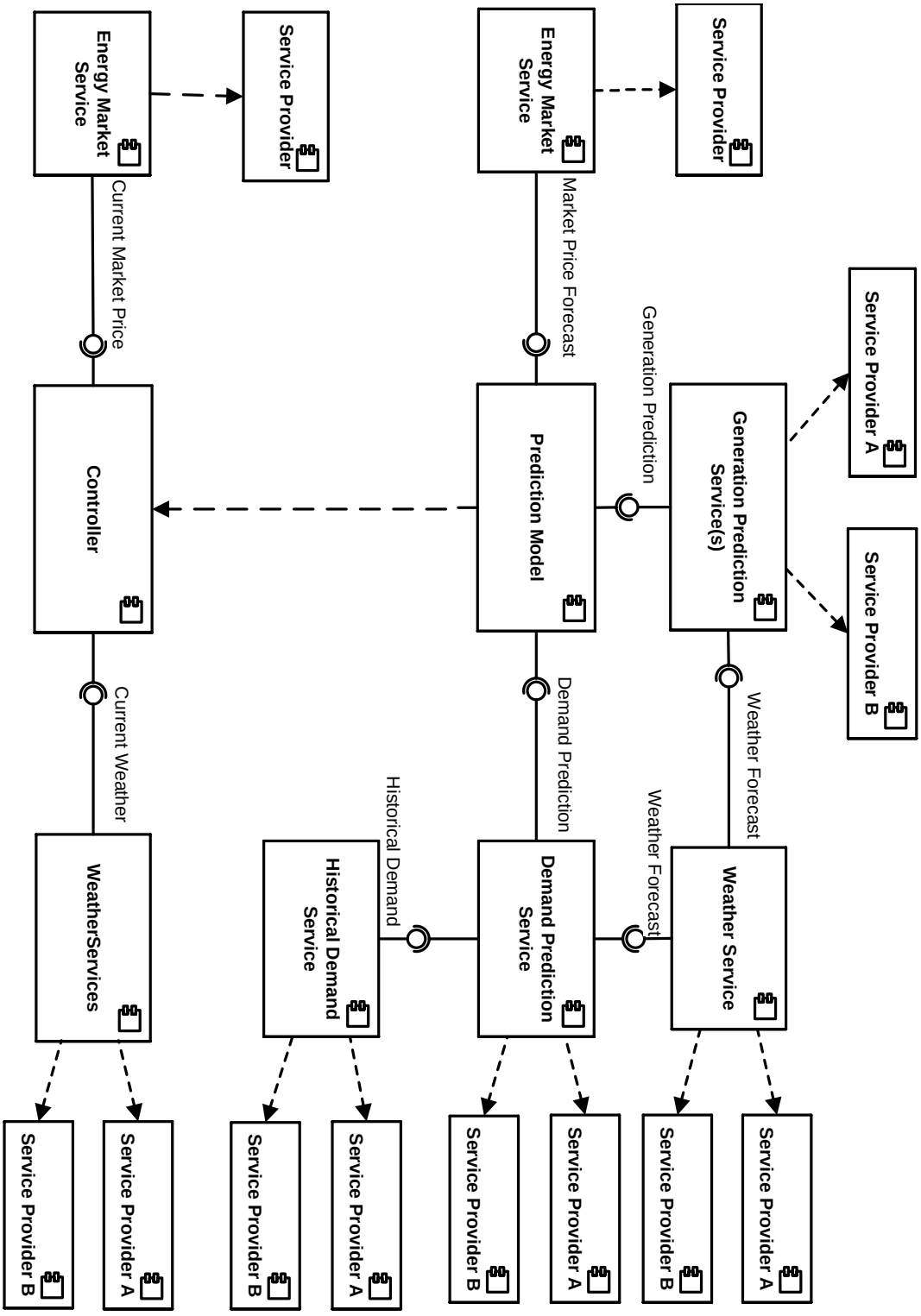


Figure 7.10: Component Diagram for SBCS

and forks are to help in presenting parallel activities.

In Figure 7.11, the SBCS main flow is represented. The process starts by accessing the SBCS states (consumption, generation and storage), and the weather forecast data. The SBCS states are used to assess the current power balance in the SSEZ. If there is no change needed from the current condition, then the system updates the log and process ends. If there is difference from the current state in the predicted demand and generation, the process evaluates system states and market prices to check possible options available to take appropriate actions. At the end, the process updates the system states and the history records.

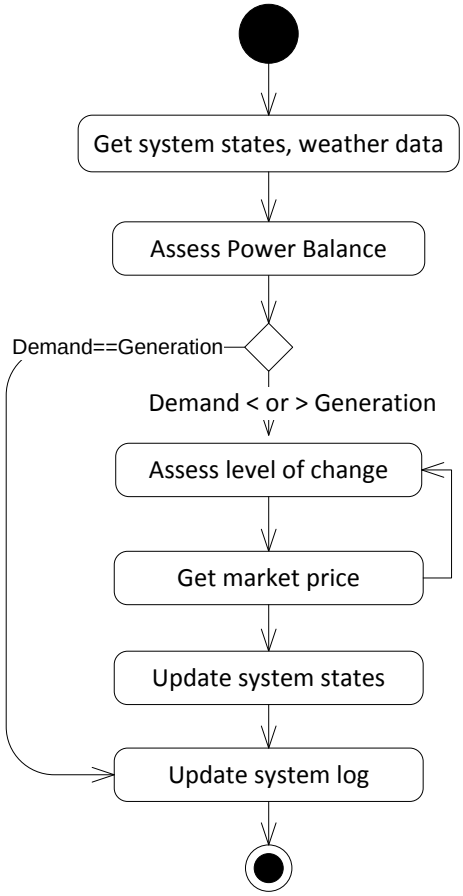


Figure 7.11: Activity Diagram showing SBCS main system flow

In Figure 7.12 further details are provided about the activity that assesses the degree of change. The process is the same as described previously. To assess

change, the storage condition is checked to see it is possible to make use of the energy available within the SSEZ. The *Market Service* is invoked to get current buy and sell prices. Depending upon storage condition, current market prices, the current condition of generation and the operating policy set by ESCO, the SBCS can decide which course of action it has to take. So, in the case of surplus energy, the SBCS has to decide about making money in the market by exporting energy or whether it should charge its storage or both. Similarly, if there is energy deficit in the zone, it has to evaluate the condition and decide upon an action from the possible options available.

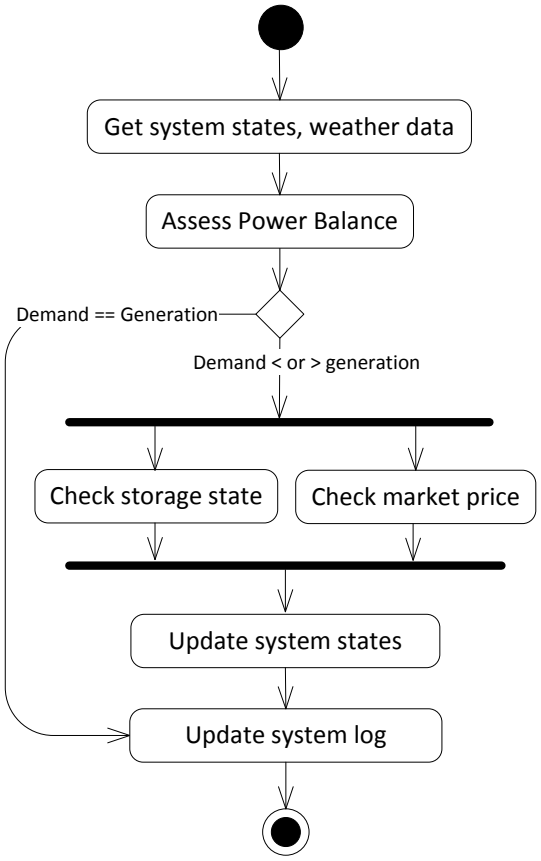


Figure 7.12: Activity Diagram providing detail of available options

Figure 7.13 describes the prediction model used in the SBCS, which is a sub-process of overall system. This flow shows the sequence of activities that occur when the SBCS decides to use prediction services provided by service providers.

The main flow remains the same as shown in 7.11 apart from invoking the prediction services and assessing the future condition of the SSEZ. To get demand prediction, current consumption is provided to the *DemandPrediction Service*. The *GenerationPrediction Service* will then use the current generation data to provide a generation forecast, along with the details of the weather forecast. The *Market Service* is used to get the market price forecast.

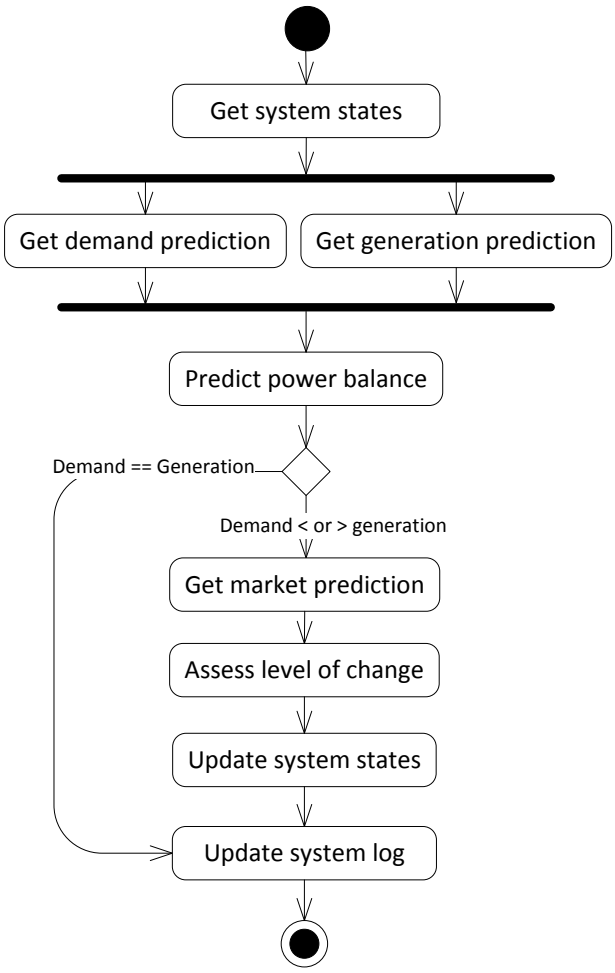


Figure 7.13: Activity Diagram with predictions scenario 1

In Figure 7.14, further details has been added to the prediction model. In this case, the historical demand data and current energy consumption values have been used to predict demand along with the weather forecast. The historical demand can be the SBCS system itself or can be purchased from third party.

For generation prediction, current generation output and weather forecast data is required. The *Energy Market Service* is invoked to get future buy and sell prices. The rest of the flow is the same.

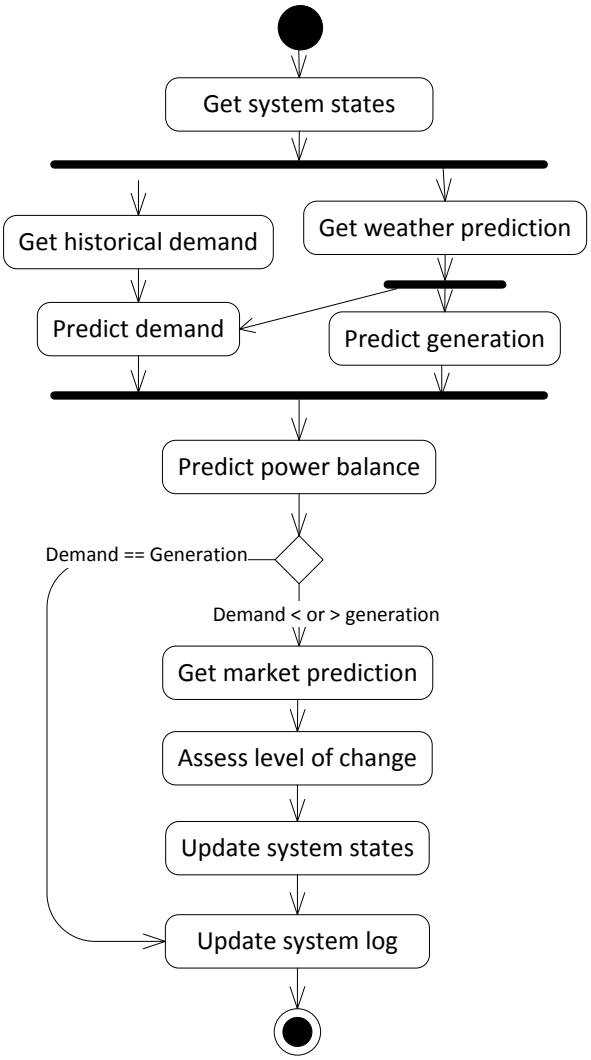


Figure 7.14: Activity Diagram with prediction scenario 2

Figure 7.15 represents the process of getting weather data from the *Weather Service*. The swimlanes used in the diagram are to separate the system and weather service provider. The registry service can be used to maintain *Weather service* information. After selecting the appropriate service it will invoke the weather service to get current and forecast weather data. There can be multiple

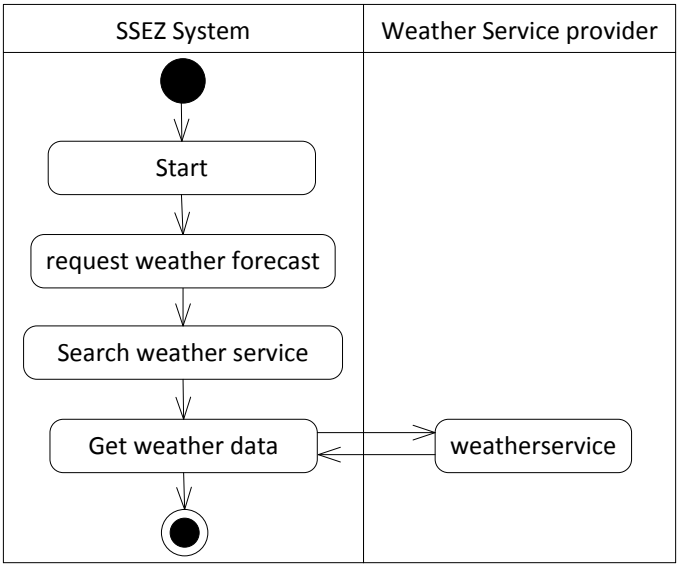


Figure 7.15: Flow to access weather data

Weather Service providers and the SBCS operating policy may include short term contracts with them. Decision about the selection of which service to use may also include the cost, and granularity of data.

The overall flow of the SBCS is shown in Figure 7.16. The process to get *Weather Service* has already been discussed. In case of the *Energy Market Service* there is a fixed service provider. The prediction model is represented as an activity in the overall flow as the details are represented in Figure 7.14.

7.3.10 Sequence Diagram

The purpose of employing a Sequence diagram is to represent the communications between of services over time. The different services are organised across the X axis and the messages sent and received among services are along the Y axis. The lifeline represents the existence of the service over a period of time and the thin rectangle shows the period of time during which a service is performing an action.

Figure 7.17 provides an initial view of how message passing is recognised among the services. The *Controller* sends requests to the generation, demand and storage services to get data. A self loop represents where *Controller* uses

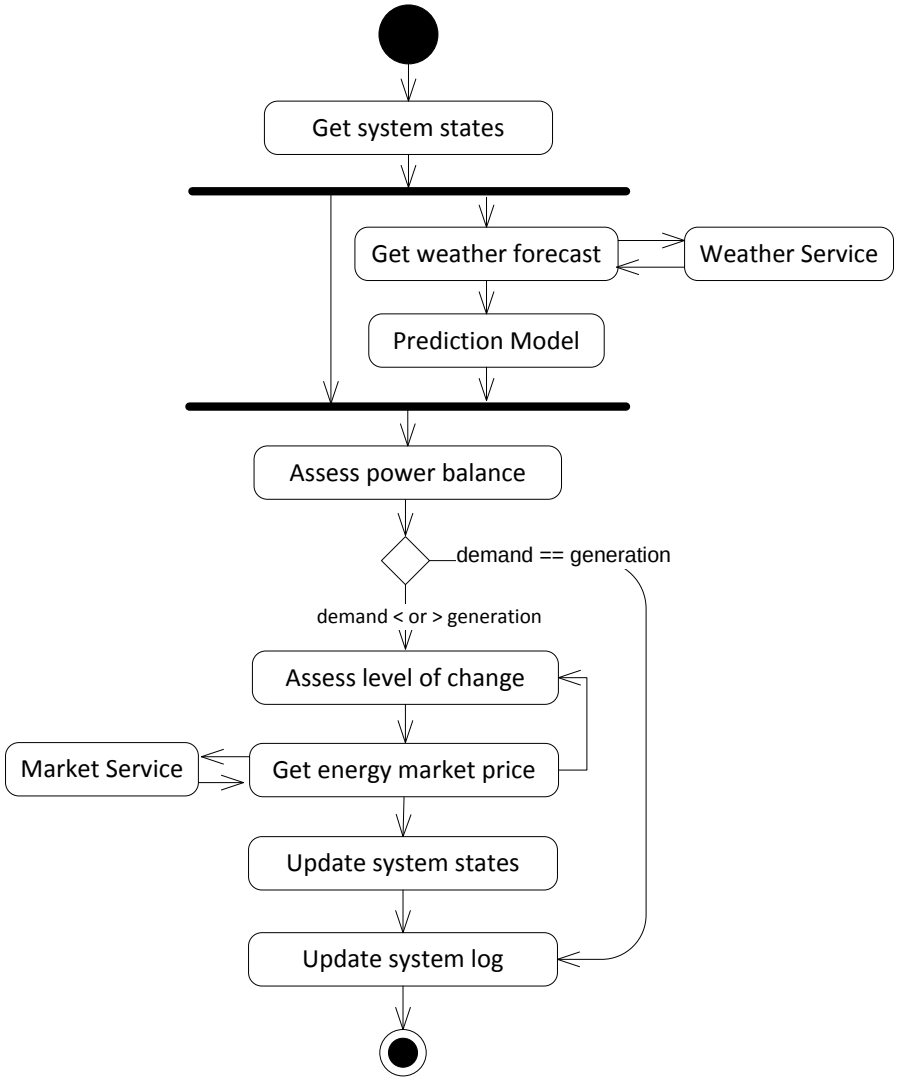


Figure 7.16: Activity Diagram to show overall SBCS flow

the input to assess the power balance. It further sends a request to the *Weather Service* to get a weather forecast. On the basis of this weather forecast, togather with the demand and generation prediction data, the *Controller* will predict the power balance, which is shown in Figure 7.18. A request is sent to the *Energy Market Service* to get current and predicted buy and sell prices. The *Controller* checks the level of change required in case of any variation in energy balance. Finally it communicates with the *SystemLog Service* to log system states. The*DemandPrediction Service* and *GenerationPrediction Service* are not shown in this diagram.

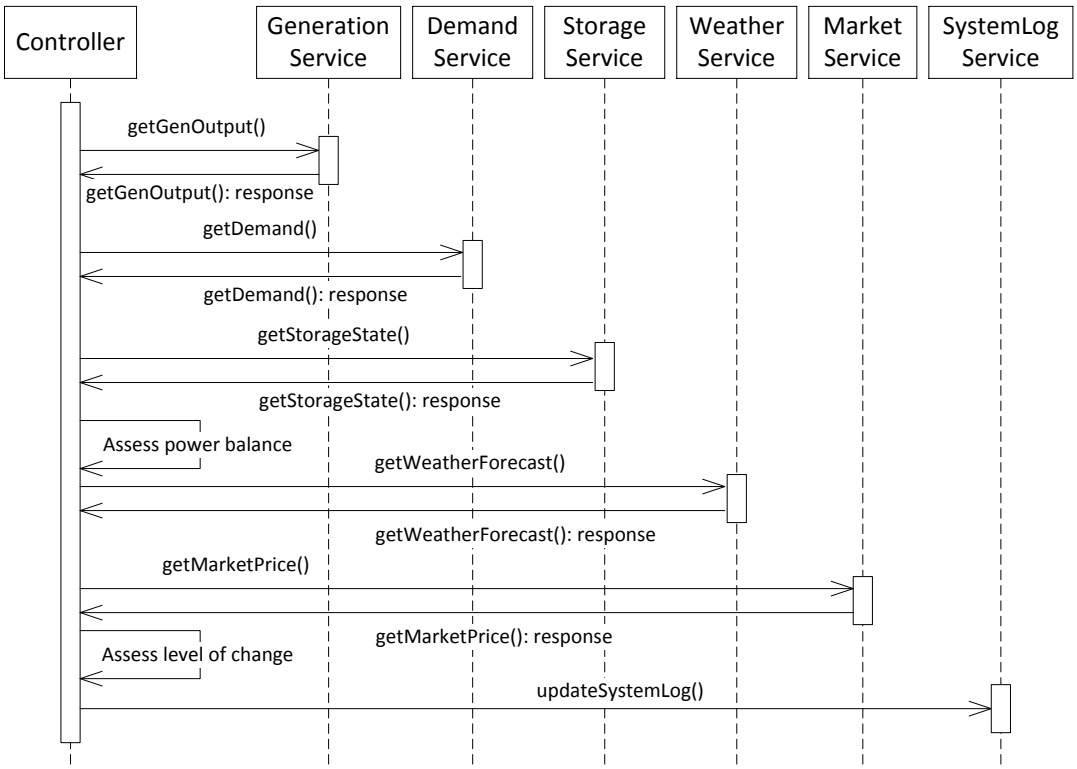


Figure 7.17: Sequence Diagram with initial system view

In Figure7.18 new services are introduced to this model. The *SystemStates Service* collects information about system state from the *Demand Service*, the *Generation Service* and the *Storage Service* and provides this to the *Controller*. The *Controller* then communicates with the *DemandPrediction Service* and provides it with the current demand level in order to get the demand forecast. For

generation prediction, the *Controller* provides generation and weather forecast to the *GenerationPrediction Service*. Once the demand and generation predictions are received, the *Controller* uses this information to predict the future power balance. It further invokes *Market Service* to get market forecast prices.

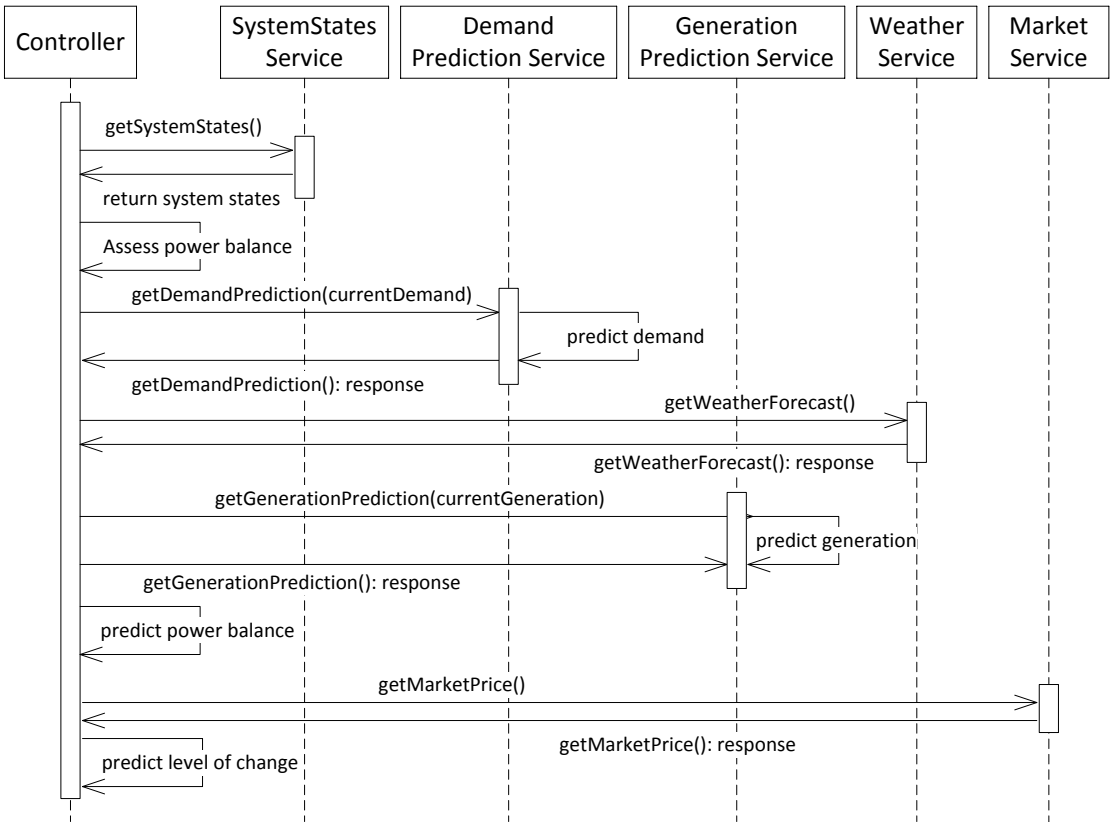


Figure 7.18: Interactions among controller and prediction services

Figure 7.19 provides details about the interactions between the *Controller* and those services that get data from external resources. The *Weather Service* interacts with the registry to get information about available weather services; and after getting the response; it invokes a weather service to get forecast data. The registry can provide more than one weather service, and so the *Weather Service* has to select the appropriate service to get data. In the case of the *Market Service*, there is only one external service provider, hence the service will be accessed directly.

Figure 7.20 describes the overall view of service interactions. The *Demand,*

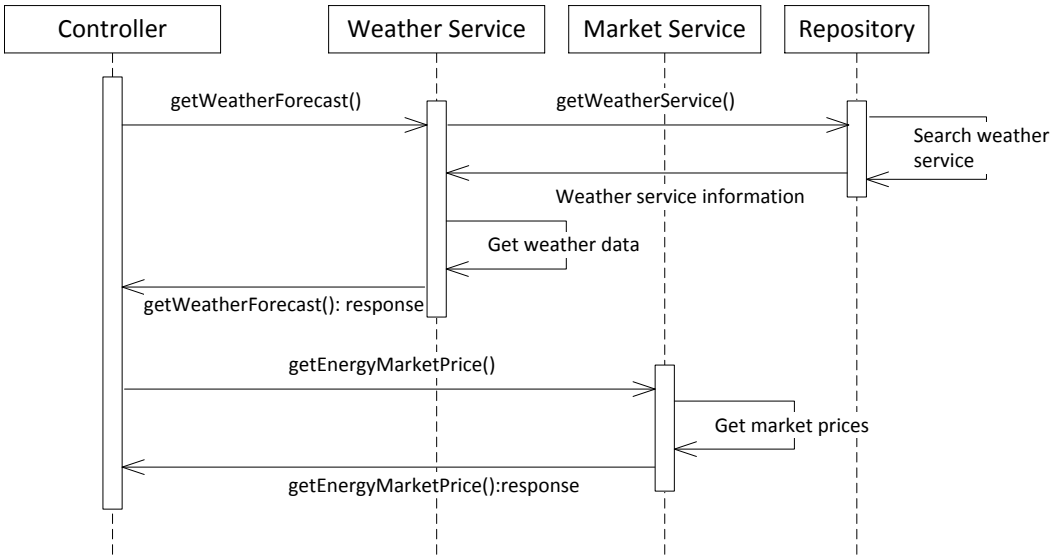


Figure 7.19: Interactions among controller, weather and market services

Generation and Weather service each have different service providers. Therefore, we have used a registry. There is the possibility that each service will use a different set of parameters and provide a different level of detail in its return.

7.3.11 Flow Chart

The Flow Chart used is to model the decisions that the controller makes in different conditions. It makes use of scenarios to represents different system states, and constructs workflow that shows overall system process. Therefore, we find flow chart an appropriate choice to be used here.

Two flow charts are constructed. Figures 7.21 and 7.22 and Figure 7.23 represent the current condition in the SSEZ. The second flow chart is constructed to represent the scenario based on predicted demand and generation values. This also shows that where changes will be made in current system states. This is represented in Figure 7.24 and Figure 7.25.

7.3.11.1 Scenario for assessing the current energy balance in the SSEZ

The process starts by getting information about different system states. It checks for the availability of the storage unit. If this is present then the controller will

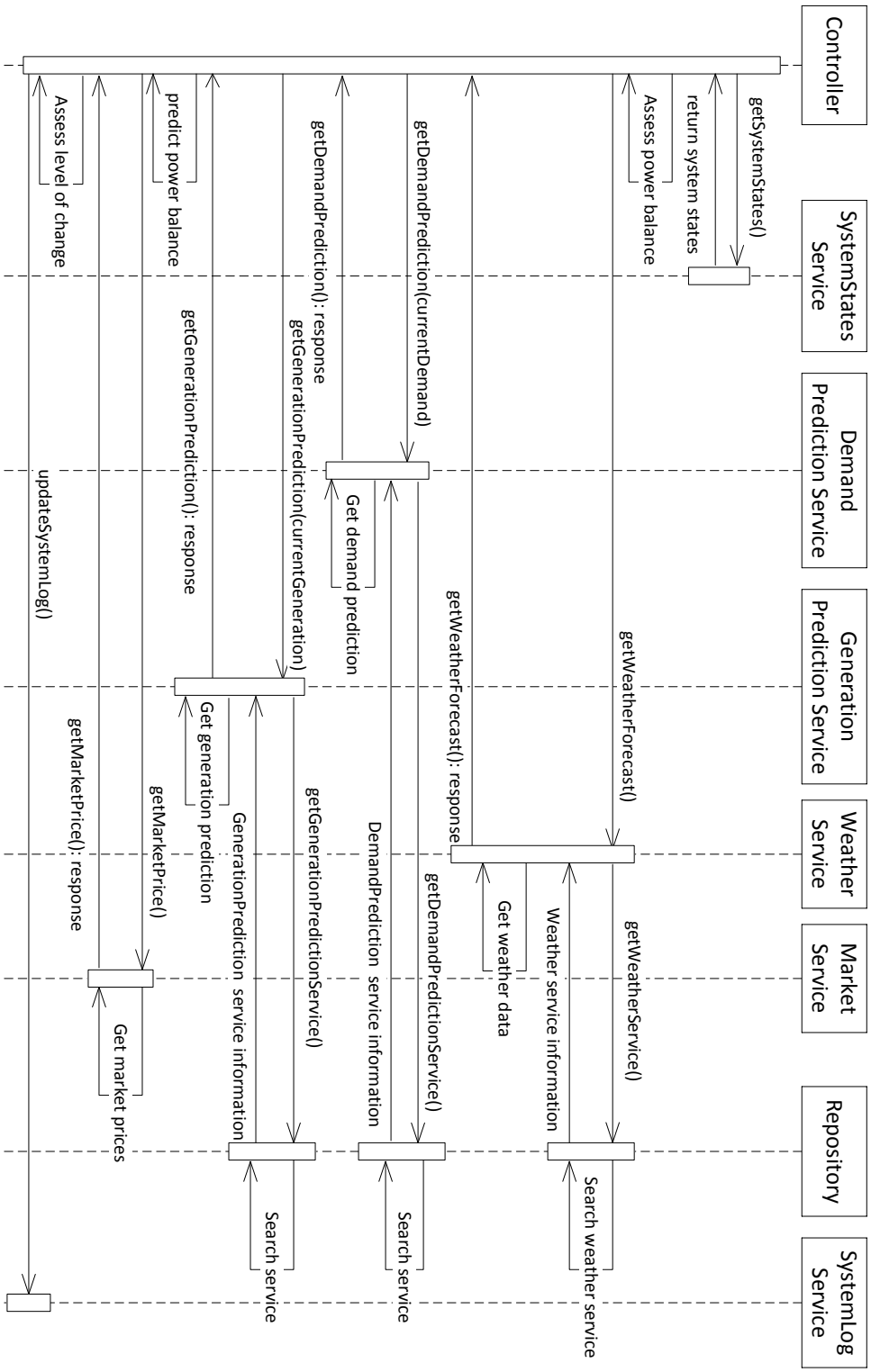


Figure 7.20: Overall system interaction view

make the task of adjusting the state of the storage unit to be its first priority before importing or exporting power to the grid.

The controller checks the power balance as shown in Figure 7.21.

- If it is balanced, the storage state of charge (SOC) is checked for its boundary conditions.
- If current demand is less than generation then branch **A** represents the flow.
- If current demand is greater than generation, it follows branch **B**.

(a) **Branch A: Demand is less than generation (Figure 7.22)**

- Calculate current surplus energy in the zone.
- Check storage state of charge (SOC).
- If storage is charging, and has reached the upper limit then stop this. Get market price and export energy. (Here the calculation can help ESCO to evaluate how much green energy is retained in the SSEZ and how much can be exported to grid.)
- If storage is charging and its SOC is equal to the lower limit, then calculate power it needs to reach the upper limit and find if there is still surplus energy available. If so, export that to the Grid. For this, the energy market service will be invoked to get the current market sell price. (Here, the function that calculates the energy sold to grid will help ESCO to calculate the money it has made in the market.)
- In any case other than the ones listed above, the controller will decide to charge its storage and export energy at the same time, or just to make money in the market by exporting all surplus energy. At this point the ESCO operating policy sets the priority for the controller to consider. This option is represented through the solid black arrow in Figure 7.22.
- If storage is stopped or discharging, in both cases the SOC limits will be checked and decisions will be made about energy export. As the scenario contains the information about storage state of charge, therefore we have represented these states through separate flows.

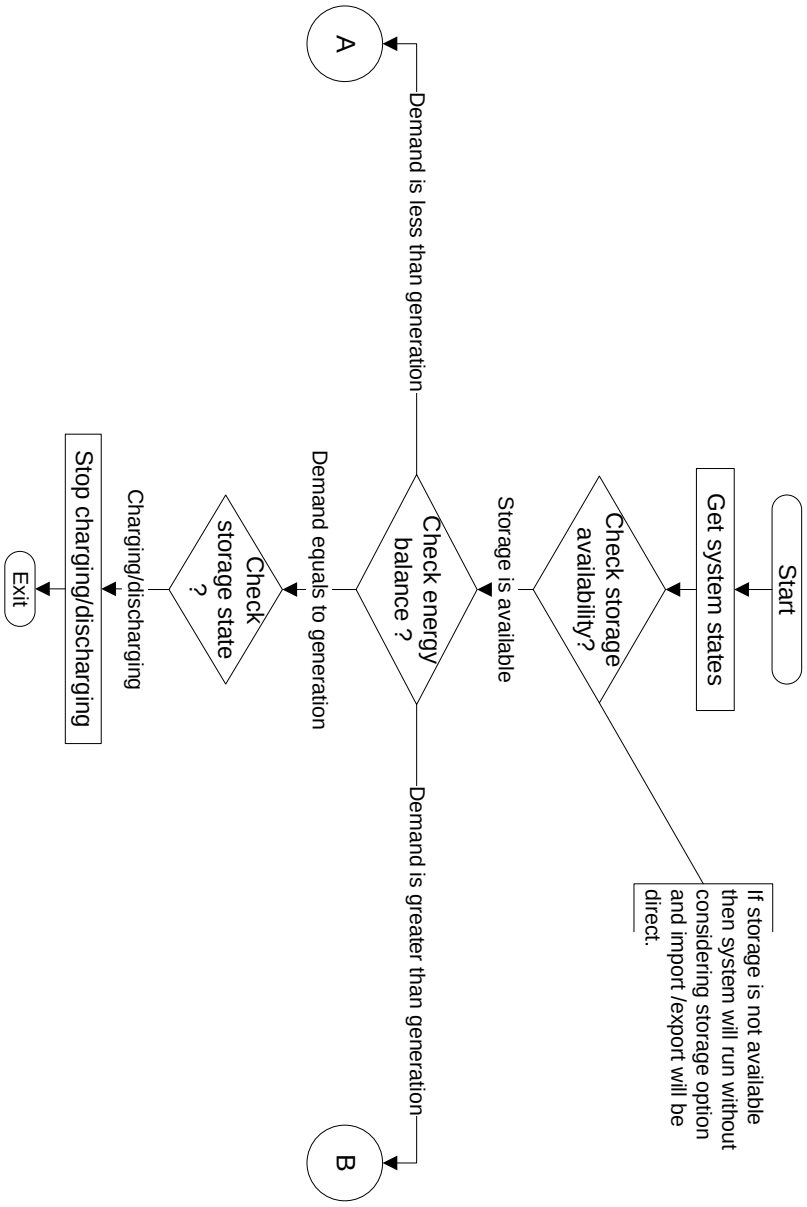


Figure 7.21: Flow Chart Main Structure

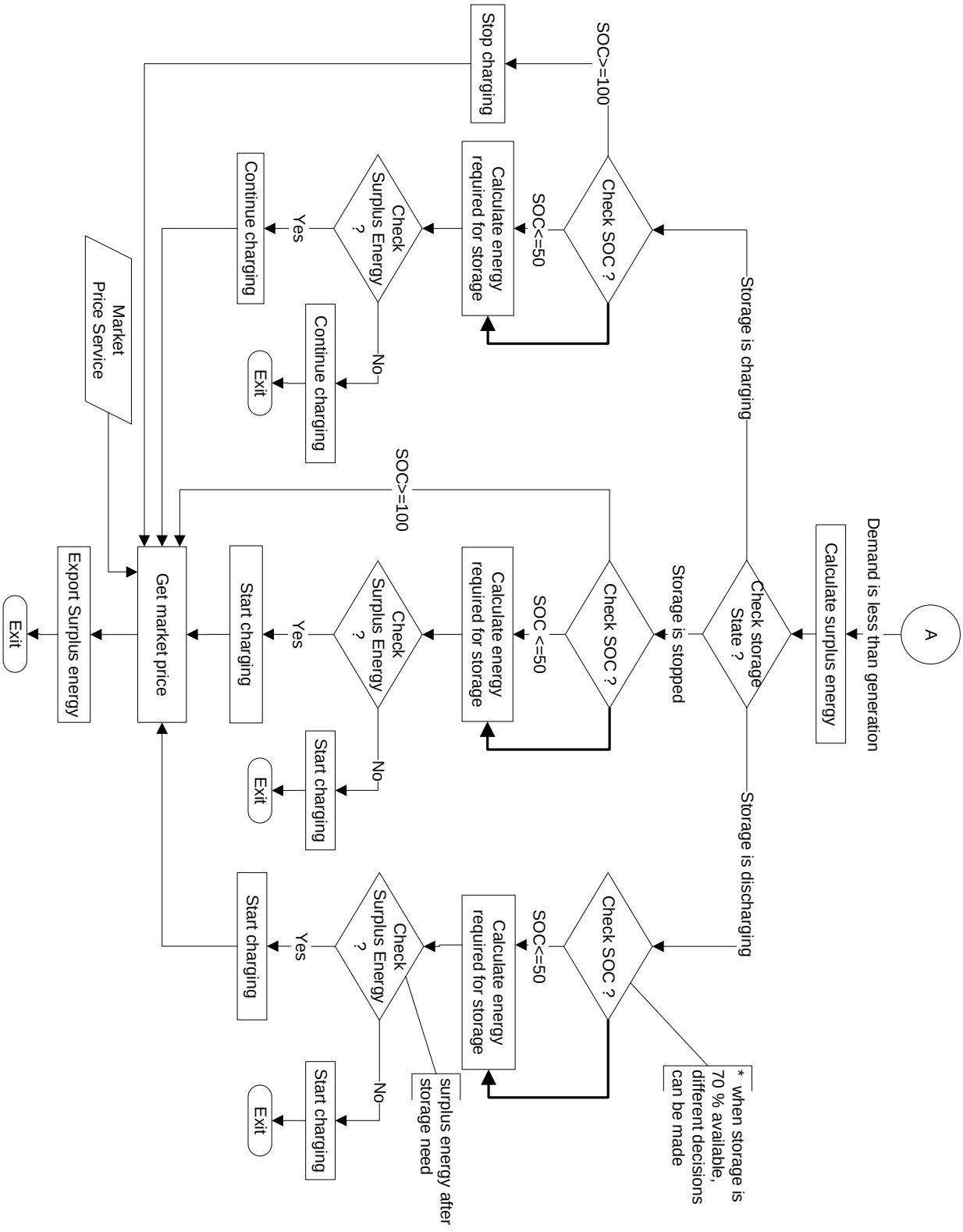


Figure 7.22: Flow Chart representing branch A

(b) **Branch B: Demand is greater than generation (Figure 7.23)**

- Calculate current energy deficit in the zone.
- Check storage state of charge (SOC).
- **If storage is charging**
  - if storage is charging, check upper and lower limits.
  - if storage is approaching upper limit. Stop charging.
  - if storage is fully charged, check if storage is enough to fulfil demand. Also check market price (to compare cost of storage use and buying price from market). If market price is favourable then preference will be to import energy instead of using storage. Otherwise Storage will be used.
  - if storage is not enough to fulfil demand then power will need to be imported from the grid. Depending upon storage condition and market price, storage can be discharged along with energy import from grid. Otherwise storage can be charged if market price is favourable.
- **If storage is stopped or discharging**
  - Check storage upper and lower limits.
  - Check demand can be fulfilled from storage. Check market buy price. Make decision about energy import.

As we mentioned earlier, in this flow we have represented the scenario when storage is available. In the absence of a storage unit, the controller will make decisions about import and export of energy more directly.

The functions that calculate green energy retained in the SSEZ and the cost of buying and selling energy to the grid are not shown here.

In the case of having surplus energy, there is more energy in the SSEZ than required. In that case demand side management (DSM) can be considered in which loads are brought forward. Which means if some equipment is going to use power later in the day can be switched on earlier to use surplus energy, such as home appliances. We have not considered this in much detail here. For this we

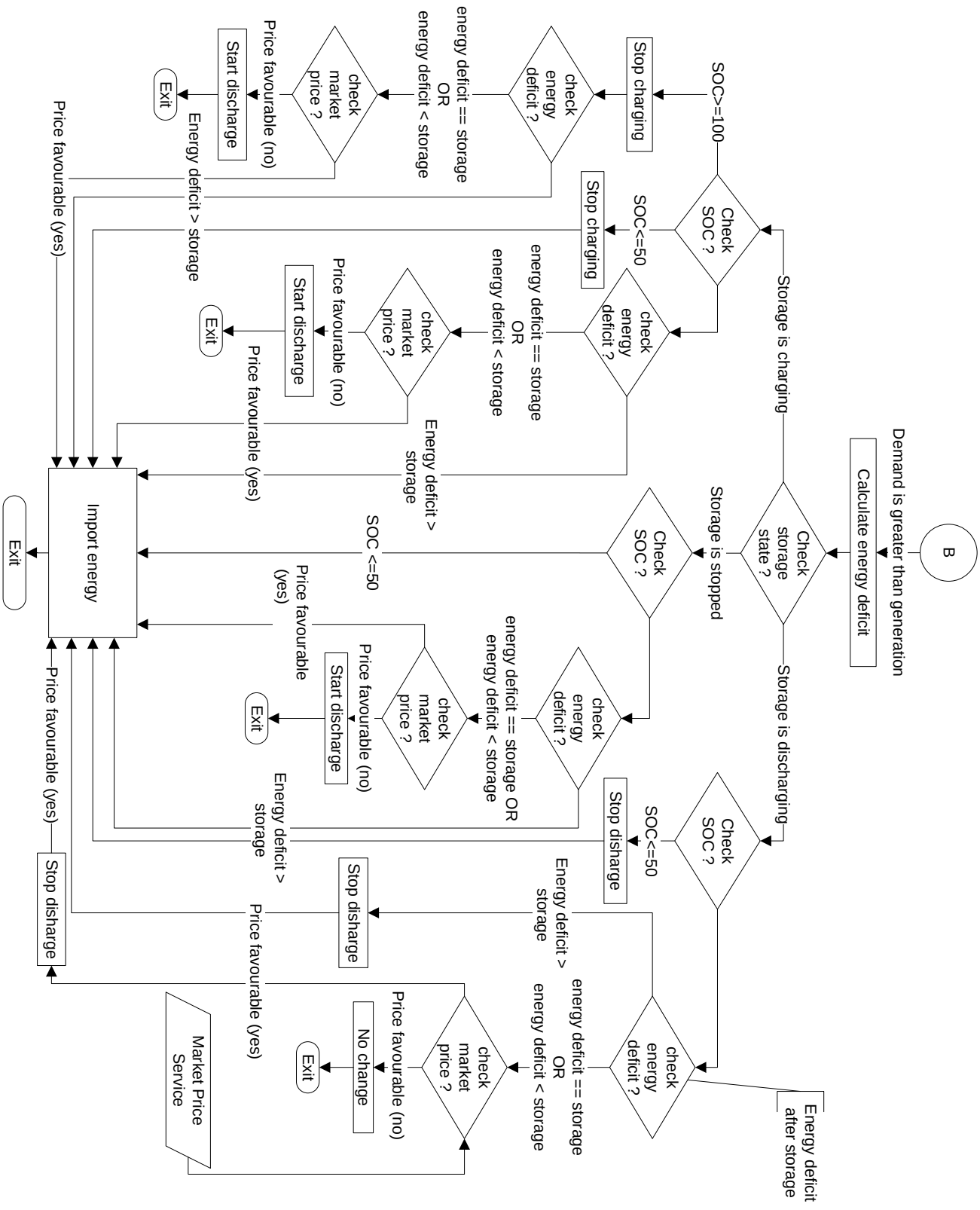


Figure 7.23: Flow Chart representing branch B

need further information about demand side management (DMS) and we need to consider the time of day. Some loads can be brought forward or deferred only at specific times, such as we can defer load of refrigerators as home appliances to night. In the case of a farm we can water the fields at night, so tube wells can be switched off in the day time and used at night.

To keep the scenario simple we have used storage for demand side management.

Another important factor related to the operating policy is to define the favourable price for importing and exporting energy. This will create constraints on the decision also. The decision also depends on the ESCO policy about CO2 emission, and the effect of using brown energy.

7.3.11.2 Scenario for predicted energy deficit in the SSEZ

In this part we discuss the scenario when there is an energy deficit in the SSEZ based on the predicted values of demand and generation. (The scenario for the case when predicted demand will be less than predicted generation is not represented.)

- Get generation and demand prediction values from generation and prediction services.
- Compares these to check the energy balance. If there is an equal balance then no change will be made in current system states as shown in Figure 7.24.
- In case of energy deficit (predicted demand is greater than predicted generation), the flow is shown through branch **A** in Figure 7.25.

7.3.11.3 Scenario for predicted energy condition in the SSEZ (Figure 7.25)

- Calculate energy deficit.
- Get current and forecasted market buy price.

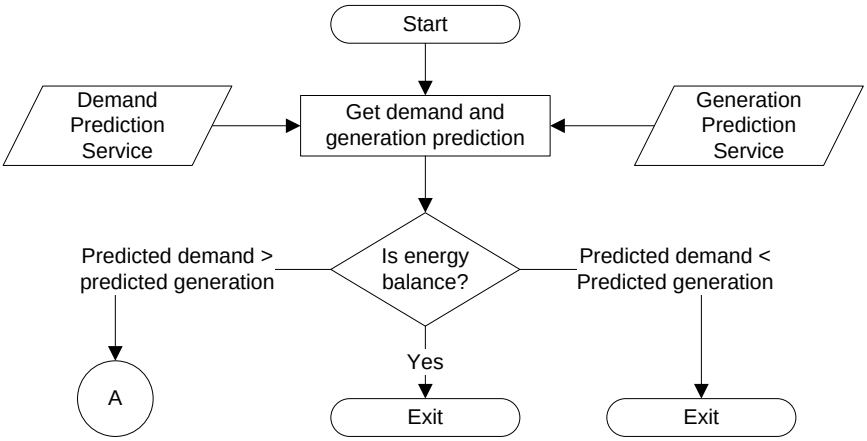


Figure 7.24: Flow Chart representing prediction Flow

- Check if predicted market buy price is favourable. Also check if current market buy price is favourable.
- If current market buy price is favourable then check current demand and generation in the SSEZ.
- If current demand is less or equal to current generation then there is the possibility that storage can be used for predicted demand.
- Compare storage with predicted energy deficit to check if storage can be used to cover the deficit. Also check storage state of charge (SOC) as if the current price is favourable, the storage can be charged to full capacity.
- If storage is fully charged and currently not discharging then no action will be taken. However, if it is discharging then it will be stopped. As the future market price is high, storage will be used for the next half hour.
- Check if there is energy export going on. In that case first priority will be to charge storage and then export energy.

Here different decisions are involved. In case when both buy and sell prices are high the factor about the use of brown energy will need to be included to calculate use of brown energy. On the other hand if the ESCO wants to make

money then it has to evaluate how long it can defer demand and make money in the market when sell price is high and demand is high too.

7.3.12 Design Decisions

The design has been developed to be independent of any implementation technology. The decisions associated with the SBCS are discussed below.

Controller: The Controller is not considered to be a service. This is because it is used locally, and needs SSEZ electrical network specific configurations and constraints. The decision to consider a particular element of functionality to be a service is appropriate when there is more than one consumer of the service. The *Controller Service* is dependent on a large number of context specific information such as electrical network configurations, constraints that include technical constraints and priorities that need information about operating policy. If this service is provided to another ESCO then all this information will be required to configure the *Controller* functionality.

Also we need to consider the ESCO long term policy about services. By this we mean that currently the ESCO is using third party services, however, at later stage if it decides to provides its functionality as service to other ESCOs then how will this be done? This is an important decision that needs to be made at the early stage of service based application development.

Registry: The registry will be used at the consumer site. This decision is made because of the domain of application. In other words we identify it as a domain specific decision. Time is an important factor in a control system and using a third party registry can be a possible constraint.

Market Service: The Market service is represented as a single service provider. Although it can be assumed that this will be offered by different service providers in case when neighbouring ESCO offer this service. This assumption add to important points in the design such as use of registry.

First, it adds further choice for *Controller*: whether buy green energy from other ESCO or use brown energy from the national grid. Second, in case of

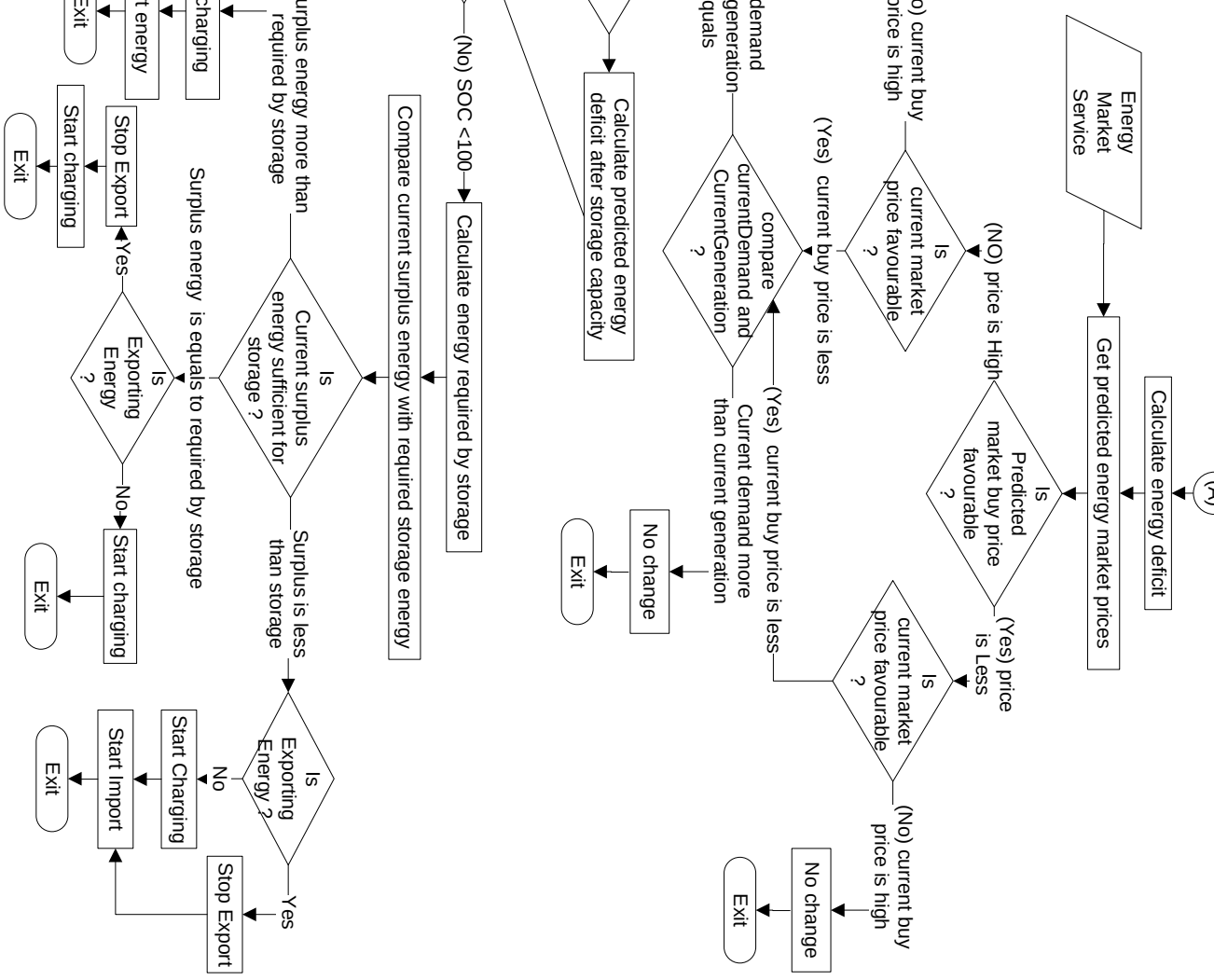


Figure 7.25: Flow Chart representing branch A for prediction Flow

these being more than one service provider for generation we need to add their information and the registry, provides a mechanism to do this.

In this design we have intentionally considered one service provider to represent the scenario where we have a mix of one and more service providers for different services. Also, we may have different contract with these providers that can be long as 6 months to one year. This feature is domain specific and depends on the need of the ESCO.

System States Service: This service is logical and internal, responsible for collecting and providing network status to the controller.

Weather Service: Weather data includes temperature, wind speed, wind direction, solar irradiance, cloud conditions, along with information about the area. The level of detail offered by each weather service can be different.

We have made the decision to include the weather service from the beginning because SBCS deals with two situations: present and future. The current weather information is treated as being part of current system state and to maintain history about the condition of the zone. The weather forecast is required to evaluate the situation in the zone and to assess the effects of upcoming events such as sports and Christmas.

Further, the ESCO is currently taking generation prediction and demand prediction services from a third party. However, at a later stage it might consider providing these services as part of its business.

Different level of details are provided by third party services. In the case of the demand and generation prediction services these may be very simple services that take weather information, demand and generation values and provide the prediction. There can also be more sophisticated services that require network information, and location information along with current demand and generation data. They can also use their own weather service to calculate demand and generation prediction.

Network Configurations: These are technical constraints that include electrical network low level details (such as assets information) that are not included in the design. This role is associated with the *Controller*.

Operating policy: This is required to determine new decision making will be performed at different levels. The policy set the priority for the *Controller* to use for making different decisions. We have discussed some examples in Flow Chart section. The decision that the SBCS will try to remain self sufficient means that if there will be more demand in the zone, the first priority for the SBCS will be to defer this. However, if the highest priority is to meet customer demand then the SBCS have to import power from grid. In that case it has to evaluate the use of green and brown energy. Also if its goal was to minimise brown energy that will also be effected by this. From *Controller* point of view operating policies are important factor in SBCS.

Non-functional Features Some design time non-functional attributes are considered, such as time and cost. These provide the main selection criteria for the services. However, when it comes to availability of the service at run time the *Controller* might need to compromise on the level of detail.

7.4 Discussion

The design activity carried out for the SBCS is different from existing studies. These studies can be broadly classified as:

- the ones that discuss and propose service oriented life cycle models such as (Offermann and Bub, 2009; Gu and Lago, 2007; Papazoglou and Heuvel, 2006; Erradi and Sriram Anand, 2006; Karhunen et al., 2005).

- There are studies that make use of UML profiles. Some of them have used UML profile with modelling techniques such as model driven architecture (MDA). The studies (Ali et al., 2010; López-Sanz et al., 2008; Zhang et al., 2006*b*; Wada et al., 2006; Amir and Zeid, 2004; Stojanovic et al., 2004) come under this category. The diagrammatical forms used are limited to class and component diagrams. There is extensive use of stereotypes to explain SOA features.

We have addressed the problem of service based application (SBA) design by producing a high level design through the use of existing diagrammatical representations. We have captured the problem from the very beginning; from requirements. We have discussed design issue from a software designer's perspective. The design decisions made at different levels are also explained.

In this section we discuss the problems of service based application design in terms of our experiences related to:

- the novelty and immaturity of the SOA paradigm, and
- the choice of design and notations used in SBCS.

7.4.1 Evolution of existing Paradigms and SOA

The structured and object oriented (OO) paradigms evolved over time and now have mature software development life cycles. Application developments based on these paradigms largely differ from each other because of their underlying assumptions, the major functional elements and focus of analysis.

Structured development techniques analyse the system from a functional view point. The main functions are identified that software need to perform. These functions are further divided into sub functions and sub tasks to provide required functionality. The process is constructed to visualise the functions working together. This is called a top-down approach. Loy (1990) has mentioned this as being a functional paradigm.

In case of object oriented development, the analysis involves thinking about objects, their attributes, and relating them to the operations defined on them Loy (1990). This is termed a bottom-up approach.

However, each new paradigm borrows techniques from previous one and adds its features into this. According to Wieringa (1998) object-oriented methods adopted structured techniques in a 'new guise' and we can benefit from this by making 'technology transfer' explicit.

The object oriented design methodologies developed over time. Figure 7.26 provides an abstract view of its evolution. The OO design was initially based on structured techniques. As the concepts matured and a shared vocabulary

emerged, the design techniques for object oriented applications also evolved. Its realisation also provided input to develop these techniques. The diagrammatical forms used to decompose the problem and to represent object behaviour and communication were developed, and in turn contributed to ideas such as design patterns, UML , RUP.

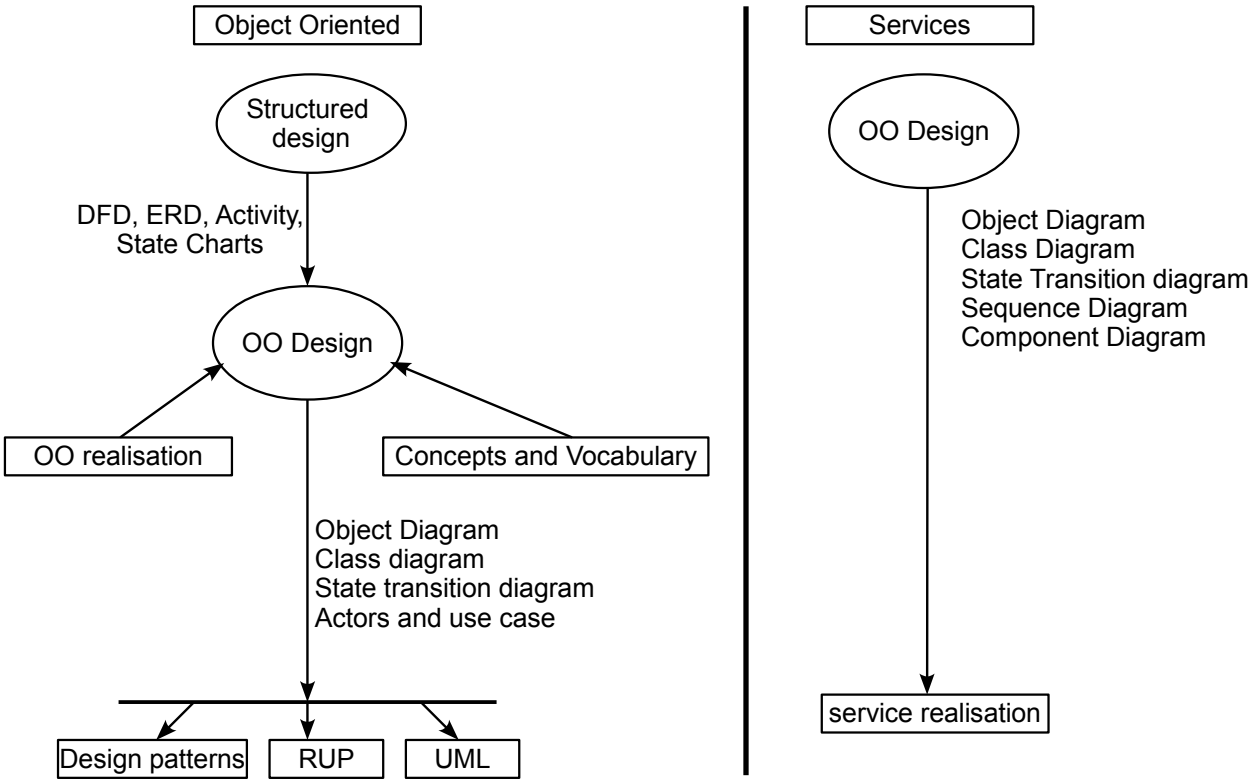


Figure 7.26: OO Design and Services

In the case of service based application development, a gap exists in design techniques (Stojanovic et al., 2004). Currently, services community is adopting OO techniques for design and its focus is more on SOA realisation through web services. There is a lack of agreement on a shared vocabulary and related concepts. Further to this the need for design techniques is not fully appreciated.

The procedures explained in literature about service based application design are usually based on technological solutions. Mostly application development is discussed in terms of web services such as the study by Papazoglou and Heuvel (2007). In other studies, service compositional aspects are discussed in terms

of service identification, publication, service selection and its composition and execution. The notion of the designing service based applications by representing its features through SOA notations is lacking. Erl (2009) has proposed a set of SOA design patterns, however, their practical realisation is missing. The life cycle for SOA, as suggested in the form of service oriented software engineering (SOSE) as it appears in literature was discussed in Chapter 3. However, this is not mature. In the case of service based application design, tools are not available, and although work flow based languages have been suggested for this, these are very specific in their scope.

7.4.2 Design and Notations

The SBCS design process has been constructed by identifying the main functions that when combined, should provide overall system functionality. We suggest that this approach is closer to that of structured design. We have identified that every functionality in the system does not need to be presented as a service. Further, we need to identify and categorise services. The ones which will be provided by service providers and the one which will be logically exist but have no service provider. The approach we have used for services is closer to OO in terms of identifying its attributes, roles and related operations. Indeed, we have used an opportunistic approach to design the SBCS SOA model.

In terms of notations, we have made use of existing diagrams. In diagrammatical representations, syntax, semantics, the annotations used to interpret a diagram, and the domain information in which the diagram is to be represented are important. The purpose of a diagram is to be used a means of communication and the lack of semantics invites multiple interpretations of the same diagram (Wieringa, 1998). Therefore they need to be used correctly to ensure that they serve the intended purpose. The notations associated with particular paradigms (in case of OO the UML) have their implicit meanings. The object diagram embodies the philosophy of 'object'.

For SBCS design, we have used different representational forms. We have also used tabular forms to map services. The notations we have used are a mix of structured and object oriented design.

- The class diagram was used for service interfaces and dependency.
- The component diagram was used to represent the services that have service providers and to represent where choices about service selection exist.
- To represent the overall flow of how services and other functions will work together, we have used the activity diagram.
- Sequence diagrams have been used to represent messages among services and other functions.
- Flow chart was used to show overall system behaviour, It represents the process where services and system internal working is combined.

In our design we have added SOA vocabulary in existing forms along with description to provide the context. In the literature, the activity and class diagrams are predominantly used for service composition and service interfaces respectively (Skogan et al., 2004). Through stereotyping meanings are given to the diagrams. To construct processes, workflow languages (BPEL, BPEL4WS etc.) are available, however, they are not that mature or semantically strong enough to express the concepts (Aalst, 2003). We have focused on the high level design and have not considered work flow languages.

7.5 Summary

In this chapter, SOA design model constructed for SSEZ control system is discussed in detail. The design process that includes different activities is described. The design is explained through existing diagrammatical forms. Different scenarios are constructed to represent SBCS behaviour. The discussion section provides a brief overview of the issue of design for service based applications.

Chapter 8

Evaluation

8.1 Introduction

This chapter discusses the evaluation which has been carried out as part of the case study. The grey area in Figure 8.1 shows where this fits in the case study process.

The chapter describes the evaluation process, the techniques involved, and the outcomes. The techniques employed in the evaluation process are discussed in terms of how they have been used; what their results are; and the lessons learned from their use. The outcomes of the evaluation are discussed in section 8.3. The review experts for the walkthrough were selected from two different domains, therefore throughout in the chapter, the expert having background in computer science is referred to as the *'software expert'* and the expert from energy engineering is referred to as the *'application domain expert'*.

8.2 The Evaluation Process

The evaluation process has been performed by combining ideas of a structured walkthrough with elements of action research. The purpose of using a walkthrough was to evaluate the use case and the resulting SOA design model through expert reviews. We considered the walkthrough technique appropriate for evaluation since we could not find an application that could be used as a baseline

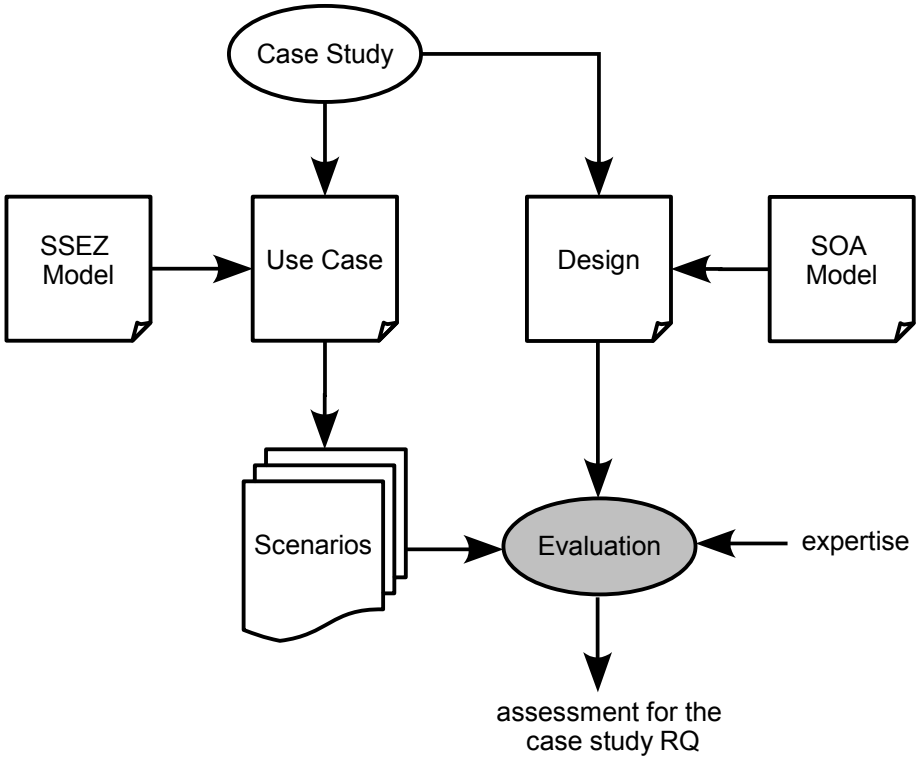


Figure 8.1: Case Study Design

for comparison. Further, the study involves knowledge of two domains that need to be validated, therefore, expert reviews are needed. By doing this we have introduced a qualitative approach into our study as also discussed by (Seaman, 1999). To perform a walkthrough, a protocol was established that contained the information about the review structure (Appendix E).

Since the walkthrough has been used for an academic purpose, which is not the usual practice, we considered it appropriate to combine this with an action research approach. Action research is a *'cyclic'* approach and is based on the process of *'plan-act-reflect'* (Oates, 2005). Hence, after each walkthrough session, interviews were conducted with the participants in order to collect feedback about the walkthrough process and the design presentation. These were semi-structured interviews and a questionnaire was prepared for this purpose (Appendix F).

The evaluation process consisted of two walkthrough sessions, each followed by an interview with the participants, as shown in Figure 8.2. The data collected

from the first interview was used to improve both the walkthrough and the design presentation where possible. The purpose of the second interview was to evaluate the effectiveness of this approach by comparing both walkthrough sessions and to identify where further improvement is required.

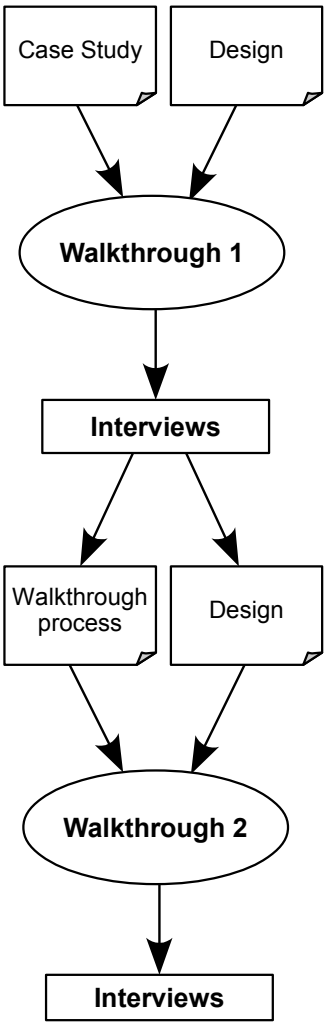


Figure 8.2: Evaluation process

The research question that we addressed through the evaluation was *"Are the design and notations used in the case study appropriate for constructing an SOA model for the specified SSEZ control system?"* and hence, *"Can a service-oriented architecture handle the problems of an SSEZ control system?"*

In next sections we describe the conduct of the walkthroughs and discuss the

data we have collected through the interviews.

8.2.1 Walkthrough Sessions

In this section we provide details about how the walkthrough sessions were conducted.

Form of review: A walkthrough is a form of review that is slightly different from other review forms such as inspection. Inspection is considered to be a more formal technique used for defect identification. The process involves more steps than a walkthrough and there is extensive use of checklists. This provides more of a mix of quantitative and qualitative data whereas walkthrough outputs are largely qualitative. We have used a walkthrough to evaluate our case study elements, including the use case and the SOA design model. In a walkthrough, the documents are made available to the review committee before the evaluation. The reviewers then review the document critically before the session takes place. During the review session, checklists are used for evaluation. Instead of preparing checklists as carried out in inspections, that seek to identify defects in the code, we have used a questionnaire for our walkthrough. A set of documents that included the walkthrough protocol, the use case chapter from this thesis and the design chapter were made available to the review team.

Further to this, we used part of the walkthrough session to present the design. This approach again makes it different from a conventional walkthrough where the reviewers are involved in document content analysis. Our approach is more exploratory, discussion oriented and informal. This fits well with the purpose of a walkthrough as described by Weinberg and Freedman (1984) who considered a walkthrough to be informal and hence appropriate for use for educational purposes.

Review Team and Roles: As identified in the protocol, the roles for the walkthrough include the moderator, reviewers and the author. The session was chaired by the moderator. A moderator with a computer science background was selected so that he would be able to track the discussion and

keep the session on schedule. The walkthrough protocol was sent to him prior to the session. The review schedule and the questionnaire were also made available to him. However, the use case and design documents were only sent to the reviewers.

The expert team included two members; one from the energy engineering research group and other from computer science. Neither expert was involved at any stage of the use case and design preparation. This was an important element to avoid bias in the review session. The fourth participant in the review was the author, who was responsible for presenting the design, along with the information about the use case.

Review Schedule: The duration for the review was two hours. This time period is considered effective for a review session (IEEESTD, 2008) and so was the time period defined in the review protocol.

For the second review, the duration for the review was again two hours. In the absence of a moderator the review questionnaire was used to keep the review focused and to make sure that all of the questions were covered in the session.

Review Procedure (First Review): The review session was chaired by the moderator who introduced the team members and provided the detail about the session. The author provided an overview of the use case and the requirements. The main elements of the system and how they link with each other were explained. This was carried out using a whiteboard. After that, discussion with reviewers was carried out, based on the questions they raised from the documents provided.

Before design presentation, the author was asked to provide a brief overview of the service oriented architecture (SOA). The design is based on SOA concepts and it was considered important to outline the concepts to the review team, as they provide the context for the design and design decisions. As we discussed in the design chapter, each development technique has its own requirements and it is important to make explicit the choice of design techniques employed. The other purpose was to provide an SOA overview

to the application domain expert.

The moderator used the review schedule as defined in the review protocol, dividing the session into three categories; requirements, assumptions and design. At the end of each category, the questionnaire was used to ensure that all the questions listed in each category are covered. The questionnaire had the same categories as were used for structuring the walkthrough session.

Data and Record Keeping: The review was conducted in an environment where audio and video facilities were available. A microphone was attached to each member of the team. There was also a room audio system that recorded the presenter's voice. Video cameras were allocated to each team member and also to capture the whiteboard activity. The complete session was recorded, which eliminated the need for taking notes. This was also useful to avoid inconsistency in the collected data and to remove any possible bias.

The second walkthrough session was largely carried out in the same manner. However, there were some modifications including the ones that were identified through the interviews with the participants, and some that were due to unforeseen reasons.

Review Documents: The first walkthrough identified that there was a need for additional documents that could help in understanding both domains. For this reason, further supporting documents were prepared. The documents were sent to reviewers and also to the moderator. The new documents included:

- Diagrams representing a system level view.
- A component diagram
- A flow chart to represent current system flow
- A flow chart to represent part of prediction flow
- A list of acronyms - a glossary was suggested but the terms were already explained in relevant chapters of the thesis.

- A document that provides a key for each notation used in the diagrams
- An SSEZ network diagram design to assist the understanding of an audience coming from outside of energy engineering.

Review Team and Roles: The team members remained the same and no changes were made in the roles. The moderator was unavailable due to an incident on the day of the second review, and no arrangements were made for another one.

Review Procedure (Second Review): The session was carried out using the pattern employed in the previous walkthrough session, but with some modifications. The session started with a formal presentation representing the problem and design. The author presented the design and explained the reason for using each representation. After that, a discussion was carried out with the reviewers. The reviewers asked questions from the documents provided and from the presentation. The review was again organised around the three main categories defined in the questionnaire, although the questionnaire was used to check that all of the questions are answered.

The interview sessions conducted after each walkthrough session are described in the next section.

8.2.2 Interview Sessions

The interviews were conducted following each walkthrough session. The purpose of this activity was to collect feedback from the participants about the walkthrough process and about the presentation of the design. We considered it necessary to hold debriefing interviews because:

- The walkthrough was conducted with a team that had little prior experience. Therefore it was considered important to collect participants' feedback.
- The walkthrough was used as part of an academic exercise, which is unusual. Therefore we considered it necessary to collect participants' views about

its effectiveness. This provides an action research element to the review process.

- The walkthrough involved knowledge about two different domains. It was important that the reviewers and author had the same understanding of the problem regardless of discipline. For this, views about representation were considered to be particularly important.

A semi-structured questionnaire was prepared to support the interview process (see Appendix I). The interview consisted of two parts. The first part was about the walkthrough process, and included the effectiveness, organisation and any improvement of the process if required. The second part was about the design presentation itself, and included the understanding of the design, its presentation and any improvements required.

The interviews were conducted separately with each participant. During the interview, the questionnaire was provided to the participants, and they were asked to discuss any other issue that they considered as being important for the walkthrough. The interviews were also recorded.

8.2.3 Data coding and Analysis

The walkthrough data was available in the form of audio and video recording. As we mentioned earlier, the walkthrough session was divided into three categories: requirements, assumptions and design. In each category, a set of questions were listed to aid the walkthrough. This categorisation made the coding a bit easier in terms of classifying the discussion and the questions. The classification was used by the reviewers whenever they have to ask any question related to previously discussed category. This categorisation helped us for analysing the walkthrough session which was done by consulting recordings.

The interviews were semi structured and participants were given the freedom to discuss any other issue they find important other than the questions listed in the interview questionnaire. The interviews were analysed by listening to audio files and with the help of questionnaire used for interviews. The summary of responses is attached in Appendix J where we have mentioned the main issues raised from the interviews.

The issues that we raised from the walkthrough and interview sessions are listed in Table 8.1. We have associated these issues with four categories: Interdisciplinary, Walkthrough structure, Experience, and Design presentation. We have not mentioned the second interview as no issues were raised by the experts about the walkthrough process or design presentation.

Table 8.1: Issues Identified from Walkthrough and Interviews

Issues (or changes)	Issue Type	W1	I1	W2
Information about design notations	Interdisciplinary	X		
Simplified Diagrams for computer scientists	Interdisciplinary	X		
Outline of SOA	Interdisciplinary	X		
List of Acronyms	Interdisciplinary	X		
Use of Powerpoint Presentation	Design presentation		X	
Need for preparatory session	Walkthrough process / Interdisciplinary / Experience		X	
Availability of documents to moderator	Walkthrough process / Experience		X	
Use of questionnaire	Walkthrough process (control)			X
Walkthrough purpose and context	Walkthrough process (protocol)		X	
Inclusion of design decisions	Design presentation			X

Views about the effectiveness of the walkthrough as assessed in the second interview, are summarised in Table 8.2.

8.2.4 Outcomes of the Interviews

The responses we got from both interview sessions are summarised below.

Walkthrough Effectiveness: From the responses collected from participants, it was clear that they found the walkthrough process to be effective in terms of:

Table 8.2: Summary of responses from second interview session

Questions	Responses
Do you think that the walkthrough was effective in terms of meeting its aims?	*"The initial presentation (first walkthrough) did not give us additional information what we have read in the document but talking about the way you came up with the solution was useful." "It was a multi-angle approach to solve the problem. How you thought about the solution was useful. Knowing the process was useful."*
What elements do you think were lacking in the organization of the walkthrough, both in terms of the process and of the material provided?	*"The material is quite comprehensive." "Should not let reviewers to set their agenda. I was interested in process and you want to concentrate on design."*
What things could be done to improve the walkthough process?	*"Process was fine. You did tell us what you want to do."*
How well were you able to understand the design of the software system?	*"The design was understandable and presentation made it a lot clearer."*
What could be done to improve the design presentation, both in terms of how it was organized and the forms used?	*"Presentation was better. The slides kept you focused and opening was a lot better." "Addition of new diagrams especially electrical network side by side a view for non-engineers was making things understandable."*
Are there better ways or forms that we could use to describe and present the design?	*"What you presented was logical and with enough detail."*

- Getting feedback on the work produced.
- Identifying the issues that had been ignored by the author, and filling any gaps in their understanding of the domain.
- Learning from these sessions, they noted that they found it interesting to view the problem from both angles; computer science and engineering.
- Information about new development approach such as SOA.

Walkthrough Organisation: The participants found the organisation of the walkthrough appropriate in terms of its structure and the schedule. The first interviews recommended keeping the same team for the next session.

In terms of walkthrough material, it was identified that there is need to provide further detail about the goals of the review. The participants initially considered it to be a thesis review, and therefore in the walkthrough the focus was more on documents than on the design.

Design Presentation: The responses we collected indicated that:

- Participants found the design understandable and comprehensive.
- The responses we got about the actual design presentation were mixed. The application domain expert considered it important to get more information about the diagram notations, which were considered implicit knowledge for the computer science participants.
- The other point mentioned by participants was the use of PowerPoint presentation.

The responses we collected through the interviews and the lessons learned from the experiences of using walkthroughs are discussed in next section.

8.2.5 Discussion on the use of Walkthroughs

The walkthrough was conducted as part of the evaluation process. The use of a walkthrough as an evaluation technique is certainly not new (Seaman, 1999;

Card et al., 1987; Weinberg and Freedman, 1984), although this is not a form that is normally used in academic research. Therefore, in this section we discuss our experience of conducting walkthroughs and the lessons we have learned from this exercise.

- From the responses from the participants in the walkthroughs, we can identify that it is necessary to explain the purpose and context of the walkthrough more clearly, especially when:

  (a) the audience is from a mix of different domains,

  (b) the participants have little or no experience of conducting walkthroughs, and

  (c) when the technique is used in a new way, such as our use of walkthroughs for academic purposes.

  We developed a review protocol and made this available to each participant. However, during the interview sessions we identified that the context was still not very clear to them until the actual walkthrough session.

- The purpose of using a walkthrough was to present the design and explore its different features through a board exercise. However, in the first session of walkthrough this turned into a formal evaluation of the use case and design documents. The reviewers focused on the content of the documents. The author explained the points which were not clear to the reviewers and where required, provided information about the terminologies and the concepts.

  This helped to identify the challenge of presenting interdisciplinary research. The solution was identified as being to produce further supporting documents in the form of glossary, list of acronyms, presentation forms for diagrams, so that people from both domains could share an understanding of the notations used for each diagram.

  For example, the network diagram was understandable by the application domain expert but it was difficult to understand for the reviewer whose

background was in computer science. However, while the notations used in the design diagrams were not an issue for the software reviewer, the application domain expert required more information to understand them.

The first walkthrough therefore helped to bring the participants from both domains to the same level of understanding. Also, the major part of the domain was covered in the first walkthrough, along with the background information about SOA for the application domain expert. This cleared the picture and in the second walkthrough the reviewers and author were able to focus on the design. As mentioned in the interviews, the first walkthrough helped with providing an understanding of the context for both domains.

- It proved challenging for the author to:

  (a) present work to an audience having different backgrounds,

  (b) simplify the problem to explain the design clearly, and

  (c) categorise the problems when they are interlinked.

- The presentation of the design was more effective when PowerPoint was used. In the first walkthrough, the author presented work by using a whiteboard. The subsequent interview with the participants identified that a better approach would be for a presentation to be used side by side with whiteboard activity, because the board discussion contains acronyms and the reviewers need to consult the documents. Also a diagram and page number can be referred to on the slide and the explanation can be carried out with the support of whiteboard.

  One of the participants found that the whiteboard presentation served the purpose and so felt that a PowerPoint presentation would not add much. However, in the second round of interviews we identified that all the participants found it a better approach to use a formal presentation. This made the walkthrough more focused and easy to analyse, and from a presenter point of view, it was easier to explain the context of each diagram in a systematic way. This indeed helped the reviewers to understand the context and purpose of each diagram used.

The reason why one of the participants was originally not in favour of the PowerPoint presentation was the use of content on the slides. It is in normal practice that people put details on slides that make it a replica of what is written in the document. Also, it puts more cognitive load on the audience. However, the presentation we used, contained only the design, and the context and purpose were explained verbally.

- From a learning perspective, the responses we collected from the participants were positive. For the application domain expert it was interesting to see an engineering problem discussed in the computer science domain. He found it interesting to see a 'multi-angle' approach towards the problem. Further, instead of being based upon an implementation point of view, the problem was discussed starting from the design, which was different. The participants and the author gained experience about dealing with the situation when two views have to be discussed in a simplified way. However, presenting both domains side by side was challenging.

- The first walkthrough session focused more upon providing the background information about energy domain and SOA to participants. The reviewers from both domains needed this to clarify the acronyms used in the document and also to get clarity about the terms used. It was important for the reviewer from engineering to mention how we have understood the domain and where we are focusing. For the software expert it was important to get domain knowledge to evaluate design. So, in this case, the first session could be viewed as a preparatory session.

- We identified that the role of the moderator becomes important when reviewers are from different domains, and when the author has the role of presenter. In such a case the moderator helps in eliminating possible bias from the reviewers and the author. In the first walkthrough, the discussion on 'weather forecast' which took a bit longer was eventually interrupted by the moderator.

  From the combined walkthrough sessions, we note that the element of control is very important, whether applied through the role of the moderator

or by imposing a strict structure. We used the latter in the second walkthrough session, when the moderator was not available. The questionnaire used in the first walkthrough to ensure that all categories and features were covered was used in the second walkthrough for the same purpose and also to keep the walkthrough focused. As the walkthrough technique is informal and discussion-oriented, there is a possibility that discussion can exceed the specified time and can involve the elements that are not the focus of walkthrough. In our case, in the absence of a moderator, the questionnaire helped to control the discussion and to keep the walkthrough to its schedule.

We would observe that the moderator role involves more than tracking time. From the interviews and walkthroughs we can identify that it is important for the moderator to have some level of knowledge about the problem discussed. Otherwise it becomes difficult to track the discussion. Therefore, a domain background alone is not enough. Knowledge about the problem is also important.

- From these experiences we can identify the following points as being important for conducting a walkthrough of this form.
  - A preparatory session may well be necessary when participants have different backgrounds.
  - Documents should be available to all participants including the moderator.
  - The presenter needs to put effort into presenting both sides of the application domain.
  - Control through a moderator or by a fixed walkthrough structure is needed.
  - Documents need representations that can be understood by both domains.
- The benefit of such walkthrough exercise as an academic discipline is that we identified through our interview with participants and from our own experience is that it provides:

- a better understanding of both disciplines and hence helps with identification of inconsistencies.
- provides a different way of analysing the problem.
- provides knowledge sharing for both disciplines.

8.3 Discussion on the outcomes of the Review

For the review, we defined three categories for discussion that included requirements, assumptions and design. For each category we allocated a time slot. The purpose was to evaluate the completeness and correctness of the use case and the design in the most effective way. For this reason the questionnaire was made available to the reviewers. This was used at the end of each category to make it sure that all questions are answered. In the rest of this section we review some key issues about the design that were raised in the review.

8.3.1 Requirements

Why are future extensions mentioned in the use case document. For example landfill gas generator is discussed but not used in the design. Why not remove this information from the use case?

We can answer this question in two ways. For designing an application, a designer needs a reasonable knowledge about the domain and the requirements. This is useful for developing the system by keeping a broad picture in the mind and able to incorporate any further extensions that are required later in the system. Service oriented architecture provides such scalability implicitly. However, from our experience we have identified that not all functionality needs to be presented as service (if we want to use the term 'service' then we have to create some categories of services and make their meaning clear. Such as we can categorise service provided by a third party (having ownership), and also those that are logical and local to application).

If we know from the beginning that we may need to expand our system in certain directions, its better to include that in the requirements under possible future extension.

We can start from a very narrow scope and later identify the things that do not fit in our model. For the design a broad picture of the system is required.

The other reason for adding this section in the use case was unfamiliarity with the domain and availability of the resources. We have collected information that can be added further in the system and can also help to understand the system. While constructing requirements we identified that energy engineers are trying to make a mix of green and brown energy generators. Because renewable resources are not fully implemented, other sources of power generation are also in use. Therefore, it is important to evaluate the CO2 emission and the cost involved in using these generation sources.

Later in the review, when the generation service was discussed, the reviewers identified that if another generator like landfill gas was added, that this would be able to fit under this service, and so in this way, SOA can provide a flexible solution for use in the energy domain.

Figure 6.2 was difficult to understand for the software expert. The figure is drawn from an engineering point of view. It represents the SSEZ electrical network and the important entities within that.

This figure was drawn to use a network model as reference point. While extracting the requirements, we were asked about the network configurations, as each network has its own properties and engineers like to know about that before discussing it. This helps them to set the context.

During the review, we determined that the figure was understandable for the application domain expert. However, for a computer scientist a more abstract figure is really required.

This problem can be identified as one arising from the interdisciplinary nature of the study. A side by side picture for both domains is required. For this reason another figure was added to the use case document.

The use case does not contain information about the assets such as transformers. This information is required if technical constraints are to be considered in the design.

The technical constraints involve details about network assets such as transformers, cables etc. This involves using further information about the thermal limits, voltages (regulation, unbalance, rise/drop), network losses, phase angles,

location of these entities (phases or network) and configuration details. We have not discussed these in details for two reasons. Firstly it involves further information about the energy domain and requires the designer to focus on the lower level details of the network.

Secondly, this information is categorised as network constraints and used by the controller to check that network measurements are not violated. This further involves information about network statutory regulations. The technical constraint depends on the configurations of individual networks, therefore, we have briefly mentioned this in the use case.

The value of the Capacity factor (CF) is used in the Table 6.4. Why is this information included at this point in the study?

While working on the requirements, we were told that the outputs of the wind turbine and PVs do not provide their full capacity. So their actual outcome is less than what is mentioned in theory. Therefore, while estimating the required number of wind turbines and PVs for our electric network we considered the capacity factor. Further, as we are not considering any other source of energy generation other than renewable sources, therefore we considered it important to mention CF.

However, from the review we identified that it is not necessary to mention CF at this point. This information is used for economic purposes and does not need to be considered. Here we found two different opinions from application domain experts, however this is not critical from the design point of view. From the domain perspective it is extra information and can help the ESCO later when calculating its yearly estimations.

The information about storage technology is important. In the use case there was inconsistency about storage capacity. Two different figures were mentioned 1MH in description of electrical network and in the table it was 2MH. Both values give different meanings.

For the application domain expert it was important to consider the type of technology used for storage and to mention this explicitly. There are different types of storage and these are used in different ways. Further, the important point is to know the difference about how much energy can be stored and how fast it can be accessed from storage. These are two different things and depends

on the type of the storage.

The use case does not provide details about the storage technology. From the requirements we identify that the storage can be assumed as a black box (a buffer). This is what we have considered in the use case. However, the use case contains the basic information about the storage unit. Here, the reviewers considered SOA a good option to be used for an energy system. At the abstract level we can provide outline information about storage, and later the details about the storage can be expanded.

The problem was raised due the difference of the storage capacity mentioned in the use case document. We are considering a 5MW network, in which a one MW load is from the storage unit. Now if we add time with this then it changes the meanings, for example, 1MH energy means 1MW power is used over an hour. However, 2MH means 1MW power in 30 minutes. As we are working on a 30 minutes time frame we have to consider that what capacity is available in 30 minutes. This means that if we want 1 MW power in 30 minutes from the storage then we need a 2MWh storage unit. The capacity mentioned in the network description is therefore consistent.

The information about the state of charge (SOC) is inconsistent.

We have mentioned different options that can be considered about the SOC in the use case. These options can be considered as operating policy of the ESCO. Both reviewers understood it differently. Both were correct. The energy engineering point of view was about storage technology and the software expert analysed this from a software architecture point of view.

The inconsistency was found to arise from the use case document and the simplified version of use case provided for review. We assumed storage SOC 50% which was mentioned in the later document, but this caveat was not made in the first one.

Was a full fault analysis carried out as part of the requirements?

The initial answer was no. The term 'faults' was not clear. Does this mean problems in the network assets or does it mean the exceptional conditions from software perspective.

At this point, both experts exchanged their views about what they mean by faults. For application domain expert, faults means the short-circuits in the

network such as when generators are broken down. The situations when circuit breakers trip and customers no longer get the power. This situation is considered as a fault case.

However, the term *'technical constraints'* is used that is different from 'faults' to explain the issues related to the control system. They are associated with thermal limits and network configurations. Therefore, according to the application domain expert, the thermal limits mentioned in the use case document are associated with control and cannot be considered as fault case.

This issue was raised as the term 'fault' used by the software expert was having different meaning to the one used in engineering. By faults he meant the exceptional conditions when controller would have to take extreme measures such as to shut down the wind farm in case when wind speed is too high. Or the situation when the SSEZ is isolated from the grid. This situation is called 'islanding mode' in energy engineering. In this case, the SSEZ would have to reduce its generation to avoid extra power in the zone that can damage the appliances.

If we consider faults from an engineering perspective then they are not included in the use case. We have mentioned in the assumptions that the network is operating in normal conditions which means no faults are present.

In the case of technical constraints, particularly network thermal limits, the basic information is available in the use case. We have not discussed this in detail for two reasons. First it involves detailed information about voltage rise and fall, and the associated frequencies. Also this information is more about network asset control. Secondly, this information is local to the application and can be added when realising the working of the controller. For this reason it was added as part of the technical constraints in the use case but not discussed in detail.

In the case of exceptions we listed scenarios (the extreme cases such as islanding) where controller decisions were required. Therefore, the reviewers were brought back to the documents where we have mentioned network assumptions and scenarios.

We categorised this discussion as a problem associated with interdisciplinary research.

How is demand side management (DSM) handled?

From the requirements we can identify that storage can be considered as load and so comes under demand side management. However, demand side management that deals with customers is not discussed. This involves information coming from each category of load and then defining the demand categories such as which demand critical at what time of the day. Also we need to involve regulations that apply to deferring the load from Ofgem. We put these issues under the category of constraints that need to become part of the role of the controller.

8.3.2 Assumptions

- In the review it was pointed out that the assumption about storage state of charge (SOC) should be considered as an operational constraint and should be removed from the assumptions.

- While examining the flow chart, the reviewers were told that storage status is checked first before importing or exporting energy to the grid. We had assumed this because our network consists of renewable energy generation sources and it was suggested that they should be fully utilised. The reason for this is to reduce the use of brown energy and increase the use of green energy.

  From the reviewers comments we could identify that this assumption needs be explicitly mentioned in the use case document. Because this is linked with cost involved in using brown and green energy. This factor will become important when we will extend our model by adding the functionality that calculates the cost involved in using both type of energy.

- ***The assumption is that the network is working in normal conditions but if it is not, then what indications does it have?***

  It was recommended that we make this assumption clearer by explaining that the use case assumes that there are no faults present in the network. This point was discussed in detail in previous section.

  In the assumptions we have mentioned that the network is working in its normal mode. However, it is not explained in terms of faults which is more

of an engineering point of view. Further we have mentioned two situations when network is not in a normal condition: one is an islanding mode when the SSEZ is disconnected from grid and there is more energy in the zone. The second situation is when wind is too high to run the wind turbines and the extreme measure taken is to shut down the wind turbines.

8.3.3 Design

How are the demand, generation, storage and weather services identified? Further, how they are connected to each other. The purpose is to get the big picture of the design model.

In the current design, the controller is the main component that connects all of the services. This question was raised in the first session of the walkthrough. The design was presented through a whiteboard. The identified services were presented, but the reviewers were unable to relate to them. For this there was a need to start from a more abstract level by providing a big picture of the system components, and then discussing each component later. In the second session the use of a Powerpoint presentation helped to represent the system with different levels of detail and the context for each diagram.

Why network constraints are not listed in the identified services?

The network constraints are more related to the functionality of the controller. Further there is no outside entity involved in this functionality, therefore this information was considered as being at a second level of detail. This is useful when controller functionality is discussed and we move towards detailed design and realisation of the energy system. Further, this aspect is important from the engineering perspective. As we discussed earlier, we have made the assumptions that network is in a normal operating condition and have avoided low level details about network assets, hence we have not collected further details about this.

As we noted, not all functionality in the application is taken from a third party and there are services that are local to the application. In the case of technical constraints, we consider these to be local to the application and not a service that is owned by a third party. Also, the controller is not a service, rather it is a functional component. If we decide to make this as service and used by other

ESCOs then we have to consider how different network configurations associated constraints and operating policies will be provided to the controller to evaluate the power balance for these networks.

From the reviewer point of view, it would be nice to include this in the current design, and to mention network constraints explicitly in the identified functional components. For the software development point of view, this is a category of constraints that become the part of functional component 'assess level of change'. In this module we add operating policy, priorities and constraints. So if we explain this functionality in detail then we need to break these constraints down to a further level of detail.

The system log is mentioned in the design, however, in the use case document no information is specified for this.

We considered the system log to be an implicit requirement, as it is important for the ESCO to know what decisions it has made in the past, and it can use this later in its prediction model. This information is now explicitly mentioned in the use case document.

Why is the weather service considered at a high level? It may not be required at this point of functional component identification.

Design is an iterative process and it is difficult to explain each time which functionality is identified at what level. Many functional components become visible during requirements analysis and further are added during design process.

We considered the weather service important in terms of its use in the system. The current weather state is included as part of system state and the forecast state is considered important, as the control system involves a prediction model. Further to this, the weather service is not provided internally. This service is required from a third party and so we considered it important that this should be made explicit in the case of service based application design. The functionality that is required from any third parties, and its use in the system should be considered at the early stage of the system.

In the design we also have to consider the availability of this service from different service providers, the exceptional cases and the dependencies upon other system functionality needed for this. For this reason we considered it appropriate to include this from the beginning. Even if at first its only goal was to keep

the weather history about the energy zone and use this further in the prediction model.

If generation prediction and demand prediction services are provided by service providers then why is the ESCO communicating with the weather service itself? Why is weather data coming from other prediction services not considered enough?

We considered it important for ESCO to communicate with the weather service because

- It uses renewable sources and needs different levels of detail from weather services.
- It needs to use the weather forecast in its prediction model.
- The detailed weather information might not be available through the generation and demand prediction services. They will provide the information that is necessary for their own functionality.
- As each service provider have a different type of service available, so we might at some point need to provide information about the weather condition in our energy zone.

Why are two scenarios shown for demand and generation prediction in activity diagram?

We have shown both the situation when these services are constructed by ESCO itself and when they are taken from a third party. There is the possibility that these services might be developed by the ESCO and offered to other ESCOs over time at a price.

There are operational goals mentioned in the use case but it is not clear which ones are considered by the controller.

Th operational goals are not discussed in the design. While representing system behaviour, we have considered three situations: when to import, export and when to remain self-sufficient. These operational goals are related to the operations of the controller. We consider operational goals as part of the ESCO policy and the priority that it sets. The ESCO can consider that its priority is to

make money in the market as the highest priority or it wants to fulfil customer demands first, and for that it decides to import brown energy when required instead of deferring the loads. We consider these as details to be included when we start to discuss the detailed design.

In the current design we have made the assumption that the ESCO will try to be self-sufficient and fulfil its customer demands first. That is why the use of storage is considered first before importing or exporting energy. We have made this assumption as in the case of renewable energy, it is recommended to utilise internal resources first. Secondly it is an important point to be considered when it comes to calculating CO2 emissions in the zone.

8.3.4 Conclusion

In this section we sum up the evaluation on the data collected from the video recordings and the interviews taken from participants.

Requirements: The requirements are understandable, and the concept of the SSEZ were correctly understood and explained. The problem was captured sufficiently. However, there were inconsistencies in the use case documents and there were some technical features that needed to be corrected. For example the assumption used about the storage state of charge (SOC) contained different figures in different sections. The use of terms energy and power need to be used carefully as both have different meanings. The storage capacity mentioned in the network configuration section have inconsistent values that need to be resolved as it is associated with storage technology and can easily be misunderstood.

The details in the use case need to be represented in such a way that it will become easy for experts from both domains to understand the problem.

Assumptions: The assumptions are valid apart from the decision made about the storage SOC. That is an operating policy issue rather than an assumption. More assumptions need to be specified about the network, such as using the network at full capacity will not violate technical constraints.

Further, the assumption that the demand prediction service will have knowledge of the network is important. Also the demand side management (DSM) is considered only in terms of storage, DSM on customer loads is not considered.

Design: The design decisions need to made explicit. The design and use case contains different levels of detail, and there is a need to define the scope that what is considered in the design. In that case the design and use case have inconsistencies.

The two possible scenarios presented about demand and generation prediction need an explanation that which is considered as final design decision. The demand prediction service should involve weather data which is not shown in the design. Further all identified services should be made visible in each diagram such as data flow diagram should represent all services.

8.4 Summary

In this chapter, the evaluation process and its outcomes are discussed in detail. The evaluation process consists of walkthroughs and interviews. The details are provided about each section. Further the reviews about requirements, assumptions and design are discussed. Finally the output of the review has been summarised in the conclusion section.

Chapter 9

Discussion

9.1 Introduction

The chapter discusses the research described in this thesis. The research is explained through the use of narrative synthesis, a technique that can be used to explain and summarise the findings of qualitative research (Cruzes and Dybå, 2011). In the last section we discuss the threats to validity associated with the research methods employed.

9.2 *Why use a multi-method approach?*

A multi-method approach is employed as this is able to provide a wider coverage of a problem space. The purpose is to investigate the phenomena using a combination of empirical research methods that increase the reliability of the study when compared to single method studies (Wood et al., 1999). This approach is described as method triangulation by Miller (2008), and can be used as a strategy for knowledge discovery as well as providing a way to deal with the limitations associated with a single empirical study.

We have used the combination of a mapping study with a case study and walkthrough. These research methods were selected in an evolutionary manner, since the phenomena under investigation were relatively new and little experience was available. In Table 9.1, we briefly explain the rationale for their selection.

Table 9.1: Empirical Methods employed in the thesis

Research Method	Purpose	Objective in this research
Mapping study	A systematic and objective way of identifying evidence related to the problem area (Kitchenham et al., 2011)	Used to investigate the SOA concept in the existing literature.
Case study	To answer 'how' or 'why' questions and to understand the phenomena in depth and in a real context (Yin, 2008)	Used to develop the use case and explore the SOA design.
Walkthrough	To evaluate the work for consistency and accuracy by employing expert knowledge.	Used to evaluate the case study outcomes.
Interviews	To get in-depth and first-hand understanding through the cycle of plan, act and reflect (Oates, 2005; Sjoberg et al., 2007)	Used with the walkthrough to analyse the effectiveness of the process.

The mapping study was performed on the topic of SOA, and the associated synthesis was focused upon the way that an SOA is described. This was performed qualitatively by carrying out a thematic analysis (Cruzes and Dybå, 2011). For this, the qualitative data consists of words and pictures, and the benefit of using a qualitative method is that it requires the researcher to explore the complexity of the problem (Seaman, 1999).

The mapping study was conducted on a non-empirical topic, making it different from most other such studies. While performing the mapping study, we observed the evolution of the SOA concept through an increasing number of definitions starting from the year 2000. The selection of papers and the synthesis process was also different. This is because we were looking for a description of an SOA, and there was no clearly defined section in the papers for this, unlike experimental results. The definition could appear anywhere in a paper, sometimes in the abstract, introduction, background or at times, in the discussion section. This required a detailed scan of each paper. Another interesting element involved was the analysis of definition sources that included a significant amount of grey literature. The outcomes of the mapping study include an SOA model

that integrates the key elements of SOA, and also provides a classification of the terms used in its definitions.

The second method we used was a case study. A case study provides a means of analysing a phenomena in depth and in its real context (Yin, 2008). The case study employed in this thesis provided the means to understand and explore the application of SOA for a real world problem related to energy engineering. Further, it involved constructing a design model for the selected case. Hence, the case study has been employed for an exploratory purpose, in order to gain insight into the application; and to explore and generate a design for this.

In an exploratory study, multiple sources can provide higher reliability than a single data source (Bratthall and Jørgensen, 2002). We used multiple data sources for the case study data. The extraction of information from different sources and integration of these pieces of information into a coherent form was quite challenging. This involved the understanding of the domain, the use of requirement elicitation techniques, especially to identify the use of right terms from energy engineering and to avoid vocabulary used in software engineering, and understanding and adopting the method useful to acquire information from engineers such as reference model of electricity network with some configuration details. We can identify this as the issue of 'multiple perspectives' described in (McLeod et al., 2011). When a reference point is not available, engineers may well provide different information and answer the question differently, which is usually a reflection of their own area of expertise. Also, the senior engineer usually provides a 'big picture' of the problem that includes the themes of their research plans. Therefore, to transform a big picture into a real model by integrating information from different sources and by using interviews required significant effort.

From their experience of conducting a case study, McLeod et al. (2011) reported on the 'relationship between research and researched'. This phenomenon is described for ethnographic studies when field researcher has close engagement with the research context. The researcher has dual roles of being a participant experiencing the research context and of being a researcher, observing, analysing and interpreting it.

In our case study, we also had close engagement with the 'case' we selected

and so we too had the dual role of participant and the observer. This added to the complexity of the task. Being the researcher requires that we have to be vigilant to ensure that the outcome should not end up as an energy thesis. During the requirements elicitation process, the researcher was fully engaged into understanding the problem domain. This can raise the risk of solving the application domain problem rather addressing our own research question. Therefore, in this time period, the researcher had to repeatedly consult the objectives set for the research and discuss these with supervisor. As an observer, we had to report on the phenomena, and had to document our own experiences and the lessons learned from the study.

McLeod et al. (2011) reported their experience of observing software development process as 'researching in action'. We use this to describe our own experience of generating requirements and constructing a design for these, since we have covered the requirements and design phases as part of the case study. Being participant and the observer, this phenomena equally applies on our case study. We participated as designer in constructing the SOA design model, and in doing so made use of design methods, strategies and notations. We also arranged discussion session with experts having background in cognitive sciences and notation design. In particular, we invited Professor Thomas Green and his team to advise on our study.

Each case study establishes its own internal logic and design principles (Perry et al., 2004). We could not find any direct comparison in the literature with the study discussed in this thesis. Further we wanted to evaluate case study in terms of identifying how accurately application domain concepts were captured and stated in the use case, and the appropriateness of existing notations in SOA design.

The validity of the case study refers to the reliability of the results (Runeson et al., 2012). We have evaluated the case study results by employing peer review (in the form of a walkthrough) a form which was used to evaluate the technical content and quality of the work (Garousi, 2010). The use case validation was performed by involving an expert from energy domain, as the design evaluation required both application domain knowledge and the software design. The walkthrough process itself required feedback from participants. Therefore, an element

of action research was introduced into the evaluation process.

The multi-method approach adopted for this thesis proved able to provide a broad coverage of the problem area. By selecting different empirical research methods, we were able to address the problem in significant detail. The research methods complemented each other and compensated the weakness inherent in the individual method (Wood et al., 1999). We have used the outcomes of one research study as an input for the other, with each research method addressing its own research question by making use of the results from the previous study.

9.3 Related Work

In this section we provide a brief overview of some work by others that can be related to this study.

9.3.1 The Mapping Study

We conducted our mapping study to address the issue of the *concept* of an SOA. We used 98 studies out of 701 that were examined in full. Data synthesis was through thematic analysis, leading to a set of fifty different terms. From this, we have produced an SOA model that provides key concepts, description and classification of terms used in the literature. We have then made use of our model by mapping its features with those of a real world problem.

The study by Boer and Farenhorst (2008), has raised a similar issue about the lack of commonly accepted definition in the area of architectural knowledge. To examine the terms employed in the published literature, and to identify different definitions available on this topic, they conducted a systematic review. They found 14 definitions out of 115 studies. Instead of a qualitative analysis, they used a quantitative approach termed reciprocal translational analysis. The details of this were not available. The study did not explain the analysis process used and did not report any threats to validity. The outcomes and observations are available. The study concluded that researchers should be precise and concrete in defining the concepts that they consider are part of the architectural knowledge. This result is similar to what we have observed from the experience of conducting

mapping study on SOA.

9.3.2 The Case Study

We have compared our case study with others in terms of addressing the issue of a real world problem, by analysing similar case studies that report on their experience.

A real world problem: The case addressed through the use case, in our case study, is constructed on a real world problem. The information has been collected from domain experts and composed into an operational model. The use case discusses the problem in detail. Further it was evaluated by application domain expert. This makes our study different from many others, where researchers have used artificial examples and addressed them as case studies. For example, a recent edition of (ERCIM-NEWS, 2013) addresses the special theme of smart energy systems. There is another study published in (Venables, 2012) about the effective use of distributed generators in the form of smart grid at the time of 'Hurrican Sandy'. A similar study which is also covered in our case study is published in (Muller, 2012), however they provided it as a future picture of renewable energy system.

Studies such as those by (Baresi et al., 2005) and (Dietrich et al., 2007) employ examples for a case study from the vehicle control domain and the shoe industry. The first one is a self-created case and used as laboratory experiment. In the second one, the author describes a general case of a supply chain. Similar studies (Gao and Tang, 2007) and (Duan, 2009) are from the textile industry and mobile networks. The study by Bakker and Iacob (2009) explains the case of a health care insurance company. The paper is part of ongoing research and information is incomplete. Bosnjak et al. (2011) have described a case study about ocean energy information management. The study describes a preliminary investigation and reports that no actual or functional domain-specific web services have been created.

Recently a study by Espinha et al. (2012) has discussed the issue of lack of real case studies in the SOA literature. They conducted a short survey of

the case studies used by researchers. They classified many of the studies as self-created and closed systems. Self-created case studies are the ones that are used for laboratory experiments and consist of self-generated scenarios, while closed system case studies are ones published by researchers working on industrial projects and therefore do not provide internal details. To cover this gap, Espinha et al. (2012) have suggested a case study which is an extension of an existing application called Apache Stonehenge. The extension is a replica of the original system and makes use of an open source platform called Turmeric SOA.

Case Studies in Software Development: Moe et al. (2010) describe a case study that was conducted to analyse teamwork in agile teams. Their study was conducted over the time period of nine months in a software development company. The study was based on observations and data was collected through semi-structured interviews and informal meetings with the team members. The study was interpretative and results were compared with previously constructed theory.

McLeod et al. (2011) also report on their experience of conducting a longitudinal case study on software development. The study was to investigate software development practices in its organisational settings. The study spans over two years, and data was collected through interviews with participants. We categorise this as an observational study as the author was not involved in the development activity. Observational studies produce qualitative data, and in case of this study the author has categorised it as an interpretative case study. Therefore, no comparison is made with any other studies.

Our case study differs from the ones summarised above. In our case study, the author contributed by being participant and the researcher. The construction of the case study includes software development phases such as requirements and design. We can categorise our study as being exploratory. We have used method and data triangulation to generate and collect data. The reliability of the outcomes (Runeson and Höst, 2009) has been addressed through the evaluation of the case study.

9.3.3 The Walkthroughs

We have employed a walkthrough for the evaluation of the case study. In this way, we used peer review in an academic context for the purpose of evaluation of our case study. The current literature, includes some studies where such a peer review technique has been used for the academic purposes. Zeid and Elswidi (2005) have used peer reviews in teaching object oriented analysis and design (OOAD) course, while in another study by Garousi (2010), peer reviews have been employed for design projects in a software engineering course.

9.3.4 Use of a Multi-method Research

- The study by Wood et al. (1999) employed a multi-method approach for an empirical investigation of object-oriented technology. For this, the study was divided into three phases. For each phase a research method was selected, such as semi-structured interviews, questionnaires and laboratory experiments. The first two phases involved participants with a background in industry and academia and with experience of working on object oriented technologies. In third phase, participants, were students who carried out experiments.

- Mingers (2003) reported the findings from performing a survey to explore the extent to which a multi-method approach is used in the information system (IS) literature. The study identifies little evidence of multi-method research published in this area such as empirical papers published in IS journals. Overall, only 20% used a combination of methods. One of the reasons for this, as identified by author, is the time and cost factor involved in employing different research methods.

- From our experience, we can identify that a multi-method approach does provide an opportunity to explore the problem in depth and from different perspectives. At the same time we agree with Wood et al. (1999) that each research method does require proper planning and design.

9.4 Threats to Validity

As we have used a multi-method approach to conduct the research discussed in this thesis, the threats associated with each research method will differ from each other. We discuss these identified threats separately for each research method.

9.4.1 The Mapping Study

The main threat to validity to be considered for the mapping study is that of *internal validity*, since this is a secondary study and does not involve human participation. Further, we have used a mapping study for a non-empirical topic which makes this study different. In such a situation, *construct validity* could also be an issue. However, systematic review techniques are now relatively well established, therefore we do not consider this to be a significant threat.

Internal Validity

Search Strings: It is possible that we have missed papers that discuss definitions of SOA and that do not use this term explicitly. However, we did prototype our search strings carefully, and it does seem unlikely that the term would not appear in a paper that was using a definition, hence this might be considered as a fairly minor threat.

Search Coverage: We have used three of the major electronic databases (IEEE, ACM and Science Direct) for our search. Normal guidelines are to use around four for a systematic literature review, but in this case we drew upon experience to reduce the number of papers that had to be sifted. However, in the later iterations, this sifting did involve following up the references of the papers found (snowballing), and this process did not point to any significant groupings of papers that we had missed.

Analysis of Papers: Guidelines on performing SLRs suggest that data extraction should be performed independently by two analysts, or that if one analyst is used, a percentage should be checked. However, this relates to

extraction of data from papers reporting empirical studies, which can involve the analyst in having to perform quite a complex task and exercise judgement. In this case, we were solely concerned with references to definitions, for which true / false decisions are relatively straightforward to make and so all data extraction was performed by one person (the author). The analysis of definitions was, however, conducted by the author and the supervisor. To check the agreement level between these, a Kappa test was applied, as discussed in Chapter 5.

The search covers papers to the end of 2009, a period in which ideas about SOA were evolving. An informal check of subsequent publications on this topic does not suggest that there have been any significant changes over the past two years.

9.4.2 The Case Study

For the case study we can identify two types of threat: *internal validity* and *external validity*.

9.4.2.1 Internal Validity

- Use of method triangulation involves multiple techniques used to collect and produce data. This was done by constructing an operational model in the form of a use case; producing a design by using existing design techniques and knowledge; and by performing a walkthrough to check the validity of the use case and the SOA design model.

- Multiple data sources were used, such as formal and informal interviews with application domain engineers, analysis of technical papers, and use of supporting documents. Further, we held sessions with experts to discuss the issue of notations for SOA design, and conducted an expert review to evaluate the case study.

9.4.2.2 External Validity

- The case study employs a single-case design, a choice that raises the possibility of bias and generalisation. We have used multiple sources of data and evaluated it through an expert team that was not involved in any stage of case study construction. Bratthall and Jørgensen (2002) have noted that use of multiple data sources in an exploratory case study makes the case study more trustworthy than the one based on single source of data.

- Being single-case, the issue of uniqueness and special access to resources applies to this case study.

9.4.3 The Walkthrough

For the walkthrough, bias is an important issue to be considered. In the case of systematic reviews the protocol and guidelines are well established and have matured over time. Therefore we can find conference series (such as EASE) addressing these issues and a continuous feedback going into this area. In the case of the case study we do find examples of protocols and guidelines, however, the range of 'types' of case studies usually employed in software engineering is a bit limited. This may be because case studies in other disciplines are often longitudinal studies and require significant time to observe the phenomena (Yin, 2008). However, we did not find a proactive approach towards this, in the form of dedicated conferences and workshops, to generate literature on case studies alone. There is a recent book by Runeson et al. (2012) on case studies and guidelines are available at (EBSE, 2013).

The literature on walkthroughs is largely available in publications dating from the 1970's and 1980's. There is a book by Yourdon (1989) on structured walkthroughs that provides some guidelines in its appendix about the structure of a walkthrough. There is another term 'cognitive' walkthrough used in the software usability area. However, we did find a lack of empirical studies that on reported experience of conducting walkthroughs and about their use in design evaluation.

For the walkthrough we could identify two main threats: *construct validity* and *internal validity*.

9.4.3.1 Construct Validity

The guidelines available for performing a walkthrough are either quite old and abstract or they are not supported by evidence. Therefore, we have made use of the guidelines available at (EBSE, 2013) and those mentioned in (IEEESTD, 2008). We tailored these to write the protocol for walkthrough. This was done in consultation with the supervisor. Further we have employed ideas from action research by taking feedback from the participants about the walkthrough process as suggested in (Seaman, 1999). This provides confidence that the form of the protocol and the walkthrough process were appropriate. Further, a questionnaire was developed to help participants provide feedback, and also to help achieve the objectives of the walkthrough.

9.4.3.2 Internal Validity

This study involved human participation, therefore, we consider bias as an important factor to be discussed here. The participants had backgrounds in different domains and with different levels of knowledge about the problem under discussion. The possible sources of bias were identified and listed in the protocol.

Selection of Experts: Two experts were involved in the study, one from each domain. This could be considered a problem in terms of ignoring any major issue during the review. However, the support of the questionnaire which was verified by the supervisor, was provided to ensure we did not ignore any important point.

Further, the case was constructed with the help of energy engineers and feedback was taken during requirement gathering to verify that what is written is understood by the author and meaningful to engineers. This does not violate the guidelines, where it is suggested that the number of experts involved can be restricted to two.

To get an expert from energy engineering the energy group was contacted and they identified the experts that had experience in the control system and also knew about the breadth and depth of renewable energy domain. This was important as researchers, in practice, focus on a branch of a specific

problem and at times find it difficult to comment on related work. We identified this problem while interacting with engineers who were involved in different areas of renewable energy research. The selection of an expert was done by energy group supervisor who knew about our work and we were not involved in this. The expert from computer science was selected on the bases of his experience in SOA in particular and in software design in general.

Involvement of Experts during Preparation: None of the experts was involved at any stage during the preparation of use case and design.

Data Consistency: There is a threat to data consistency while taking notes by the observers, which can be controlled by assigning more than one observer at the time of the inspection (Seaman, 1999). We have handled this issue by keeping a record of all sessions in the form of audio and video files. This reduces the chance of inconsistency in data collection. This also helped in analysing the sessions and viewing the discussion in its context.

Need for two walkthroughs: The evaluation process was performed by employing two walkthroughs and following interview sessions. This structure was constructed by focusing on the point which we mentioned earlier that the participants were having different backgrounds, experiences and understanding of the problem. The documents included Chapters 6 and Chapter 7, and the review protocol (Appendix E), and were made available to the participants before the walkthrough sessions to inform them about the background. Also the first walkthrough session mainly served the purpose of being a preparatory session.

Analysis: The analysis of the walkthrough was done by the author alone. The recordings were available and questionnaires were developed to aid these sessions and interviews. The categorisation of the discussion and question asked did not create any problem. Rather these helped in classification. The reviewers used these classification when they wanted to mention a point relevant to previous category. Further, the supervisor was involved in the preparation and the process of evaluation, although he was not present

during the walkthrough and interview sessions. The issues were discussed with him and the details about evaluation discussed in Chapter 8 were reviewed by the supervisor.

9.5 Lesson Learned

Some key lessons learned from this research process are listed below.

- **Lesson 1: Identify the availability of third party services and any requirement for new services. Also consider the ownership of new services.**

  The availability of third party services, any need for new services, their ownership and associated contracts and the requirements for the registry should be identified from the very beginning of the SBA design.

- **Lesson 2: Application domain constraints need to be mapped with the technology at the early stage of SBA design.**

  The application domain constraints restrict the design in certain ways. Therefore, the SBA design process for each domain will be different. In our case, the SBA design model was developed for a real time application, which is different from ones that involves a purely business scenario, such as purchase and shipment from Amazon.

- **Lesson 3: Design decisions are key elements for SBA development.**

  For SBA development, there is a need to be very clear from the beginning about the service model to be used. By this, we mean the understanding of the SOA model. We have argued in this thesis that SOA is an emerging paradigm and its concepts need a shared understanding. In the absence of this there are different interpretation of these terms, influencing the design of any applications. Therefore, for SBA development, decisions about the third party services and the long term business planning for newly developed services is essential.

- **Lesson 4: The documentation needs to clearly specify terminology.**

  This mainly stems from the problems that occur in interdisciplinary research. Each domain has its own vocabulary which incorporates implicit knowledge from the domain. Dealing with both energy engineering and computer science disciplines requires a shared vocabulary, which we have provided in the form of a glossary. This is also needed to make the document clearer for both audiences.

- **Lesson 5: A 'dry run' for the design walkthrough sessions is an effective way of dealing with the needs of interdisciplinary research.**

  This was undertaken by making two walkthrough sessions for the design evaluation. Clearly, the first walkthrough session provided experience and confidence of dealing with both domains.

9.6 Summary

In this chapter, we have discussed the research conducted throughout this thesis. We have explained the reasons behind employing a multi-method approach and the selection of different research methods. We have analysed studies that relate to each research method. We have discussed the threats to validity associated with each research method. In doing so we have provided a holistic view of the research along with our interpretation.

Chapter 10

Conclusion

The chapter summarises the findings of the research discussed in this thesis. Each research study is summarised, and used to answer the research questions posed in Chapter 4. The contributions made by this research are also explained, togather with some ideas for possible future research on SOA design.

10.1 Thesis Summary

This thesis describes a multi-method study based upon using a mapping study, a case study and a walkthrough. The mapping study was performed to investigate the use of the SOA concept in existing literature. We considered it important to explore the concept in depth as the literature on SOA provides different interpretations of this concept and the term is often used in an ad hoc manner. In the mapping study, we identified 921 studies on SOA, after which we selected 98 studies for analysis.

We investigated the descriptions provided in the text and extracted the definitions. The terms in the definitions used to explain SOA characteristics were tabulated against each definition, providing a set of fifty different terms. This was then used to construct an SOA model that classifies the terms and groups them against a set of more concrete identifiers. To avoid multiple interpretations, as we found in the mapping study, we provided description to each of the key identifiers.

By analysing the definition sources, we found that many definitions are taken from vendor-specific web sites. The most prominent were from W3C, IBM and OASIS. Thomas Erl and M. P. Papazoglou were are also cited increasingly, however they never offered definitions in their own texts. The definition that was found to be consistent over time and was cited without changes in the text was the one offered by OASIS.

The case study covered the process of developing of SOA design model and associated operational model and the evaluation of both of them through a walkthrough. We took the problem from the domain of renewable energy. The use case provided an operational model of a control system for small scale energy zone (SSEZ). The SOA design model was constructed by integrating design knowledge from the software design theories, SOA and software architecture.

At the final stage we performed two walkthroughs as a part of the case study evaluation, to evaluate the outcomes of the case study through expert knowledge. We also investigated the effectiveness of the walkthrough process for this type of model.

10.2 Research Outcomes

In Chapter 1 we described a set of research objectives for this thesis. The research questions identified and addressed through each research method were then discussed in Chapter 4. Below we briefly explain how each question is answered through the set of studies.

1. **Mapping Study**

   What are the key characteristics of a Service Oriented Architecture?

   We answered this question by employing a mapping study discussed in Chapter 5. We identified 921 studies through the search strings we defined to find relevant studies from electronic databases. We selected 701 of these after applying inclusion and exclusion criteria, and thoroughly examined these studies to find any SOA descriptions in the text. A full analysis of SOA definition was carried out on the final selection of 98 papers.

We have described analysis process in Figure 5.4.

2. **Case Study**

   The case study was conducted according to the protocol described in Appendix B. Questions were:

   "Can the characteristics of an SSEZ control system be successfully modelled through the construction of a use case model?"

   This question was addressed by constructing an operational model of an SSEZ control system in the form of a use case, as described in Chapter 6. The use case was developed as part of the case study process. The use case contains the information about the configuration of SSEZ electrical network. It explains the operational goals and the data sources inside and outside the control system. The use case characteristics were mapped with the SOA model constructed through the mapping study. This demonstrates that the selected use case, as a representative example, can successfully model the SSEZ control system.

   "How can SOA attributes be modelled using abstract diagrammatical forms?" and ***"How can such abstract models be developed?"***

   In Chapter 7, a high level design model was successfully constructed based on SOA features. The process of developing the design from requirements, and presenting that through a set of notations and existing diagrams is explained in detail. A number of scenarios from the use case were used to explain different features of the control system. The design model was constructed by integrating knowledge of software design, SOA and software architecture. As such, this demonstrates one approach to developing an abstract design model.

3. **Walkthrough**

   A protocol was developed for the walkthrough Appendix E to evaluate the use case and the SOA design model.

"Are the design, and the notations used, appropriate for the construction of an SOA model for the SSEZ control system?"

In Chapter 8, the evaluation process is explained. The process consists of two walkthroughs with semi structured interviews being conducted after each walkthrough session. We have reported our experience of organising the walkthroughs and discussed the lessons learned from this activity. We addressed the research question by evaluating the requirements, assumptions and the SOA design model, concluding that these forms were appropriate.

10.3 Contributions

- The mapping study has been conducted on non-empirical topic. This makes it different from previous studies, both in the evidence-based literature and the services literature (Anjum and Budgen, 2012*a*).
- The mapping study identifies the need for a commonly accepted vocabulary for SOA that can be used by researchers to explain SOA concept. This will help limit multiple interpretations of this concept and assist designers in communicating their ideas.
- The case study was used to develop a real world problem. The use case can be considered a representative example for service based application design. The use case was developed with the help of energy engineers and is evaluated for consistency and accuracy.
- The study provides the experience of constructing an SOA design model by employing existing diagrammatical forms on a real world problem (Anjum and Budgen, 2012*b*). This will aid in filling the gap that exist in the literature about service based application design. This also raise the issue of new notations required for service based application design and providing an example that is not based upon a 'toy' problem.

10.4 Future Directions for Research

Future work is needed in the following directions.

- There is need to devise notations that can express all of the characteristics of an SOA. For this, our integrated model of SOA can be used. Further to this, for notations interpretation rules need to be defined (Wieringa, 1998) instead of borrowing existing notations especially from the UML, where all forms of diagram are not used very widely (Budgen et al., 2011)

- The research could be extended by including non functional properties at the design level. Right now, cost and time are considered as a constraint for service selection. However, to progress to detailed design will require the choice of implementation technology. That will help to evaluate quality attributes of performance and reliability.

- The design could be realised by developing the necessary web services. This will help to analyse the real time response of such system. This can be done in two ways. First by considering fixed service providers for each service, to help with in realising the system as a set of services, and so that the performance and time can be calculated.

  Second by adding multiple service providers. This will involve the addition of a service registry to keep service provider information. In (Bianco et al., 2007) this is considered as dynamic binding. In this case contacts and interfaces are pre-negotiated by service providers. The information about non-functional features like time and cost will be available with the information about how to invoke the service. At runtime the services will be selected on the bases of time and cost from the registry and then will be invoked to get the desired functionality.

  The comparison of these two realisations will help to assess the most suitable form of solution for service based applications in this domain.

- The case study can be extended by involving multiple service providers for energy generation that will be other ESCOs. They can offer different prices to buy and sell energy. Instead of buying brown energy from national grid,

an ESCO could decide to buy green energy from a neighbouring ESCO. In this case an important need will be for publication of such services from an ESCO. For this reason instead of a private registry, a broker service may be required.

10.5 Summary

The main contribution of this thesis has been the systematic analyse of the concept of service oriented architecture (SOA) and to design a real-world service based application (SBA) for energy control system. The widely published literature on SOA represents the popularity of this concept in software community. However, evidence for its applicability to real problems is lacking in the literature. Also, experience of developing service based applications and the processes followed are not discussed in the literature.

The research has contributed by exploring the concept of SOA taken from the existing literature, constructing an integrated SOA model, and exploring the design of service based application through a real world problem. The work can be extended by adding further detail in the case study and the design model to develop a full functional model.

From this research we conclude that to fully exploit the benefits of the SOA, the research community needs to share experience of developing service based applications with evidence from real world case studies. For this, the first step would be to develop a shared understanding of SOA concepts, introducing notations that can provide semantics and syntax to represent these concepts and finally the realisation of SOA design models in a real-time environment.

Appendix A

A.1 Search String

Due to the lack of standardisation of search interfaces in IEEE and ACM databases, the following search criteria were implemented:

A.1.1 IEEE Xplore

- Search type: keywords or phrases search in all fields (titles, abstract, etc.)
- Date Range: 2000-2009

A.1.2 ACM

- Search type: Word or phrase find [any field] with all of this text (search strings).
- Publication year: 2000-2009
- Publication Type: Journal, Proceeding, Transaction

A.1.3 Science Direct

- Search type: Search string in Abstract, Title and keywords
- Date Range: 2000-2009
- Source: All sources

- Subject: Computer Science
- Publication Type: Journal

Appendix B

B.1 Case Study Protocol

B.1.1 Change Record

Table B.1: Change Record

Version	Change
1.0	initial draft
1.1	new sections
1.3	modification in different sections

B.1.2 Background

Case study methodology is an empirical flexible design study used to understand a certain phenomena or to construct a theory. According to Yin (2008) " a case study is an empirical inquiry that investigates a contemporary phenomenon especially when the boundaries between phenomenon and context are not clearly evident". The case study has become a popular research method in empirical software engineering and is used in the literature to understand, explain or demonstrate the capabilities of a new technique, method, tool, process, technology or organisational structure (Perry et al., 2004).

Studies using this methodology range from very ambitious and well organised studies in the field, to small toy examples that claim to be case studies (Runeson and Höst, 2009; Runeson et al., 2012). The latter form is quite prominent in the

area of service based applications (SBA) where artificially constructed examples are widely used. Therefore, this research has sought to use a real world case study, based upon an energy engineering use case, in order to address the problem of service based system design as rigorously as possible.

The extensive literature on the service oriented architecture reflects academic and industry interest in this area. The efforts to implement this concept can be seen in the form of solutions devised in different domains, especially telecommunication, health care, and device automation. A problem with these models is that they all claim the uniqueness of their methods. However, the process of *developing* a service based solution, particularly its design and the evaluation of proposed models is largely absent. SOA solutions are explained through "toy" examples such as travel planning, car rental, and online purchase, without any evidence of either how they were derived, or of their real time performance. Also, the scope of these examples is too narrow to be used to address the research problem discussed here. It is therefore hard to say how these methods will work in real time and what will be their underlying strengths and weaknesses. It is also difficult for both architects and developers to trust and adopt such solutions without having much evidence about their run time characteristics.

To find a use case with characteristics similar to those of SOA as identified through the mapping study (Chapter 5), we examined several different domains such as economics, stock exchange, electronics, and some branches of engineering. Domain experts were contacted within the University to find their views and the possibility of using their domain knowledge. Initially, the stock exchange was considered as a suitable domain, but without the support of a suitably interested domain expert it was not possible to adopt this. Most responses came from the School of Engineering & Computing Sciences and a use case from the Energy Group was eventually selected as they are also interested in investigating the usefulness of service-based solutions for the industry.

The overarching research question for this case study is:

"Can the characteristics of an SSEZ control system be successfully modelled through the construction of a use case model?"

- By 'characteristics' we mean different aspects of the SSEZ control system

that include functions it performs, the processes it run to complete the tasks, the organisation of its resources etc..

- 'Successfully' means we are able to capture and describe the SSEZ control system features accurately.
- By 'use case model' we mean the configuration and operational details of the SSEZ control system.

To construct a use case, the case must have following features:

- Services are available both locally and remotely with (possibly) multiple providers being available for supplying a remote service.
- The configuration of service assembly may need to be changed at run time (through run time discovery, selection and composition).

The next section, provides a brief outline of the main characteristics of the energy system.

B.1.3 Energy Systems

The term *renewable energy* means energy derived from a broad spectrum of resources, all of which are based on self-renewing energy sources such as sunlight, wind, flowing water, the earth's internal heat, as well as biomass such as energy crops, agricultural and industrial waste, and municipal waste. These resources can be used to produce electricity for all economic sectors, fuels for transportation, and heat for buildings and industrial processes (Bull, 2001).

Affordable, sustainable and reliable energy supplies are key objectives of the U.K. government's energy policy. Currently, the two long-term energy challenges are to tackle climate change by reducing carbon dioxide emission both within the UK and abroad and to ensure secure, clean and affordable energy forms to reduce dependency on imported fuel. If CO2 emissions are to be reduced and in particular, if they are to be reduced by around 60% by 2050 as suggested by the Royal Commission on Environment Pollution (RCEP 2000), then most of the

existing electricity generation capacity in Britain have to be replaced (Soni and Özveren, 2007).

The integration of renewable energies into the electrical power supply is of growing relevance. Hybrid energy, which is mainly based on the contribution of locally available renewable energy sources, represents an innovative and sustainable solution for decentralised and remote power supply (Soni and Özveren, 2006). That is the reason the role of Energy Services Companies (ESCOs) is becoming very important. These companies govern and manage small scale energy zones (SSEZ) that supply locally generated electricity to their clients in the industrial, commercial and domestic sectors. Bertoldi et al.,(2007) describe an ESCO as:

"a natural or legal person that delivers energy services and/or other energy efficiency improvement measures in a user facility or premises, and accepts some degree of financial risk in so doing. The payment for the services delivered is based (either wholly or in part) on the achievement of energy efficiency improvements and on the meeting of the other agreed performance criteria."

In a further discussion, Bertoldi et al., (2007) describes the key characteristics of an ESCO as:

- Guaranteeing the energy savings and/or provision of the same level of energy service at lower cost;
- Remuneration is directly tied to the energy savings achieved;
- It can either finance, or assist in arranging financing for the installation of an energy project it implements by providing a savings guarantee.
- Provision of integrated energy services to customers (mainly large energy users, but also utilities), which may include implementing energy-efficiency projects (and also renewable energy projects) (Bertoldi et al., 2006).

As a result of government policy and the emergence of ESCOs, electrical power systems have been changing from conventional electricity generation towards green energy sources, which means that the structure for the management

of electricity that is currently in use will also need to change in order to fulfil the new requirements that this creates.

Furthermore, the distributed renewable generation will make an increasingly important contribution to electrical energy production in the future, and the integration of these highly variable, widely distributed resources therefore will need new approaches to power system operation and control. For example in the case of solar energy, the generation varies both hourly and seasonally, ranging between peak and zero generation, and as these are quite different patterns, the control systems will need to anticipate the fluctuations in both in order to satisfy consumers and compete in the energy market.

In other words, to deliver significant energy efficiency improvements, sophisticated operation and management tools are required which can allow an ESCO to process large amounts of real-time data and use this to make real-time decisions regarding the optimum way to operate an SSEZ. Such tools need to be able to model a wide range of generation sources, manipulate demand profiles, trade energy with European markets, and trade white and green certificates[1].

B.1.3.1 Small Scale Energy Zones (SSEZ)

An SSEZ is defined as a controllable low voltage distribution network (LVDN) that consists of a number of different small scale embedded generators (SSEGs), distributed energy storage units (ESUs) and customer demands (Cipcigan et al., 2009).

The UK electricity market is currently using late forecasting of generation output (one hour in advance) but it is expensive to collate this information from many SSEGs, on an ongoing basis. If these SSEGs are integrated and their out-

[1]Trade White Certificate(TWC) is a new policy instrument in the field of energy efficiency. Its basic idea is that energy suppliers or distributors must fulfil specific energy saving targets by implementing energy efficiency measures towards their clients within a specific time frame. Energy suppliers or distributors that save more energy than their targets can sell these surplus energy efficiency equivalents in the form of TWC to suppliers/distributors that cannot fulfil their targets. Trade Green Certificate (TGC) are about renewable energy. The targets of TGC schemes are to reduce oil dependency, meet Kyoto Protocol commitments (reduction of greenhouse gas (GHG) emissions), security of energy supply, and diversification of RE sources. The hierarchy of these targets can vary at some extent on national energy market characteristics and incumbent industrial structures (Oikonomou and Mundaca, 2008)

puts are combined within an SSEZ with controllable loads and storage devices, then as a group of generators it will be possible to trade larger amounts of electricity. This will enable the SSEGs to command a higher price in the electricity market that will in turn increase their value and stimulate their growth (Cipcigan et al., 2009).

An SSEZ is a concept that is similar to, and complementary to, that of the MicroGrid. However, while research on MicroGrids focuses on alternative future network designs, SSEZs exclusively consider the addition of SSEGs to existing LV networks (Trichakis et al., 2009). Both of the concepts needs to be coupled with an appropriate active control approach in order to make them successful.

An SSEZ must be able to overcome a number of associated LVDN constraints and meet a number of operational goals. Besides the technically driven goal of trying to operate an SSEZ within predefined statutory regulations and equipment ratings, an SSEZ should also have the ability to provide predictable and controllable demand and generation. In addition, an SSEZ could be used to provide ancillary services to distributed network operators (DNO's) which could include voltage support, power quality improvements or reductions in minutes lost by customers (Cipcigan et al., 2009).

From a management perspective, the key elements that an ESCO has to handle in its electrical network include:

1. Power flow control
2. Electricity generation
3. Metering /Billing
4. Energy Storage
5. Energy Supply and distribution
6. Demand side management[1].

[1]Demand side management (DSM) provides the way to plan, implement and monitor electric utility activities that are designed to influence customer usage of electricity in ways that will produce desired changes in the utility's load shape, i.e., changes in the time pattern and magnitude of a network load. Utility programs falling under the umbrella of DSM include: load management, new uses, strategic conservation, electrification, customer generation, and adjustments in market share (Gellings, 1985)

7. Active network configuration

8. Islanding[1]

In the above list, the energy market, weather forecast services and the national grid form a set of external bodies with which an ESCO needs to communicate. In this case study, the distribution network operator (DNO)[2] is considered as being part of the ESCO and the revenue that the ESCO is generating is not only coming from selling energy but also by providing ancillary services. An example of an SSEZ is shown in Figure B.1.

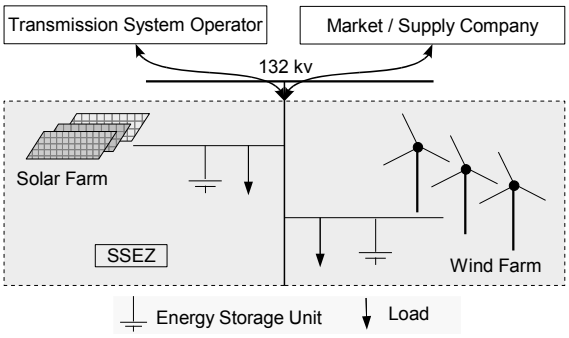


Figure B.1: Example of a Small Scale Energy Zone

The main task for the ESCO is to perform minute by minute management of its SSEZ within its network constraints and those constraints created by energy market conditions. The low voltage network constraints are determined by statutory regulations and also by equipment rating. Therefore, depending upon the network conditions, five operational goals that need to be met in order to run an SSEZ have been identified in (Trichakis et al., 2008). These goals are summarised below (but one thing to remember is that not all goals can be met at the same time).

[1]Islanding refers to the situation where distributed generator(s)(DG) continues to maintain the network voltage and frequency within regulatory limits to a location even after disconnection from the power utility (Smith et al., 2000).

[2]Electricity distribution networks carry electricity from the transmission systems and some generators that are connected to the distribution networks to industrial, commercial and domestic users. Domestic and most commercial consumers buy their electricity from suppliers who pay the DNOs for transporting their customers' electricity along their networks. Suppliers pass on these costs to consumers. Distribution costs account for about 20 per cent of electricity bills (Ofgem, 2012).

1. Zero power export: If local generation capacity is less than peak local demand, the goal could be to maintain a zero power export position to the distribution network.

2. Zero power import: If local generation capacity exceeds peak local demand, the SSEZ could aim for zero power import.

3. Zero power import and export (self-sufficient): If there is a close match between peak local demand and local generation capacity, the SSEZ could attempt to operate self-sufficiently, with no power exchange with the distribution network.

4. Constant power import: this involves operating with a fixed power demand, by having a constant level of power import from the distribution network.

5. Dispatchable power export: involves providing dispatchable power to the distribution network over a specified time period.

These goals are important for decision making and constrain the choices for an ESCO at any particular time. Decisions about SSEZ management be made at half-hourly intervals in accordance with UK and most European electricity market procedures. All of these operational goals are concerned with providing predictable and controllable demand/generation to the distributed network (Trichakis et al., 2008).

The goals represent different operational scenarios, and in order to maintain the SSEZ in the intended state, some form of control needs to be exercised. This in turn will need to draw upon information provided by the various internal and external elements at regular intervals, in order to produce a continually revised plan for SSEZ operation.

B.1.4 Design

In order to address the research question outlined in the previous section, it is necessary to design the case study protocol. The design guidelines used here are taken from the case study template available at `http://www.dur.ac.uk/ebse/templates.php`.

This is a single-case study. The 'case' (or unit of analysis) is the SSEZ control system. The selection of the case is on the basis of feasibility and access of resources within the School.

An operational model (use case) that can provide the basis for exercising the case within the case study through specific scenarios will be developed. The structure of the case study design is shown in Figure B.2.

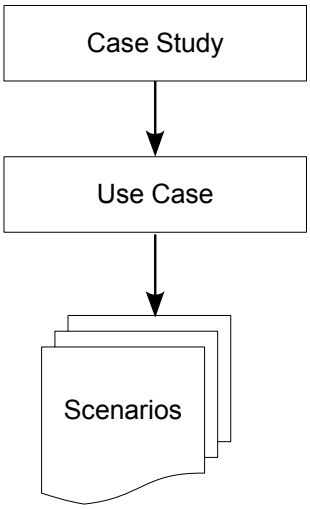


Figure B.2: Case Study Structure

The characteristics that the use case needs to meet are described below.

- *Adaptability:* A case needs to provide scope for employing adaptability to handle unpredictable environment or run time changes. This means at different points of time, different configurations of software elements will be required, where these may be anything from needing a different combination of services to be assembled, to just finding a replacement for a service provider. This means that the requirements may change in different ways, so that the axes along which adaptation occurs will differ from instance to instance.

  Apart from changes of requirements, the situation may arise whereby a service may not be available at some specific time, and so other service providers or sets of services will need to be used to ensure that the service

is delivered on time. The selection process might draw upon previous experience through mechanisms such as case based in order reasoning to handle such a situation.

- *Multiple Sources:* Where appropriate, it should be possible for a particular functionality (or service) to be delivered by different independent and remotely available sources. As a result the set of services selected at run-time may differ each time that the service is required.

- *Negotiation:* Access to, and selection of, resources (including service components and any data services) will be through a process of negotiation with service providers rather than control of local resources.

B.1.5 Data Collection

Data Requirement: The data required to develop the use case consists of the information that is considered the part of requirement specification in software development. The data includes SSEZ network configurations, SSEZ operational features, and network operational data. A sample dataset also needs to be constructed in order to check the behaviour of the SSEZ.

Sources of data collection: To collect data for the case study, triangulation will be used in order to provide the relevant evidence from multiple sources. The sources of information to be used are:

- *Interviews with domain experts:* Interviews will be conducted with domain export(s). The interviews will be recorded and notes will be taken to verify the information and so as not to skip anything.

- *Supporting documents:* The supporting material provided by the domain expert(s) in the form of reports, research papers, and links to relevant websites will be used to understand the domain and access data. For example, the Met Office website can be used to get information about a weather forecast service, and Elexon can provide data about demand, generation and energy market balancing, and settlement data for electricity distribution.

- *Informal discussions:* As the case study is taken from a domain other than computing, to understand the terminologies and processes, researchers from the same domain with different research focus will also be contacted for informal discussions and to collect data (for example, output of wind turbines, consumer demand data etc. with these being obtained from individual researchers who are working on these topics).

Record Keeping: To ensure that the information and data compiled through interviews and documents is correct, information will be processed in the form of narratives, tabulation and diagrams. They will be discussed in following interview sessions to get feedback and fill any gaps if exists. The audio recording will be used in order to check data consistency.

The information collected through different resources will be maintained through document version control. After conducting interviews and analysing supporting documents, the document will be updated. The document will either be sent to a domain expert or used in the next interview.

B.1.6 Analysis

Case study includes the creation of use case model and the design for SSEZ control system. The data that will be generated and the process that will be followed is shown in Figure B.3. The use case model will contain the details of SSEZ control system which includes the operational aspects and the physical configuration of the electrical network. This will help in generating different scenarios to realise the behaviour of the SSEZ control system. The details of the use case will be documented in the case study chapter. The case study data will further be used to develop a design model for SSEZ control system. This design model will make use of SOA model constructed in our earlier study on SOA discussed in Chapter 5. The design details will be discussed in the form of a chapter. A research paper will also be produced on the case study.

The use case and design model produced from the case study will be evaluated to answer the research question raised in this protocol. The evaluation process will help to identify the gaps and limitations of this case study.

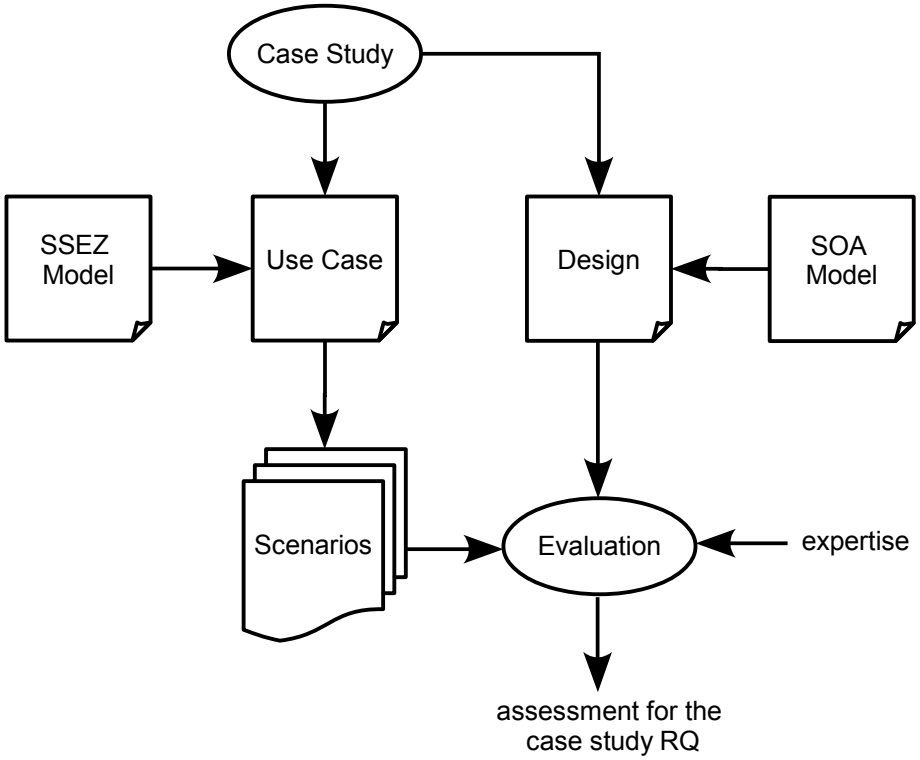


Figure B.3: Process of Case Study Analysis

B.2 Validity

Internal Validity

- Are the selected scenarios accurate enough (appropriate) to be used to evaluate the model?

External Validity

- Are the scenarios representative enough to allow generalisation of the results of using the model?

B.2.1 Study Limitations

Measurement and evaluation can be considered as issues. There are no prior framework or studies available to be used for comparison. Existing approaches have no set measurement criteria that can be used in this study. Energy domain is new for the author and the limited knowledge of domain may cause inconsistency in the data. There is possibility that design model might omit key aspects of domain.

B.2.2 Reporting

The target audience is composed of software engineers and SOA community. The data collected will be used to construct a use case that will explain the SSEZ control system specifications. Further the characteristics of the use case will be mapped with SOA features and a research paper will be published.

B.2.3 Schedule

The details about case study schedule are given in Table B.2.

Table B.2: Case Study Schedule

Task	Time
Formal Meetings with Energy engineers	one in two months
Data collection and document analysis	6 months
Data analysis and document generation	3 months

Appendix C

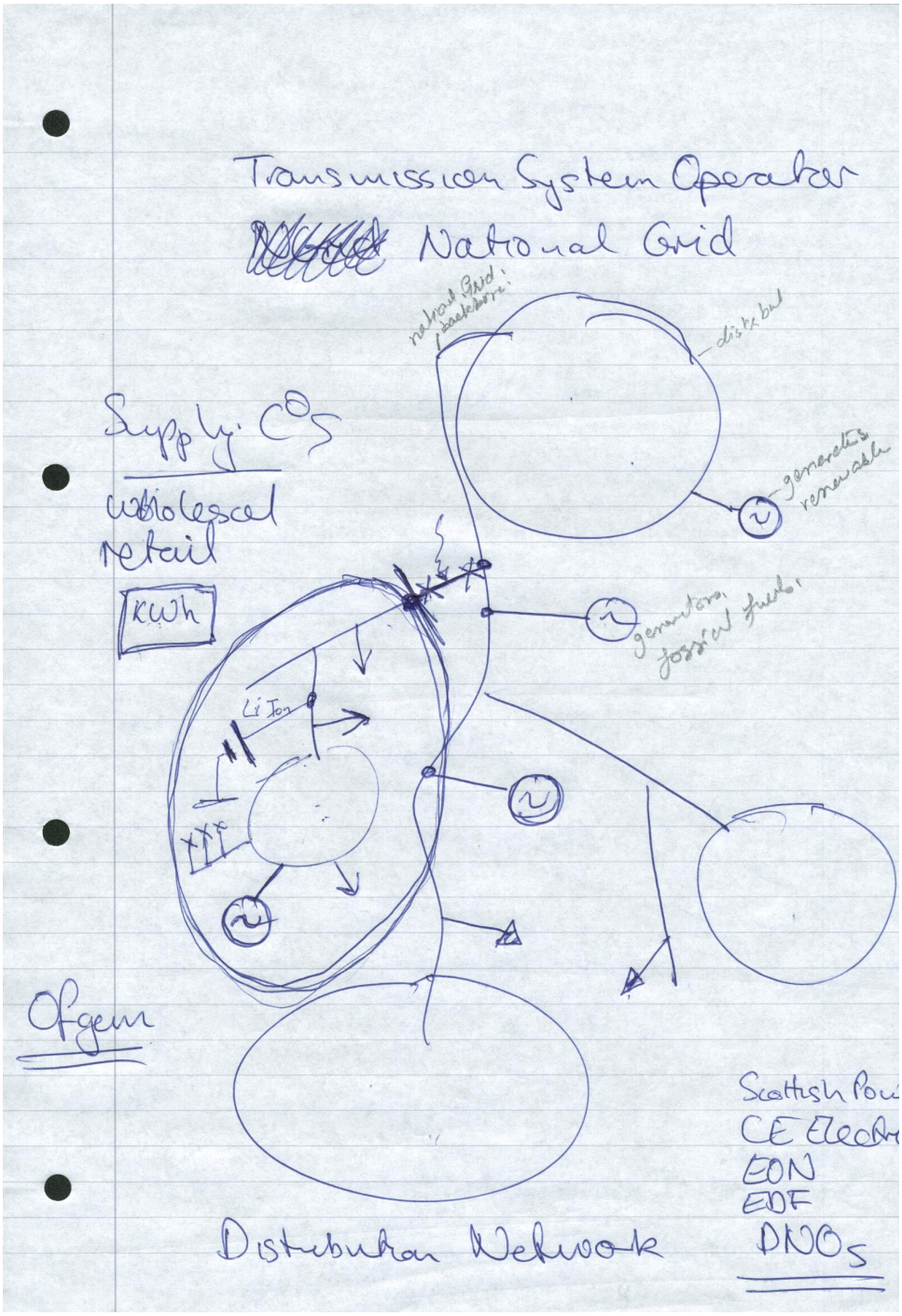


Figure C.1: Requirements figure 1

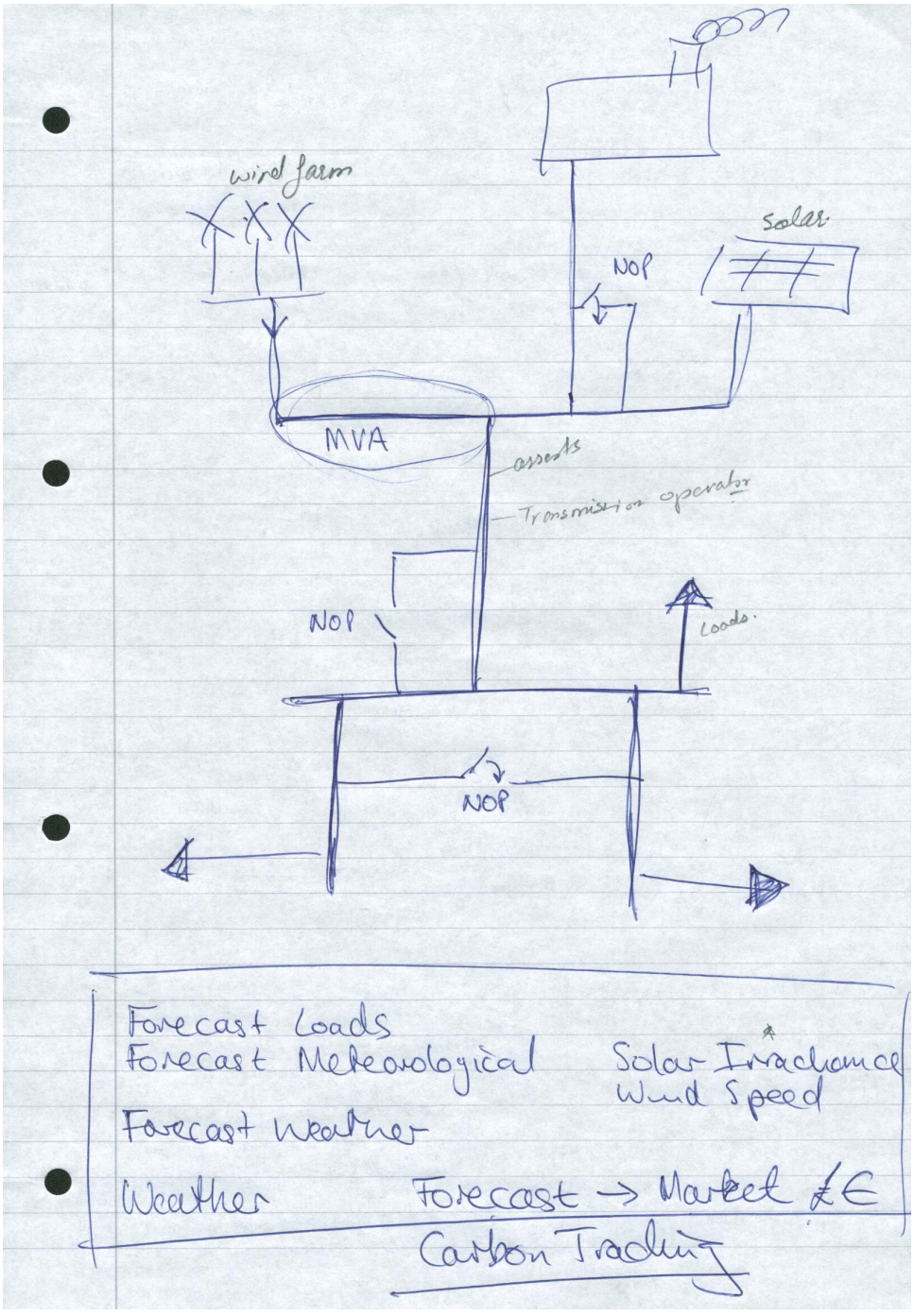


Figure C.2: Requirements figure 2

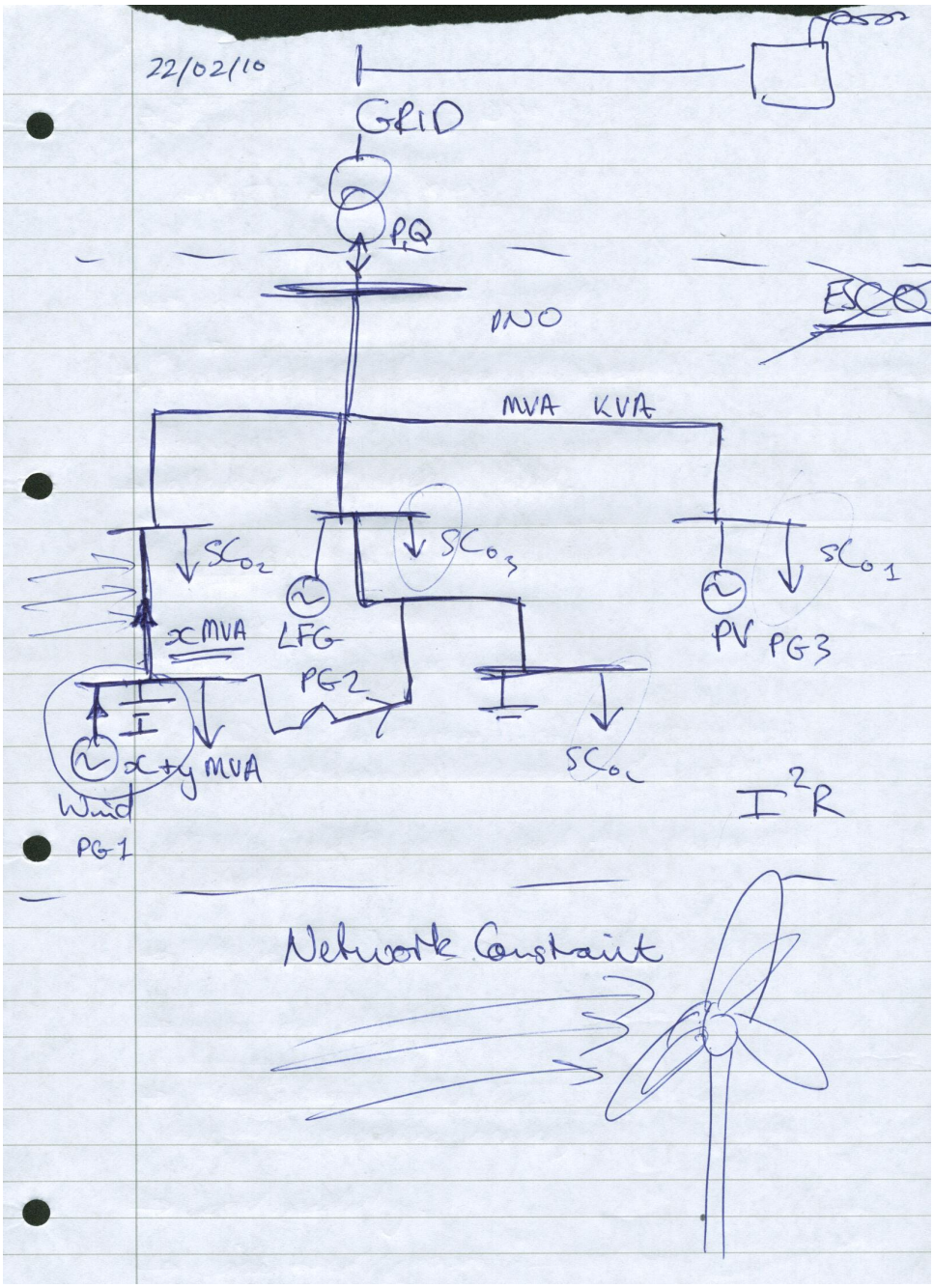


Figure C.3: Requirements figure 3

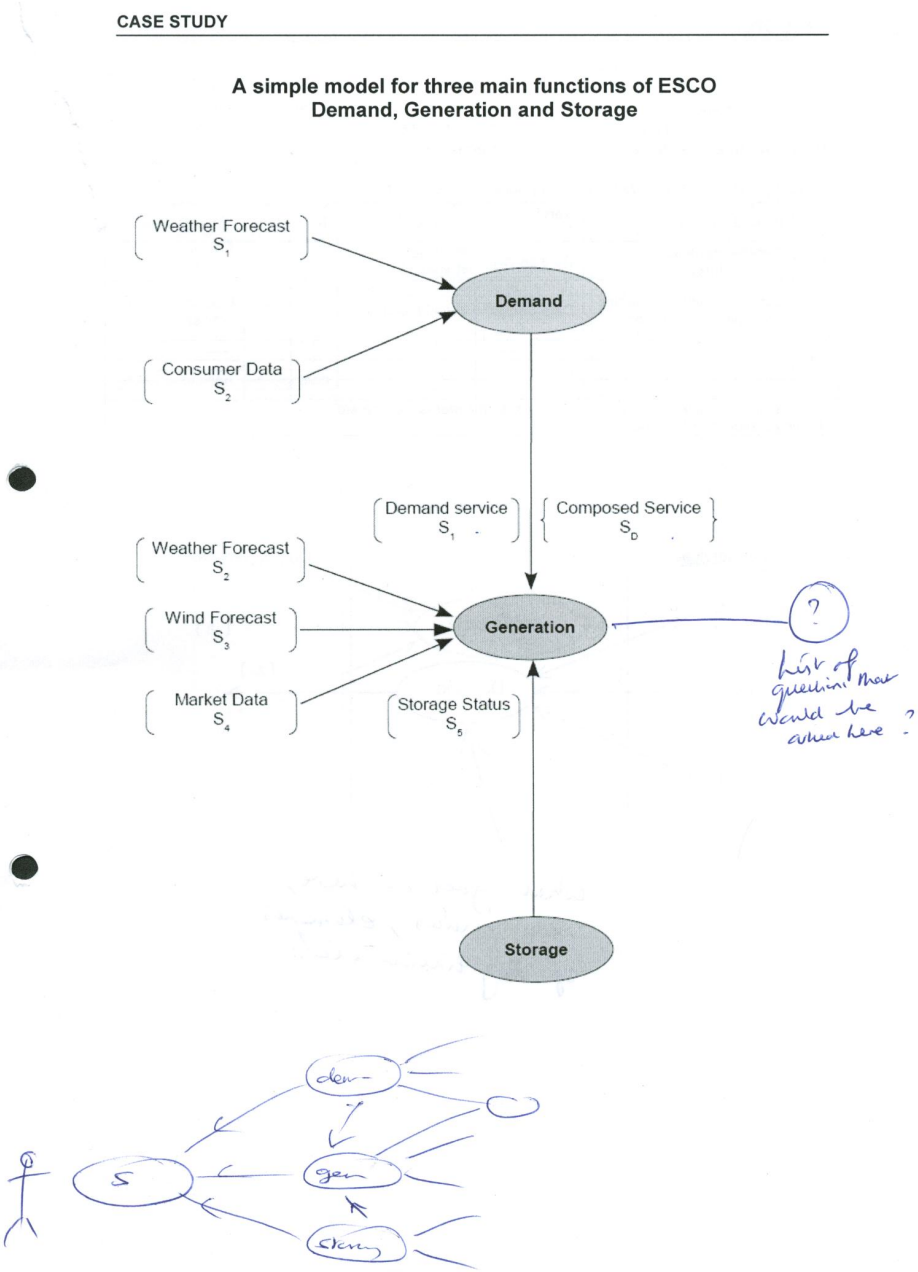


Figure C.4: Requirements figure 4

CASE STUDY

Service: Demand
Type: Composed Service
Description: actual demand and demand forecast

To compose demand service, following services are required:

Sr. No.	Input Services	Service Type	Source*	N	C	Other Details
1	Weather Forecast (temperature)	Data service	Website (External)			
2	Consumer data (residential / commercial / industrial)	Data Service	Smart metering			Consumer profiles
3						

** Source can be a company / website / other system/ internal service etc.*
N-> negotiation, C-> Contract

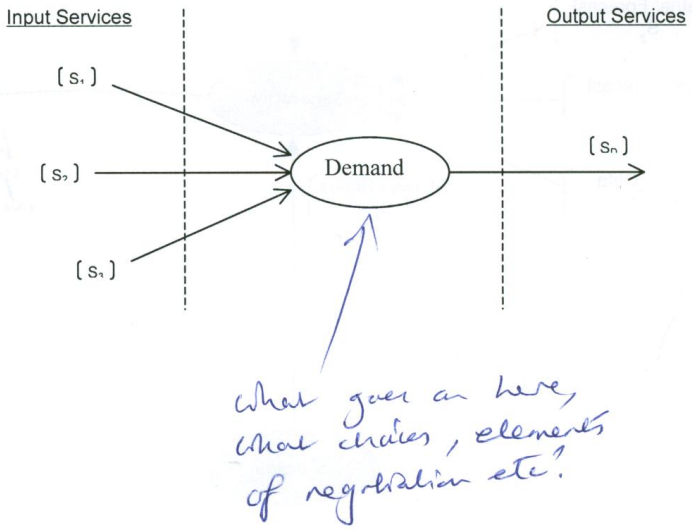


Figure C.5: Requirements figure 5

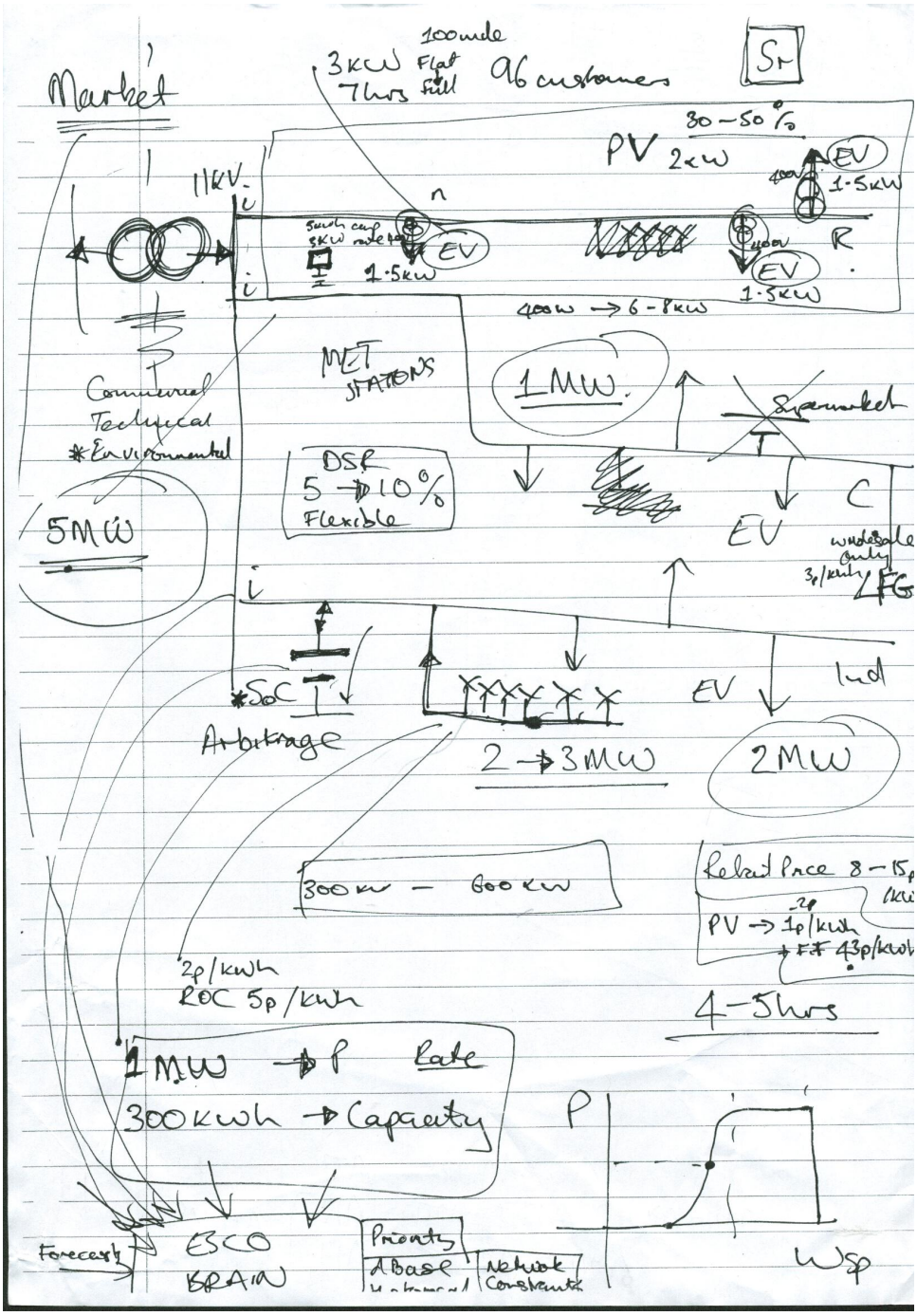


Figure C.6: Requirements figure 6

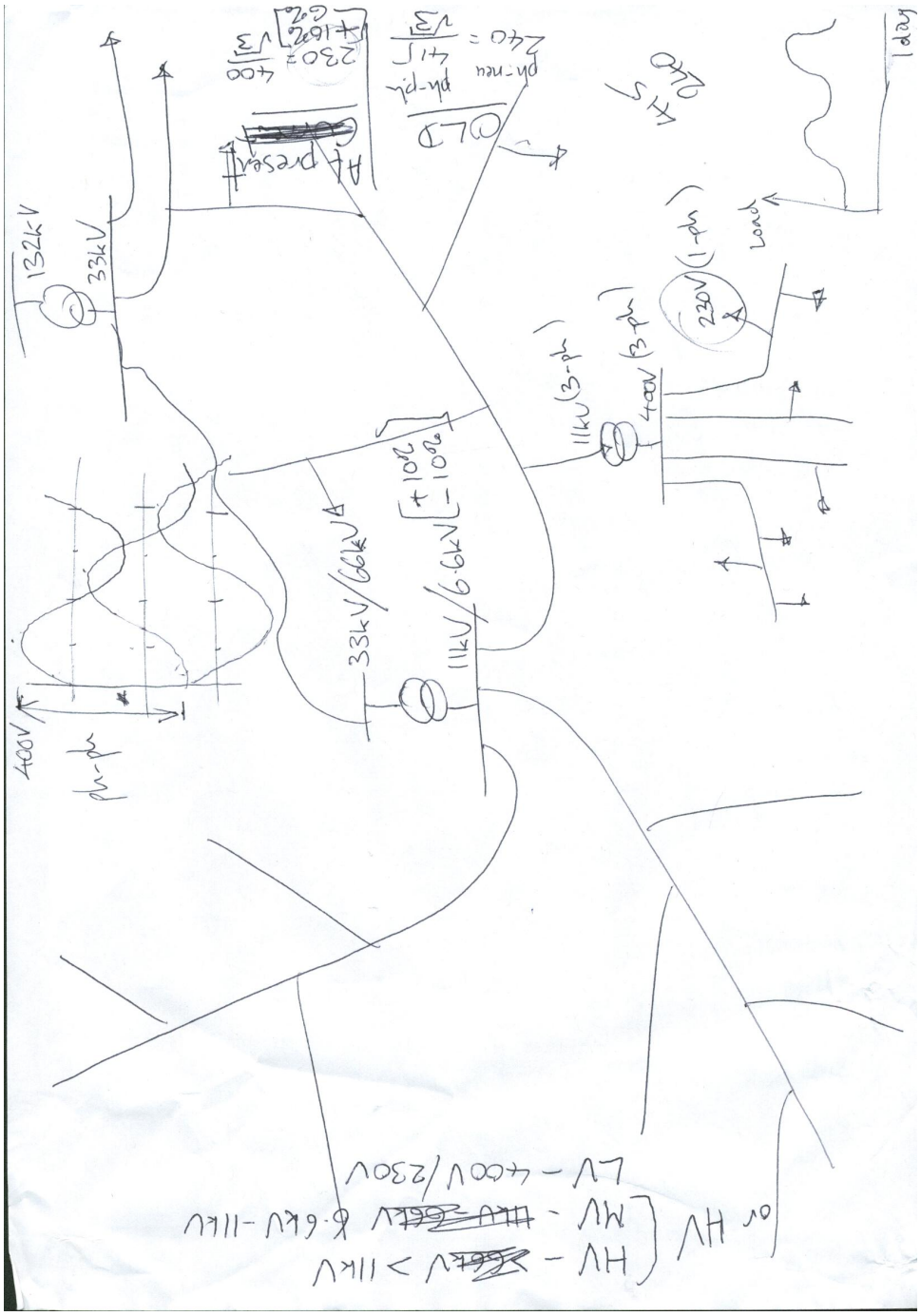


Figure C.7: Requirements figure 7

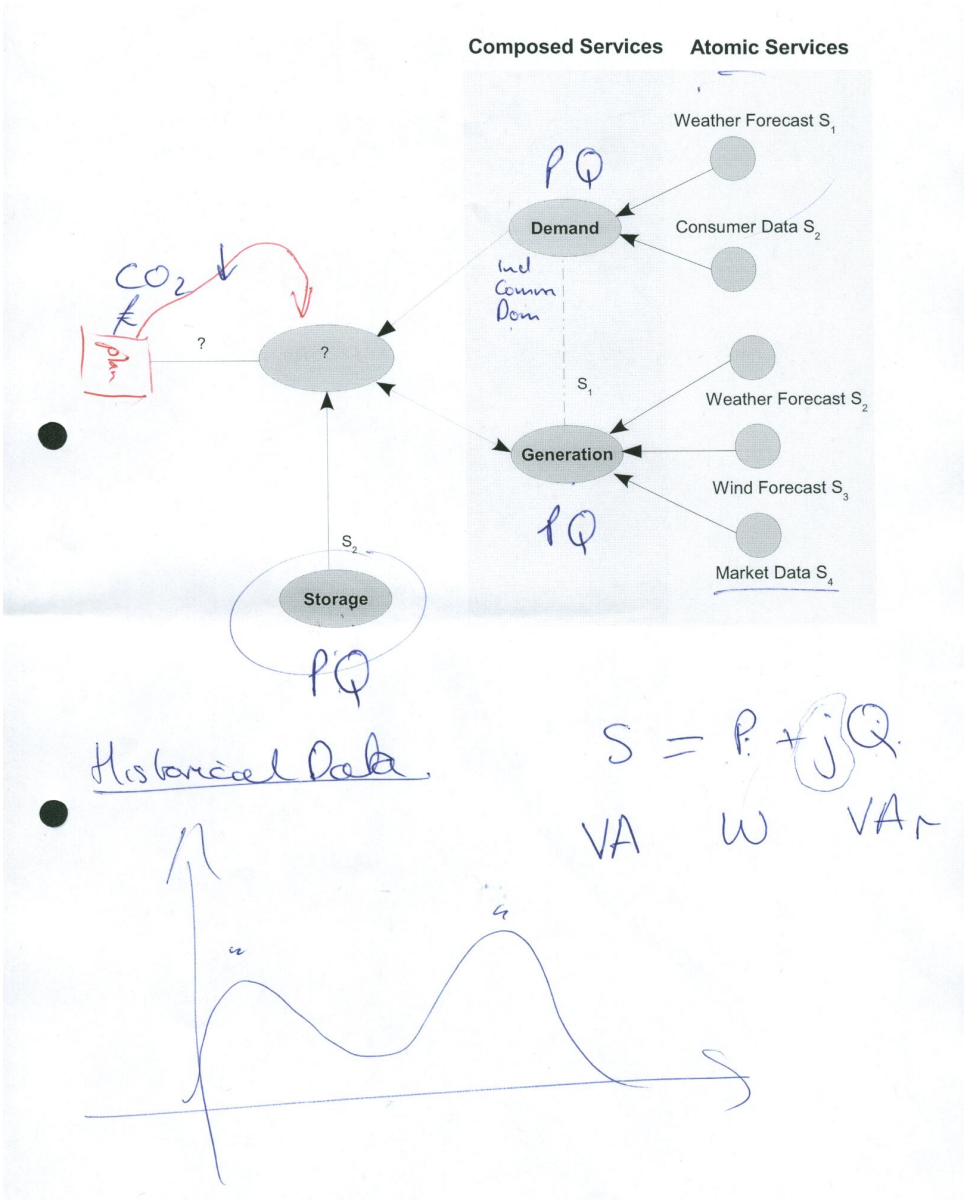
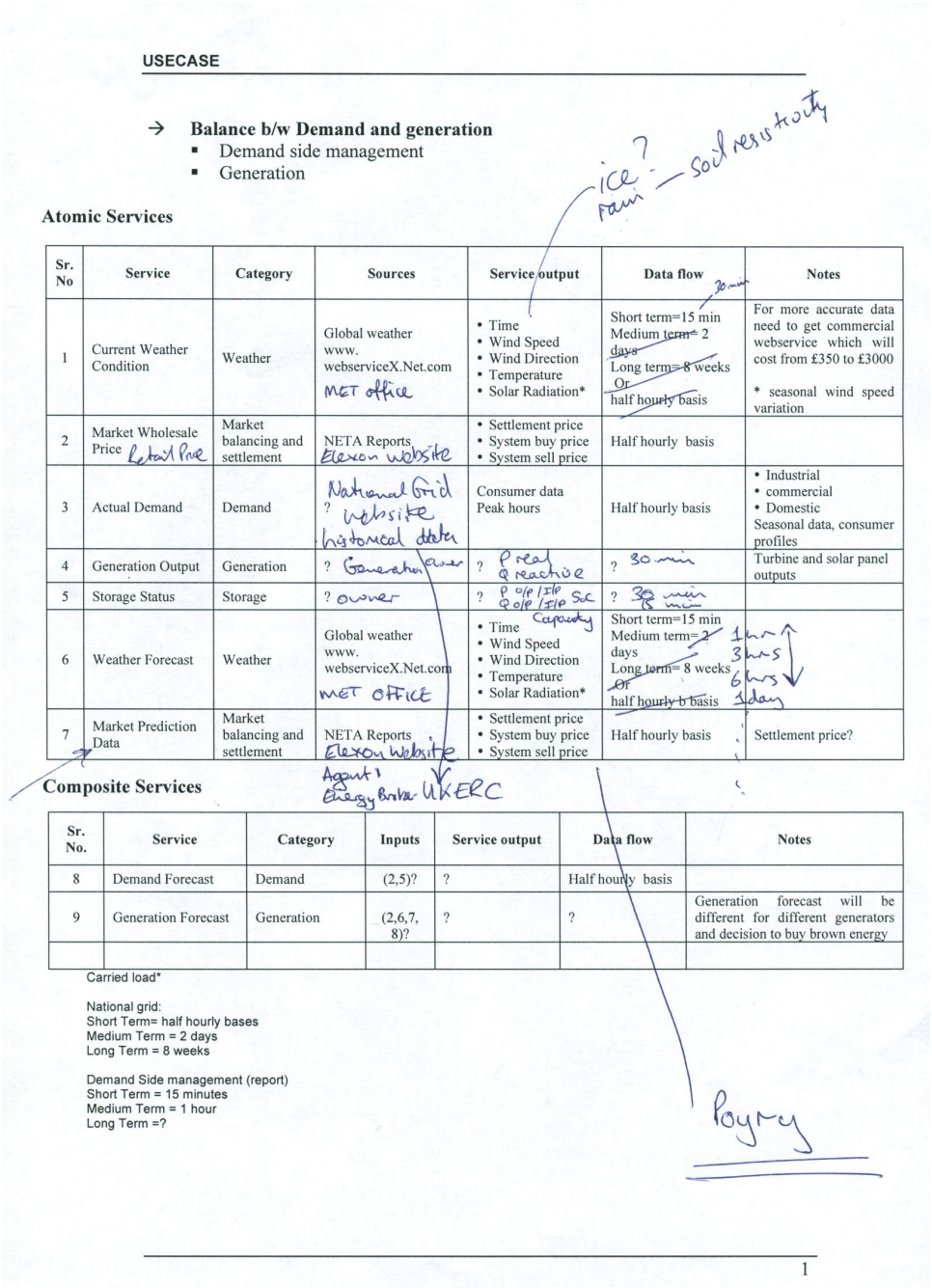


Figure C.8: Requirements figure 8



USECASE

→ **Balance b/w Demand and generation**
- Demand side management
- Generation

Atomic Services

Sr. No	Service	Category	Sources	Service output	Data flow	Notes
1	Current Weather Condition	Weather	Global weather www. webserviceX.Net.com MET office	• Time • Wind Speed • Wind Direction • Temperature • Solar Radiation*	Short term=15 min Medium term= 2 days Long term= 8 weeks Or half hourly basis	For more accurate data need to get commercial webservice which will cost from £350 to £3000 * seasonal wind speed variation
2	Market Wholesale Price Retail Price	Market balancing and settlement	NETA Reports Elexon website	• Settlement price • System buy price • System sell price	Half hourly basis	
3	Actual Demand	Demand	National Grid ? website historical data	Consumer data Peak hours	Half hourly basis	• Industrial • commercial • Domestic Seasonal data, consumer profiles
4	Generation Output	Generation	? Generation owner	? P real Q reactive	? 30 min	Turbine and solar panel outputs
5	Storage Status	Storage	? owner	? P o/p I/p Q o/p I/p Soc Capacity	? 30 min 15 min	
6	Weather Forecast	Weather	Global weather www. webserviceX.Net.com MET OFFICE	• Time • Wind Speed • Wind Direction • Temperature • Solar Radiation*	Short term=15 min Medium term= 2 days Long term= 8 weeks Or half hourly b basis	1hr 3hrs 6hrs 1day
7	Market Prediction Data	Market balancing and settlement	NETA Reports Elexon Website	• Settlement price • System buy price • System sell price	Half hourly basis	Settlement price?

Composite Services

Agent / Energy Broker UKERC

Sr. No.	Service	Category	Inputs	Service output	Data flow	Notes
8	Demand Forecast	Demand	(2,5)?	?	Half hourly basis	
9	Generation Forecast	Generation	(2,6,7, 8)?	?	?	Generation forecast will be different for different generators and decision to buy brown energy

Carried load*

National grid:
Short Term= half hourly bases
Medium Term = 2 days
Long Term = 8 weeks

Demand Side management (report)
Short Term = 15 minutes
Medium Term = 1 hour
Long Term =?

ice? rain — soil resistivity

Poyry

1

Figure C.9: Requirements figure 9

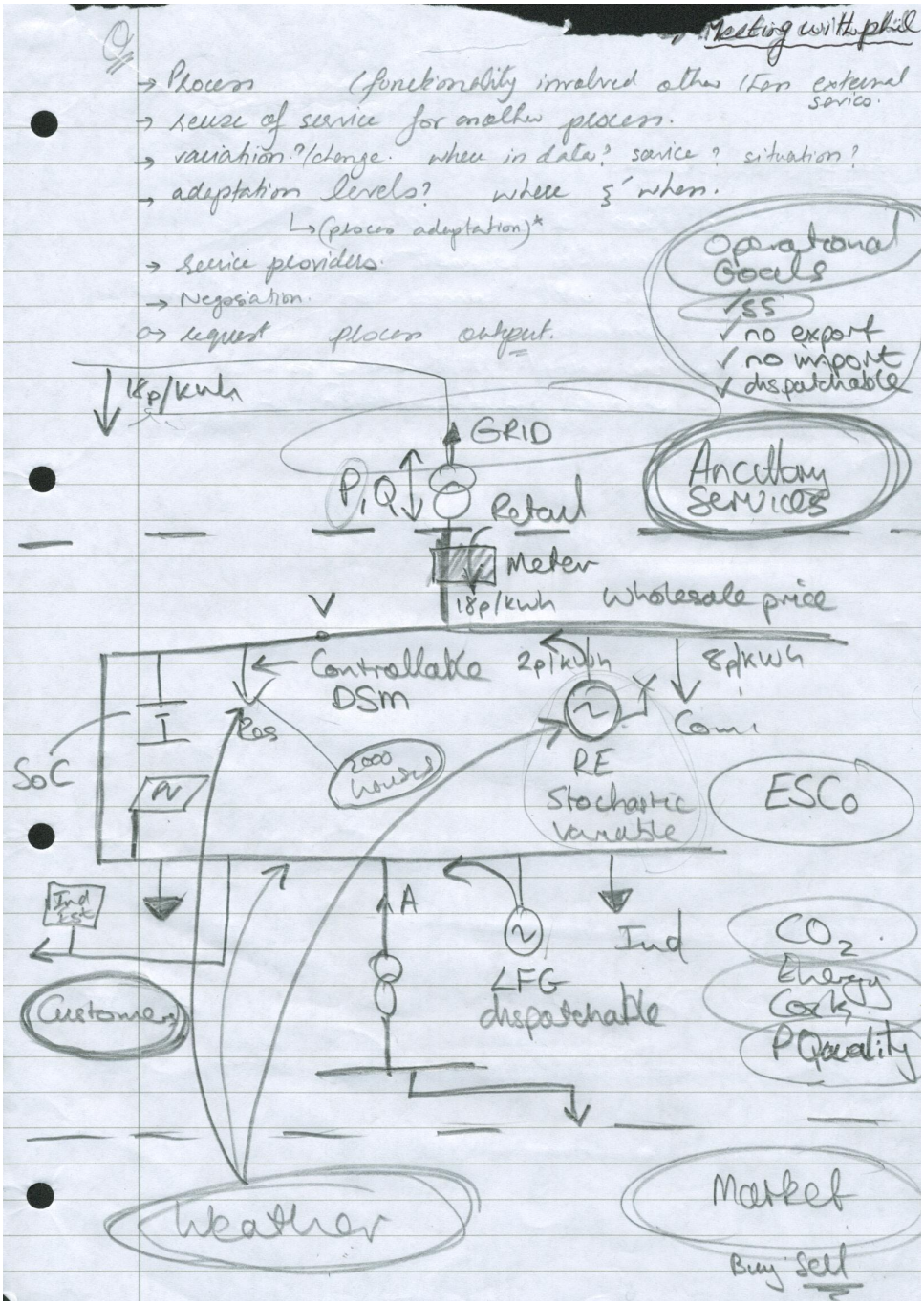


Figure C.10: Requirements figure 10

Figure C.11: Requirements figure 11

Figure C.12: Requirements figure 12

Figure C.13: Requirements figure 13

Appendix D

D.1 Use Case Related Details

Balancing Market: The balancing mechanism is one of the tools available to the National Grid to enable it to balance electricity supply and demand at close to real time levels. It is needed because electricity cannot be stored on any large scale and so must be produced at the time of demand. Where the National Grid predicts that there will be a discrepancy between the amount of electricity produced and that which will be needed during a certain time period, they may accept a bid or offer to either increase or decrease generation (or consumption). The balancing mechanism is used to balance supply and demand in each half hour trading period of every day[1].

Distributed Generation (DG) Power Output Control: The high power flows resulting from wind generation at high wind speeds can be accommodated because the wind speed also has a positive effect on component cooling mechanisms. The control system compares component real-time thermal ratings with network power flows and produces set points that are fed back to the DG for implementation.

Distribution network operator: The DNO is responsible for monitoring how much current is flowing down their cable in real time.

Network Assets: Overhead lines, electric cables, power transformers.

[1]http://www.elexon.co.uk, accessed August 2010

Power system component Ratings: Static, seasonal or real-time thermal ratings

Supply Company: The supply company should be able to estimate its customer demand accurately and able to buy carefully and not lose money in balancing market. A balance between electricity supply and demand is needed at all times to ensure a stable and reliable market. This can be done in two ways, either through Supply-side Management by adding supply when demand is high, or through Demand-Side Management (DSM), which involves curtailing the system demand when supply availability is less. For short term measures, supply-side management is not effective as it takes a long time for generating units to start up (if these are available) and so these cannot meet the rising demand immediately, whereas demand side management can be implemented immediately and in more economic ways in order to keep the balance.

System Sell Price and System Buy Price: The System Sell Price (SSP) and the System Buy Price (SBP) are the cash-out prices or imbalance prices that are used to settle the difference between contracted generation or consumption and the amount that was actually generated or consumed in each half hour trading period. SSP is paid to BSC Trading Parties who have a net surplus of imbalance energy, and SBP is paid by BSC Trading Parties who have a net deficit of imbalance energy. These prices are designed to reflect either the prices associated with the balancing mechanism offers and bids selected by National Grid to balance the energy flows in the Transmission System, or to reflect the prices associated with the sale and purchase of (short-term) energy ahead of Gate Closure (set at one hour before each half our trading period) in the forwards and spot markets.

Retail price: The electricity retail price for different sources is as follows:

- Domestic use: 8 to 15 p/kWh
- PV output:1p/kWh + feed-in-tariff 43p/kWh = 44p/kWh
- Wind farm price: 2p/kWh + Renewable obligation 4p/kWh = 6p/kWh

- Battery usage charges are negotiated in advance anddepends on current price.
- Landfill gas get wholesale price 3p/kWh only.

D.2 Network Details

The Table D.1 provides information about the change in the pitch angle according to wind speed.

Table D.1: Pitch angle and wind speed (Zhang et al., 2008)

Wind Speed (m/s)	Pitch Angle (degree)
4 - 13	0
13.5 - 15	0 - 5
15 - 16	5 - 8
16 - 17	8 - 10
17 - 18	10 - 11
18 - 19	11 - 13.5
19 - 20	13.5 - 14.5
20 - 21	14.5 - 16
21 - 22	16 - 18
21 - 23	18 - 19
23 - 24	19 - 20
24 - 25	20 - 23

Mostly 2v and 24v batteries are available. The limits required to charge and discharge batteries are applied on the volatage state and the percentage of SOC. The table D.3 provides the relationship between voltage and the SOC percentage.

The table D.2 provides the relationship between voltage and the SOC percentage along with the possible conditions that can be applied at different level.

D.3 Possible Network Extension

- Adding a local dirty generator means that a landfill gas generator can be added in the network configurations, requires the model to consider envi-

Table D.2: Voltage Condition and possible current/future states

Condition	Current State	Future State
$V \geq 12.63$	charging	stop charging
$V \geq 12.63$	not charging	not charging
$V \geq 12.54$	charging	not charging
$V \geq 12.45$	not charging	not charging
$V \geq 12.39$	charging	charging
$V \geq 12.39$	discharging	discharging
$V \geq 12.39$	not charging	charging
$V \geq 12.27$	discharging	discharging
$V \geq 12.27$	charging	charging
$V \geq 12.27$	not charging	charging
$V \leq 12.18$	charging	charging
$V \leq 12.18$	discharging	stop discharge
$V \leq 11.97$	charging	charging
$V \leq 11.97$	discharging	stop discharge

Table D.3: Relationship between voltage and SOC

Voltage (V)	SOC (%)
12.63	100
12.54	90
12.45	80
12.39	75
12.27	60
12.18	50
11.97	25
11.76	completely discharged

ronmental factors by calculating the trade-off between using clean and dirty generators when the day is neither windy nor sunny.

- Batteries can be added to the houses for domestic customers to store energy and use at period of high demand.
- Electrical vehicles (EV) can be included, requiring a slow charge of 3 kW. The demand varies on customer use. This takes 7 hours from flat car battery to full, with 100 mile range. EVs may be considered as active loads, increasing the demand on the network during charging, and as generators when operating in regeneration mode.
- The network constraints can be considered in the extended model that include voltage rise and voltage unbalance limits determined by statutory regulations and operating distribution network circuits above their thermal limits and reverse power flow through distribution transformers by thermal rating.
- Wind farms can be owned by any other third party and the ESCO then has to negotiate with them to import power in its network.

Appendix E

E.1 Review Protocol

E.1.1 Change Record

Table E.1: Change Record

Version	Change
1.0	initial draft
1.1	new sections with added details
1.3	details in sections, new section validity

E.1.2 Background

Review is considered an evaluation technique that is employed in order to evaluate the quality of the work product (such as requirement specification, design document or source code) (Garousi, 2010). In a review, issues are raised by the reviewers about the work product and an issue list is provided to the producers so that they can resolve them. A review is conducted by the *people* who are not involved in producing the work product in order to avoid possible bias. Fixing the defects found at the early stages of software development involve less cost than for those that are identified at a later stage. The statistics collected from organizations have shown that properly conducted review can eliminate 60% to 90% of existing defects from a workproduct (Fagan, 1976). Therefore the use of reviews is suggested for each stage of software development (Ackerman et al., 1989).

Approaches to conducting a review differ from each other in certain aspects. Weinberg and Freedman (1984) describe the main review approaches as being:

- Inspections: Focus on a set of questions and a checklist of issues /questions prepared for the review. The list of points in the checklist determines the flow of the review.
- Walkthrough: The author presents the work to the reviewers step by step. The review is organised around the structure of the material being presented.
- Plain review: This depends on the flow of the meeting as it progresses.
- Round-robin: A cyclic approach is used among the participants and each participant in turn gets to raise an issue.

There is no strict line between different review forms. They are adapted according to the demand of the work product under review. The earliest forms of review were focused upon finding defects in code. For this reason, inspections are a popular way to evaluate source code. Also they are formal and focus narrowly on the problem by employing checklists.

From the review forms described by Weinberg and Freedman (1984), a walkthrough is the one that is considered less formal than an inspection (Fagan, 1976) and so could also be used for analysis purposes with a large audience. In a walkthrough, the preparation is done by the presenter who is very familiar with the work product or the author. This reduces the load upon the participants, and a large amount of material can be presented quickly to a large number of participants. However, a walkthrough can generate a large number of diverse views about the presented material (Weinberg and Freedman, 1984). For this reason, an element of control is usually introduced by employing a person to act in the role of a *moderator* in review sessions.

To get the maximum benefit from a review, the objectives of the review need to be made explicit through a systematic review process. The process must ensure that the design is covered completely and in detail by the review team (Parnas and Weiss, 1985). In the guidelines of IEEE (IEEESTD, 2008), for a review to be

considered a systematic walk-through, a team of at least two members (including the author) should be assembled. Also, the resulting review report is an important part of the review that serves as a formal commitment by technically competent and unbiased people that a piece of work is complete, correct and dependable (Weinberg and Freedman, 1984).

In the literature, the use of checklists is suggested for different forms of review and for the different phases of software development. A summary by Brykczynski (1999) provides a categorization of 117 checklists from 24 sources. This study suggests that tailored checklists should be developed to meet the purpose of a review and to help the reviewers to identify the defects. Also, checklists need not be too lengthy and should be limited to one page in length.

We have employed a mapping study to find evidence from the literature about the attributes that constitute an SOA. Further we have developed a case study from energy engineering about the management of an SSEZ control system. The case study is used to construct an SOA design model through existing notations. In order to evaluate the requirements gathered for the SSEZ control system, and to assess the appropriateness of the design model, along with the suitability of notations used, we propose to conduct a walkthrough. As described by (Budgen, 2003), a walkthrough is a useful techniques for assessing the structural and behavioural aspects of the design. A well planned review utilises the skills of the review team who have domain and technical knowledge to estimate the future situation from the available design information. The overarching research question for this review therefore is:

"*Are the design and notations used appropriate for the construction of an SOA model for the specified SSEZ control system?*"

Further to this, the review will assess the following issues:

(a) Are the domain problem characteristics defined correctly?

(b) Are the main operations of the system covered?

(c) Are the assumptions valid?

(d) Are the functional components and processes sufficient to realise overall system functionality?

(e) Is the system behaviour represented correctly?

E.1.3 Design

In order to address the research questions outlined in the previous section, it is necessary to design the review so as to collect necessary data. The data collected through the review will be used to analyse the expert views. The design principles used here are taken from the case study template available at http://www.dur.ac.uk/ebse/templates.php. As the template was meant to be used for case study protocol, we have tailored this to fulfil the review requirements (as discussed in (IEEESTD, 2008; Ackerman et al., 1989; Weinberg and Freedman, 1984)). This was done with the consultation of the supervisor. We set our review plan as follows:

Form of review: The purpose of the review is to collect expert views about the design and to collect data to explain the research questions. The review is on the topic of design which has different needs to reviews conducted for code. Hence, a walkthrough approach has been selected for this purpose. We have prepared a semi-structured questionnaire (Appendix F) to collect data. This will make the session more objective and help to make the participants concentrate on a particular aspects of the review. The questionnaire was prepared by adapting the guidelines given in (Budgen, 2012), whenever they apply.

Roles: The roles involved in the review are those of moderator, author (designer) and reviewers. IEEESTD (2008) guidelines suggest that the number of people involved should be between 2 to 7. In this review, the team will consist of five persons. In the guidelines there is a role for a recorder to take notes for the review. This role is replaced with the environment we have selected to record the review.

Population: We need to find a set of reviewers who have knowledge of the domain and also have some software design experience. Also the role of moderator will need to be assigned.

Selection of team: We have a number of possible sources of experts. We can invite energy engineers including teachers and researchers within the school. For design experts, we can contact commercial software developers, and teachers who have SOA experience in the market and in the universities respectively.

Review Length: In the guidelines available for review, it is recommended that the review time should not exceed two hours. Longer than this will make the review less effective. Therefore we have allocated two hours for a review session.

Review Structure: The review session is divided into two stages.

- In the first part of the review, scenarios will be used to generate requirements issues.
- In the second part, the design model will be exercised by the author.

Data Requirements: We aim to collect data about the issues raised by the reviewers about the requirements specification of domain problem, the design model, and the notations used.

Record Keeping: The session will be recorded to help with analysis. For recording, we will make use of the voice and video environment available in the school. This will provide fuller data about the discussion occurring during the review and will capture the whiteboard activities.

E.1.4 Data Preparation and Collection

- The purpose of walkthrough is to identify gaps and issues associated with the proposed design. For this reason, the issues that need to be identified fall in three categories: requirements, SOA design, and notations. Further categories could be defined or reviewers could be asked to provide these.

- For data collection, the questionnaire will provide a guideline. The questions cover requirements, assumptions, and design. The use case document (appendix G) contains representative scenarios from the problem domain.

This will provide the basis for generating requirement issues. The design activity that will be carried out using a whiteboard, will help with identifying issues related to requirements, SOA and notations.

- For data collection, we have to ensure that the issues covered in our questionnaire are addressed. We also need to ensure that the discussion by reviewers is not going beyond the allocated time. It is also important to keep the flow of the review and bring the reviewers back to the main topic if they are drifting away.

E.1.5 Analysis

The data collected from review will be used in the discussion chapter of the thesis. The data will provide the issues raised by reviewers. These issues will be grouped according to the categories we have defined earlier. Further, they will be analysed to address the reserach questions. The gaps identified in the review will also address the completeness and correctness of the proposed solution. A walkthrough is an informal apporach for identifying issues in the workproduct. For this reason we expect that the review session may raise some issues for that we have not provided any specific category.

E.1.6 Threats to Validity

- The experiences of reviewers can cause conflict during review session.
- A conflict can arise due to the difference of both verbal imagery and mental imagery between author and reviewers.
- The questionnaire, and use case document may create bias, but have been reviewed by the supervisor.
- A possible threat about data consistency can arise during participant observation (Seaman, 1999). In reviews a recorder is usually meant to take notes about the review session. This could effect the accuracy of the data and can limit the understanding of the notes to him. This threat is handled

through the use of a video recording of the review. This resolves the chance of any possible conflict in data collection.

E.1.7 Study Limitations

The reviewers will be from application disciplines rather than software design experts. Also the expertise of the author may effect the review.

E.1.8 Reporting

The target audience is composed of software engineers (largely designers), SOA community and notation designers. We will describe the data collected in the evaluation chapter. A "lesson learned" report will be prepared about the experience of conducting a review for design.

E.1.9 Schedule

The details about review schedule are given in TableE.2.

Table E.2: Review Schedule

Task	Time
Design review questions and supporting documents	4 weeks
Review and collect data	1 day
Review Schedule:	
Introduction	5 minutes
Present the design	15 minutes
Requirements questions	30 minutes
Assumptions questions	10 minutes
Design questions	60 minutes
Analyse data & generate document	4 weeks

Appendix F

F.1 Questionnaire

F.1.1 Version Control

Table F.1: Questionnaire version control

Version	Details
1.0	initial draft
1.1	changes in different sections
1.2	change in questions

The review questions are divided into three categories: requirements, assumptions and design.

F.1.2 Requirements:

- Are main features of the SSEZ control system covered?
- Does the description of the application provided explain SSEZ operations accurately?
- Is the domain analysis complete, consistent, and accurate?
- Do the scenarios represent the domain problem correctly?

F.1.3 Assumptions:

- Are the assumptions valid?
- Are the assumptions sufficient for the scope of the exercise?

F.1.4 Design:

- Does the design represent the main functions of the SSEZ control system?
- Are all functions described in sufficient detail?
- Are the main functional components and processes as defined sufficient to realise overall system functionality?
- Is the system behaviour presented in design model correctly?
- Does the design provide for necessary system behaviour in different situations?
- Is the level of decomposition sufficient to identify all activities?
- Is the design consistent with the requirements?
- Are the interfaces specified to a sufficient level of detail?

Appendix G

G.1 Use Case Document

G.1.1 Version Control

Table G.1: Use Case version control

Version	Details
1.0	initial draft
1.1	changes in different sections

G.1.2 Use case

Operational management of an SSEZ is aimed at organising its electrical network to make maximum use of its renewable resources (e.g. wind turbines). The energy control system required for an SSEZ should be able to check the energy balance in the zone. In the case of surplus energy it should be able to sell energy to the grid by getting a price from energy market; while in the case of an energy deficit, the system should try to buy energy from the market at a low price where possible. It also needs to communicate with external data sources such as a weather service to get current and forecast weather data. The priorities, for operational management are:

- Make maximum use of energy sources, and trying to avoid using brown energy where possible.

- Satisfying customer demand.
- Keeping a balance between demand and generation.
- Making money by selling surplus energy.
- Stopping any wind turbines will be the last option in extreme conditions.

The information sources involved are:

- Generation output level
- Demand level
- State of storage unit
- External weather service
- External energy market service

Assumptions:

- The network is working in normal operating conditions and no faults present.
- The data coming from wind turbines is in the form of aggregate (not individual) turbine data.
- The demand, storage unit and wind farm data is accessed through interfaces in the form of services.
- The storage unit will remain 50% charged all the time.
- The system parameters need to be revised at half hour intervals in accordance with UK and most European electricity market procedures.

Scenario 1: Consider a windy day, the demand for electricity is low and generation is more than required by consumers in the zone.

Scenario 2: Consider the situation when there is a football match in the evening and energy demand is high. The weather is warm and wind turbines are running on average capacity. There is an energy deficit in the zone.

Scenario 3: Consider the case when energy market sell price is high and there is energy need in the zone.

Scenario 4: Market service is not available and there is energy deficit in the zone. Consider the same if there is surplus energy in the zone.

Scenario 5: The storage is half empty; a few turbines are out of order so the generation output level is low in the zone. Due to cold weather the demand is high and energy buy price in the market is also high.

Scenario 6: The wind is good and demand is low in the zone. The storage is fully charged. However, the link is down and SSEZ could not get the market price.

Appendix H

H.1 Notations

H.1.1 Activity Diagram

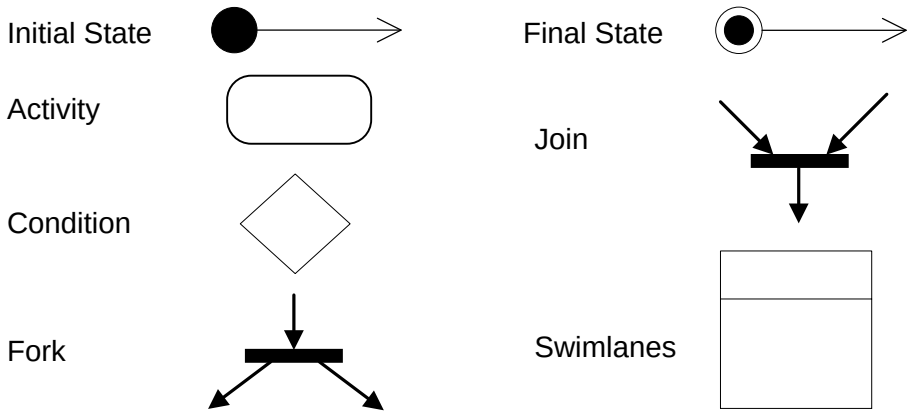


Figure H.1: Activity Diagram Notations

H.1.2 Class Diagram

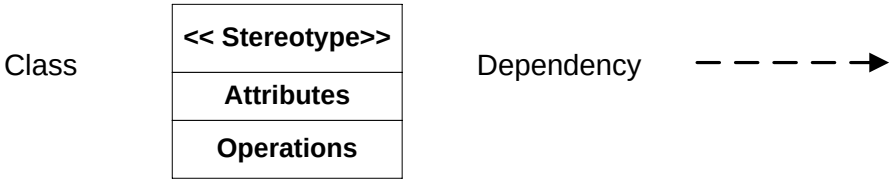


Figure H.2: Class Diagram Notations

H.1.3 Component Diagram

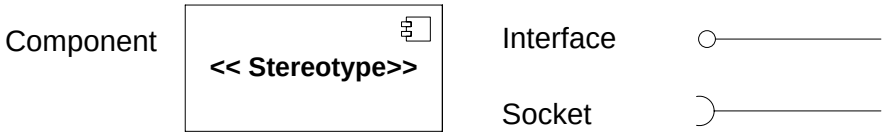


Figure H.3: Component Diagram Notations

H.1.4 Data Flow Diagram

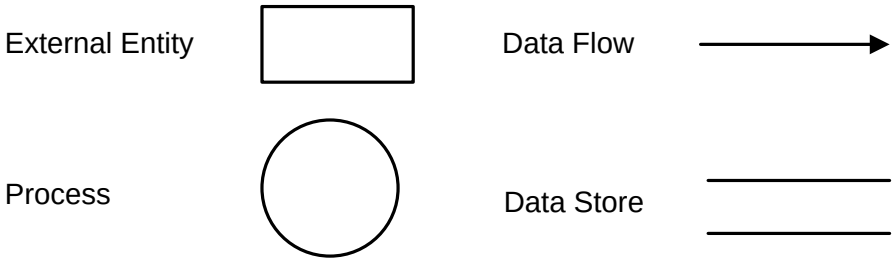


Figure H.4: Data Flow Diagram Notations

H.1.5 Sequence Diagram

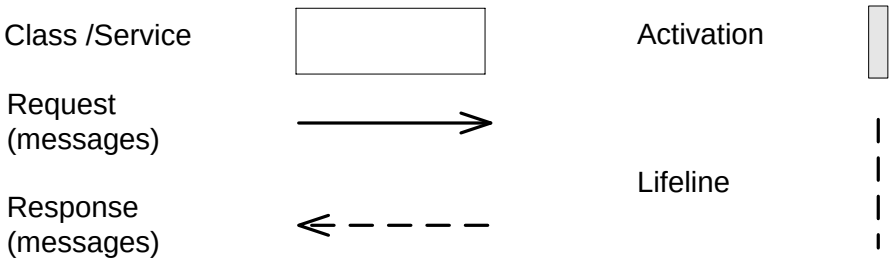


Figure H.5: Sequence Diagram Notations

Appendix I

I.1 Interview Questionnaire

I.1.1 Part 1: Reviewing the walkthrough process itself

- Do you think that the walkthrough was effective in terms of meeting its aims?
- What elements do you think were lacking in the organization of the walkthrough, both in terms of the process and of the material provided?
- What things could be done to improve the walkthough process?

I.1.2 Part 2: Presentation of the design

- How well were you able to understand the design of the software system?
- What could be done to improve the design presentation, both in terms of how it was organized and the forms used?
- Are there better ways or forms that we could use to describe and present the design?

Appendix J

J.1 Summary of Responses from First Interview Session

Table J.1: Summary of Interview responses Table 1

Questions	Participant 1	Participant 2	Participant 3
(a)	Yes, Useful in terms of learning like knowing about SOA.	Great idea, effective in terms of getting feedback on your work. Filling gaps if any in understanding the domains. Computer science way of looking things and looking from engineering view point was different.	Interesting, we learned from this exercise. Walkthrough is effective in terms of identifying issues that were overlooked.
(b)	The purpose and context need to make clear. Keywords used on the board can be difficult to understand and need frequent consultation from documents.	Need to mention where to focus in the document. The moderator was not involved in discussion so was not sure were to prompt us.	Not clear what walkthrough was about.

Table J.2: Summary of Interview responses Table 2

Questions	Participant 1	Participant 2	Participant 3
(c)	PowerPoint presentation would be effective to track sessions.	Power point would be quicker. Whiteboard was free form in the beginning and then it became focused.	Process was well and moderator did a good job. Keep the same team. Administration, timings and process is fine.
(d)	Avoid notes style. Keywords need to backed by presentation.	Presentation in terms of explaining both application domains was useful.	Was not very clear in the beginning. Need to mention particular section in the design document to focus on. Acronyms are problem.
(e)		Diagrams with further details will be helpful. Through PowerPoint presentation it will be easy to provide overall picture.	Simplicity is Important. It is difficult to please both domains. Next session don't need background information on services and windfarms.
(f)		The diagrams need more expressiveness to other domains. Computer science diagram was difficult to grasp for energy engineer and the same way for computer scientist it was difficult to understand the electrical network diagram. Glossary would be helpful.	PowerPoint will not create much difference. Diagrams are fine need more description in the document. Reviewers have learnt from this.

References

Aalst, W. V. D. (2003), 'Dont go with the flow: Web services composition standards exposed', *IEEE Intelligent Systems* **18**(1), 72–76. 140

Ackerman, A. F., Buchwald, L. S. and Lewski, F. H. (1989), 'Software inspections: an effective verification process', *IEEE Software* **6**(3), 31–36. 223, 226

Adelson, B. and Soloway, E. (1985), 'The role of domain expenence in software design', *IEEE Transactions on Software Engineering* **SE-11**(11), 1351–1360. 7, 29, 30

Agarwal, V., Chafle, G., Mittal, S. and Srivastava, B. (2008), Understanding approaches for web service composition and execution, *in* 'Proceedings of the 1st Bangalore Annual Compute Conference', ACM, pp. 1:1–1:8. 23

Ali, N., Nellipaiappan, R., Chandran, R. and Babar, M. A. (2010), Model driven support for the service oriented architecture modeling language, *in* 'Proceedings of the 2nd International Workshop on Principles of Engineering Service-Oriented Systems', ACM, pp. 8–14. 136

Allison, D. S., Yamany, H. F. E. and Capretz, M. A. M. (2009), Metamodel for privacy policies within SOA, *in* 'Proceedings of the 2009 ICSE Workshop on Software Engineering for Secure Systems (IWSESS'09)', IEEE Computer Society, pp. 40–46. 65

Amir, R. and Zeid, A. (2004), A UML profile for service oriented architectures, *in* 'Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications', ACM, pp. 192–193. 136

Anderson, J. R. (1981), *Cognitive skills and their acquisition*, Lawrence Erlbaum. 30

Anjum, M. and Budgen, D. (2012*a*), A mapping study of the definitions used for service oriented architecture, *in* '16th International Conference on Evaluation Assessment in Software Engineering (EASE'12)', IET Press, pp. 57–61. 185

Anjum, M. and Budgen, D. (2012*b*), Modelling the design for an SOA system to control a small scale energy zone, *in* 'IEEE 36th Annual Computer Software and Applications Conference Workshops, 2012 (COMPSACW'12)', IEEE Computer Society, pp. 538–543. 75, 185

Ardissono, L., Furnari, R., Goy, A., Petrone, G. and Segnan, M. (2006), 'Fault tolerant web service orchestration by means of diagnosis', pp. 2–16. 73

Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S. and Holley, K. (2008), 'SOMA: A method for developing service-oriented solutions', *IBM Systems Journal* **47**(3), 377–396. 37

Bakker, H. and Iacob, M. E. (2009), Web-services in the dutch healthcare insurance sector: expected versus achieved benefits, *in* 'Proceedings of the 2009 ACM symposium on Applied Computing (SAC'09)', ACM, pp. 1617–1618. 65, 172

Baligand, F. and Monfort, V. (2004), A concrete solution for web services adaptability using policies and aspects, *in* '2nd international conference on Service oriented computing (ICSOC '04)', ACM, pp. 134–142. 25

Barata, J., Ribeiro, L. and Colombo, A. (2007), Diagnosis using service oriented architectures SOA, *in* '5th IEEE International Conference on Industrial Informatics', IEEE Computer Society, pp. 1203–1208. 65

Baresi, L., Ghezzi, C. and Guinea, S. (2004), Smart monitors for composed services, *in* 'Proceedings of the 2nd international conference on Service oriented computing', ACM, pp. 193–202. 73

Baresi, L., Ghezzi, C., Miele, A., Miraz, M., Naggi, A. and Pacifici, F. (2005), Hybrid service-oriented architectures: a case-study in the automotive domain, *in* 'Proceedings of the 5th international workshop on Software engineering and middleware (SEM '05)', ACM, pp. 62–68. 65, 172

Baresi, L., Heckel, R. and Sebastian Thöne, D. V. (2003), Modeling and validation of service-oriented architectures: application vs. style, *in* '9th European software engineering conference held jointly with 11th ACMSIGSOFT international symposium on Foundations of software engineering (ESEC/FSE-11)', ACM, pp. 68–77. 20, 58, 65, 66

Bertoldi, P., Boza-Kiss, B. and Rezessy, S. (2007), Latest development of energy service companies across europe, A european ESCO update, Institute for Environment and Sustainability. 193

Bertoldi, P., Rezessy, S. and Vine, E. (2006), 'Energy service companies in european countries: Current status and a strategy to foster their development', *Energy Policy* **34**, 1818–1832. 193

Bianco, P., Kotermanski, R. and Merson, P. F. (2007), Evaluating a service-oriented architecture, Technical report, CMU/SEI-2007-TR-015, Carnegie Mellon University. 101, 186

Bianculli, D. and Ghezzi, C. (2007), Monitoring conversational web services, *in* '2nd international workshop on Service oriented software engineering: in conjunction with the 6th ESEC/FSE joint meeting', ACM, pp. 15–21. 73

Bierhoff, K., Grechanik, M. and Liongosari, E. S. (2007), Architectural mismatch in service-oriented architectures, *in* 'International Workshop on Systems Development in SOA Environments (SDSOA'07)', IEEE Computer Society. 65

Bocchi, L. and Ciancarini, P. (2006), On the impact of formal methods in the SOA, *in* 'Electronic Notes in Theoretical Computer Science', Vol. 160, Elsevier, pp. 113–126. 65

Boer, R. C. D. and Farenhorst, R. (2008), In search of 'architectural knowledge', *in* 'Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge (SHARK'08)', ACM, pp. 71–78. 6, 68, 171

Bosnjak, A., Huang, S. and Mulcahy, J. J. (2011), Leveraging service oriented architecture: A case study for ocean energy information management, *in* 'IEEE International Conference on Information Reuse and Integration (IRI), 2011', pp. 108–112. 172

Bratthall, L. and Jørgensen, M. (2002), 'Can you trust a single data source exploratory software engineering case study?', *Empirical Software Engineering* **7**(1), 9–26. 169, 177

Braun, C. and Winter, R. (2007), Integration of IT service management into enterprise architecture, *in* 'ACM symposium on Applied computing 2007 (SAC'07)', ACM, pp. 1215–1219. 65

Braun, V. and Clarke, V. (2006), 'Using thematic analysis in psychology', *Qualitative research in psychology* **3**(2), 77–101. 59

Breivold, H. P. and Larsson, M. (2007), Component-based and service-oriented software engineering: Key concepts and principles, *in* '33rd EUROMICRO Conference on Software Engineering and Advanced Applications, 2007', IEEE, pp. 13–20. 3

Brereton, P. and Budgen, D. (2000), 'Component-based systems: a classification of issues', *IEEE Computer* **33**, 54–62. 4, 5, 13, 15

Brereton, P., Budgen, D., Bennnett, K., Munro, M., Layzell, P., MaCaulay, L., Griffiths, D. and Stannett, C. (1999), 'The future of software', *Communications of the ACM* **42**, 78–84. 2, 18

Brereton, P., Kitchenham, B., Budgen, D. and Li, Z. (2008), Using a protocol template for case study planning, *in* '12th International Conference on Evaluation and Assessment in Software Engineering (EASE'08)', pp. 1–8. 47

Briscoe, G. and Wilde, P. D. (2006), Digital ecosystems: Evolving service-orientated architectures, *in* '1st Bio-Inspired Models of Network, Information and Computing Systems.', IEEE, pp. 1–6. 65, 66

Brown, A. and Short, K. (1997), On components and objects: the foundations of component-based development, *in* 'Proceedings of the Fifth International Symposium on Assessment of Software Tools and Technologies, 1997', IEEE, pp. 112–121. 6, 12

Brown, A. W. and Wallnau, K. C. (1998), 'The current state of CBSE', *IEEE Software* **15**(5), 37–46. 13

Brykczynski, B. (1999), 'A survey of software inspection checklists', *ACM SIGSOFT Software Engineering Notes* **24**(1), 82–89. 225

Budgen, D. (2003), *Software Design*, 2nd edn, Pearson Addison Wesley. 39, 49, 67, 94, 95, 99, 225

Budgen, D. (2012), Conducting a semi-structured interview to identify issues for evidence-informed software engineering. Internal report. 226

Budgen, D., Brereton, P. and Turner, M. (2004), Codifying a service architectural style, *in* '28th Annual International Computer Software and Applications Conference (COMPSAC'04)', IEEE Computer Society, pp. 16–22. 2, 18

Budgen, D., Brereton, P., Turner, M. and Kitchenham, B. (2008), Using mapping studies in software engineering, *in* 'The 20th Annual Psychology of Programming Interest Group Conference', pp. 195–204. 52

Budgen, D., Burn, A., Brereton, P., Kitchenham, B. and Pretorius, R. (2011), 'Empirical evidence about the UML: A systematic literature review', *Software — Practice and Experience* **41**(4), 363–392. 40, 186

Budgen, D. and Zhang, C. (2009), Preliminary reporting guidelines for experience papers, *in* '13th International Conference on Evaluation and Assessment in Software Engineering (EASE'09)'. 55

Bull, S. (2001), Renewable energy today and tomorrow, *in* 'Proceedings of the IEEE', National Renewable Energy Laboratory, Golden, CO, IEEE, pp. 1216–1226. 192

Candido, G., Barata, J., Colombo, A. W. and FrançoisJammes (2009), 'SOA in reconfigurable supply chains: A research roadmap', *Engineering Applications of Artificial Intelligence* **22**(6), 939–949. 65

Canfora, G., Fasolino, A. R., Frattolillo, G. and Tramontana, P. (2008), 'A wrapping approach for migrating legacy system interactive functionalities to service oriented architectures', *Journal of Systems and Software* **81**(4), 463–480. 65

Card, D. N., Garry, F. E. M. and Page, G. T. (1987), 'Evaluating software engineering technologies', *IEEE Transactions on Software Engineering* **13**(7), 845–851. 152

Carney, D. and Long, F. (2000), 'What do you mean by COTS? finally, a useful answer', *IEEE Software* **17**(2), 83–86. 14, 15

Casola, V., Mancini, E. P., Mazzocca, N., Rak, M. and Villano, U. (2008), 'Self-optimization of secure web services', *Computer Communications* **31**(18), 4312–4323. 57

Cervantes, H. and Hall, R. (2005), Technical concepts of service orientation, *in* 'Service-oriented software system engineering: challenges and practices', IGI Global, p. 1. 27

Chang, S. H. (2007), A systematic analysis and design approach to develop adaptable services in service oriented computing, *in* '2007 IEEE Congress on Services', pp. 375–378. 36

Chen, X., Cai, W., Turner, S. and Wang, Y. (2006), SOAr-DSGrid: Service-oriented architecture for distributed simulation, *in* '20th Workshop on Principles of Advanced and Distributed Simulation (PADS'06)', IEEE Computer Society, pp. 65–73. 18, 21

Chmielewski, U., Brinkman, R. and Hoepman, J.-H. (2008), Using JASON to secure SOA, *in* 'Proceedings of the 2008 workshop on Middleware security (MidSec'08)', ACM, pp. 13–18. 65

Choi, S. W., Her, J. S. and Kim, S. D. (2007), Modeling QoS attributes and metrics for evaluating services in SOA considering consumers: Perspective as the first class requirement, *in* 'Proceedings of the The 2nd IEEE Asia-Pacific Service Computing Conference (APSCC'07)', IEEE Computer Society, pp. 398–405. 57, 65

Cipcigan, L., Taylor, P. and Lyons, P. (2009), 'A dynamic virtual power station model comprising small-scale energy zones', *International Journal of Renewable Energy Technology* **1**, 173–191. 194, 195

Cotroneo, D., Graziano, A. and Russo, S. (2004), Security requirements in service oriented architectures for ubiquitous computing, *in* '2nd workshop on Middleware for pervasive and ad-hoc computing, ACM International Conference Proceeding Series', ACM, pp. 172–177. 18

Crnkovic, I. and Larsson, M. (2000), A case study: demands on component-based development, *in* 'Proceedings of the International Conference on Software Engineering, 2000', pp. 23–31. 4, 5, 17

Crnkovic, I. and Larsson, M. (2002), 'Challenges of component-based development', *Journal of Systems and Software* **61**(3), 201–212. 5, 16

Cruzes, D. S. and Dybå, T. (2010), Synthesizing evidence in software engineering research, *in* 'Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement', ACM. 60

Cruzes, D. S. and Dybå, T. (2011), 'Research synthesis in software engineering: A tertiary study', *Information and Software Technology* **53**, 440–455. 45, 59, 60, 167, 168

Cusumano, M. (2010), 'Cloud computing and SaaS as new computing platforms', *Communications of the ACM* **53**(4), 27–29. 19

Dan, X., Shi, Y., Tao, Z., Xiang-Yang, J. and Jun-Feng, L. Z.-Q. Y. (2006), An approach for describing SOA, *in* 'International Conference on Wireless Communications, Networking and Mobile Computing, 2006 (WiCOM 2006)', IEEE Computer Society, pp. 1–4. 65

DeMarco, T. (1979), *Structured analysis and system specification*, Yourdon Press. 109

Demirkan, H., Kauffman, R. J., Vayghan, J. A., Hans-GeorgFill, Karagiannis, D. and Maglio, P. P. (2008), 'Service-oriented technology and management: Perspectives on research andpractice for the coming decade', *Electronic Commerce Research and Applications* **7**(4), 356–376. 65

Depuru, S. S. S. R., Wang, L., Devabhaktuni, V. and Gudi, N. (2011), 'Smart meters for power grid: Challenges, issues, advantages and status', *Renewable and sustainable energy reviews* **15**(6), 2736–2742. 88

Dietrich, A. J., Kirn, S. and Sugumaran, V. (2007), 'A service-oriented architecture for mass customization: A shoe industry case study', *IEEE Transactions on Engineering Management* **54**(1), 190–204. 172

Dijkstra, E. W. (1976), *A discipline of programming*, Vol. 1, Prentice-Lall Englewood Cliffs, NJ. 28

Dimitrov, V. (2008), Development of applications with service-oriented architecture for grid, *in* 'Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing (CompSysTech'08)', ACM, pp. 1–6. 65

Duan, Q. (2009), Applying the service-oriented architecture for network discovery and selection in the next generation wireless mobile networks, *in* 'International Conference on Network-Based Information Systems (NBIS'09)', IEEE Computer Society, pp. 380–385. 65, 172

Duan, S. and Yuan, X. (2007), Software IC revised: A new approach of component-based software design with software slots, *in* 'Sixth International

IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, 2007 (ICCBSS'07)', pp. 73–81. 17

Dybå, T., Dingsyr, T. and Hanssen, G. K. (2007), Applying systematic reviews to diverse study types: An experience report, *in* 'First International Symposium on Empirical Software Engineering and Measurement, 2007 (ESEM'07)', pp. 225–234. 44

Dybå, T., Kitchenham, B. A. and Jrgensen, M. (2005), 'Evidence-based software engineering for practitioners', *IEEE Software* **22**(1), 58–65. 44

EBSE (2013). `http://www.dur.ac.uk/ebse/`. 177, 178

EnergyLibrary (2011). `http://theenergylibrary.com/node/3863`. 83

ERCIM-NEWS (2013), 'Special theme: Smart energy systems', *ERCIM-NEWS 92* . `http://ercim-news.ercim.eu/`. 172

Erl, T. (2004), *Service-Oriented Architecture: A Field Guide to Integrating XML and WebServices*, Prentice Hall. 66

Erl, T. (2005), *Service-Oriented Architecture (SOA): Concepts, Technology, and Design*, Prentice Hall. 66

Erl, T. (2009), *SOA Design Patterns*, Prentice Hall. 139

Erradi, A., Anand, S. and Kulkarni, N. (2006), Evaluation of strategies for integrating legacy applications as services in a service oriented architecture, *in* 'IEEE International Conference on Services Computing (SCC'06)', IEEE Computer Society, pp. 257–260. 36, 65

Erradi, A. and Sriram Anand, N. K. (2006), SOAF: An architectural framework for service definition and realization, *in* 'IEEE International Conference on Services Computing, 2006 (SCC'06)', pp. 151–158. 136

Espinha, E., Chen, C., Zaidman, A. and Gross, H.-G. (2012), Maintenance research in SOA - towards a standard case study, *in* '16th European Conference on Software Maintenance and Reengineering, 2012 (CSMR'12)', IEEE, pp. 391–396. 72, 73, 172, 173

Expsito, A. G., Ramos, J. L. M. and Santos, J. R. (2004), 'Slack bus selection to minimize the system power imbalance in load-flow studies', *IEEE Transactions on Power Systems* **19**(2), 987–995. 85

Fagan, M. E. (1976), 'Design and code inspections to reduce errors in program development', *IBM Systems Journal* **38**(2.3), 258–287. 48, 223, 224

Fichman, R. G. and Kemerer, C. F. (1992), 'Object-oriented and conventional analysis and design methodologies', *Computer* **25**(10), 22–39. 31, 32

Fornasa, M., Zingirian, N., Maresca, M. and Baglietto, P. (2006), VISIONS: A service oriented architecture for remote vehicle inspection, *in* 'IEEE Intelligent Transportation Systems Conference (ITSC'06)', IEEE, pp. 163–168. 65

Fujii, K. and Suda, T. (2005), 'Semantics-based dynamic service composition', *IEEE Journal on Selected Areas in Communications* **23**(12), 2361–2372. 23

Gao, Z. and Tang, B. (2007), Research on service oriented platform of integration application for textile industry in China, *in* 'IEEE International Conference on e-Business Engineering (ICEBE'07)', IEEE Computer Society., pp. 371–374. 65, 172

Garlan, D., Allen, R. and Ockerbloom, J. (1995), 'Architectural mismatch: why reuse is so hard', *IEEE Software* **12**(6), 17–26. 4, 16

Garousi, V. (2010), 'Applying peer reviews in software engineering education: An experiment and lessons learned', *IEEE Transactions on Education* **53**(2), 182–193. 170, 174, 223

Garrison, J. B. and Webber, M. E. (2011), 'An integrated energy storage scheme for a dispatchable solar and wind powered energy system', *Journal of Renewable and Sustainable Energy* **3**(4), 043101. 83

Gasikanti, A., Thomas, J. P. and Thomas, M. (2007), Information discovery across organizational boundaries through local caching, *in* 'IEEE International Conference on Services Computing (SCC'07)', IEEE Computer Society, pp. 522–529. 57, 65

Geisterfer, C. M. and Ghosh, S. (2006), Software component specification: a study in perspective of component selection and reuse, *in* 'Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems, 2006', p. 9. 4, 16

Gellings, C. (1985), 'The concept of demand-side management for electric utilities', *Proceedings of the IEEE* **73**(10), 1468–1470. 195

Gogouvitis, S. V., Kousiouris, G., Konstanteli, K., Polychniatis, T., Menychtas, A., Kyriazis, D. and Varvarigou, T. (2008), Realtime-enabled workflow management in service oriented infrastructures, *in* 'AREA '08: Proceeding of the 1st ACM workshop on Analysis and retrieval ofevents/actions and workflows in video streams', ACM, pp. 119–124. 65

Gooneratne, N. and Tari, Z. (2008), Matching independent global constraints for composite web services, *in* 'Proceeding of the 17th international conference on World Wide Web (WWW'08)', ACM, pp. 765–774. 23

Green, T. and Blackwell, A. (1998), Cognitive dimensions of information artefacts: a tutorial, *in* 'BCS HCI Conference'. 39

Griffin, D. and Pesch, D. (2007), 'A survey on web services in telecommunications', *IEEE Communications Magazine* **45**, 28–35. 65

Gu, Q. and Lago, P. (2007), A stakeholder-driven service life cycle model for soa, *in* '2nd international workshop on Service oriented software engineering: in conjunction with the 6th ESEC/FSE joint meeting', ACM, New York, NY, USA, pp. 1–7. 24, 36, 136

Gu, Q. and Lago, P. (2009), 'Exploring service-oriented system engineering challenges: a systematic literature review', *Service Oriented Computing and Applications* **3**(3), 171–188. 68

Gu, Q. and Lago, P. (2011), 'Guiding the selection of service-oriented software engineering methodologies', *Service Oriented Computing and Applications* **5**(4), 203–223. 35, 37

Gu, Q. and van Vliet, H. (2009), SOA decision making - what do we need to know, *in* 'Proceedings of the 2009 ICSE Workshop on Sharing and Reusing Architectural Knowledge (SHARK'09)', IEEE Computer Society, pp. 25–32. 65

Guindon, R. (1990*a*), 'Designing the design process: Exploiting opportunistic thoughts', *Human-Computer Interaction* **5**(2), 305–344. 30

Guindon, R. (1990*b*), 'Knowledge exploited by experts during software system design', *International Journal of Man-Machine Studies* **33**(3), 279–304. 29, 30, 38, 95, 96

Guindon, R. and Curtis, B. (1988), Control of cognitive processes during software design: what tools are needed?, *in* 'Proceedings of the SIGCHI Conference on Human Factors in Computing Systems', ACM, New York, NY, USA, pp. 263–268. 29

Harrison, A. and Taylor, I. J. (2005), WSPeer - an interface to web service hosting and invocation, *in* '19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)', Vol. 5, IEEE Computer Society, pp. 175a–175a. 25, 51

Heineman, G. T. and Councill, W. T. (2001), *Component-Based Software Engineering: Putting the Pieces Together*, Addison-Wesley Professional. 6, 12

Henningsson, S., Svensson, C. and Vallen, L. (2007), Mastering the integration chaos following frequent m&as: Is integrationwith SOA technology, *in* 'Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07)', IEEE Computer Society. 65

Hoel, T. (2006), Service oriented architectures without openness - a contradiction of terms.reflection on the Norwegian situation, *in* '6th IEEE International Conference on Advanced Learning Technologies (ICALT'06)', IEEE Computer Society, pp. 883–885. 65

Hopkins, J. (2000), 'Component primer', *Communications of the ACM* **43**(10), 27–30. 13

Hourdin, V., Tigli, J.-Y., Lavirotte, S. and Riveill, G. R. M. (2008), SLCA, composite services for ubiquitous computing, *in* 'Proceedings of the International Conference on Mobile Technology, Applications,and Systems (Mobility'08)', ACM. 65

Howerton, J. T. (2007), 'Service-oriented architecture and web 2.0', *IT Professional* **9**(3). 65

Hrastnik, P. and Winiwarter, W. (2007), Coordination in service oriented architectures using transaction processing concepts, *in* '18th International Workshop on Database and Expert Systems Applications (DEXA'07)', IEEE, pp. 855–860. 65

Huang, S. and Fan, Y. (2007), Model driven and service oriented enterprise integration - the method, framework and platform, *in* '6th International Advanced Language Processing and Web Information Technology (ALPIT'07)', IEEE Computer Society, pp. 504–509. 65, 66

Huhns, M. and Singh, M. P. (2005), 'Service-oriented computing: key concepts and principles', *IEEE Internet Computing* **9**(1), 75–81. xiv, 20

IEEESTD (2008), 'IEEE standard for software reviews and audits', *IEEE STD 1028-2008* pp. 1–52. 145, 178, 224, 226

Iivari, J. (1995), 'Object-orientation as structural, functional and behavioural modelling: a comparison of six methods for object-oriented analysis', *Information and Software Technology* **37**(3), 155–163. xvii, 33, 34

Jammes, F., Mensch, A. and Smit, H. (2005), Service-oriented device communications using the devices profile for webservices, *in* '3rd international workshop on Middleware for pervasive and ad-hoc computing (MPAC'05)', ACM, pp. 1–8. 56

Jardim-Goncalves, R., Grilo, A. and Steiger-Garcao, A. (2006), 'Challenging the interoperability between computers in industry with MDA and SOA', *Computers in Industry* **57**(8-9), 679–689. 65

Jørstad, I., Dustdar, S. and Thanh, D. V. (2005), A service oriented architecture framework for collaborative services, *in* '14th IEEE International Workshops on Enabling Technologies: Infrastructurefor Collaborative Enterprise (WET-ICE05)', IEEE Computer Society, pp. 121–125. 65

Karhunen, H., Jantti, M. and Eerola, A. (2005), Service-oriented software engineering (SOSE) framework, *in* 'Proceedings of the International Conference on Services Systems and Services Management, 2005 (ICSSSM'05)', Vol. 2, pp. 1199–1204. 35, 136

Kemerer, C. F. and Paulk, M. C. (2009), 'The impact of design and dode reviews on software quality: An empirical study based on PSP data', *IEEE Transactions on Software Engineering* **35**(4), 534–550. 48

Khoshnevis, S., Aliee, F. S. and Jamshidi, P. (2009), Model driven approach to service oriented enterprise architecture, *in* 'IEEE Asia-Pacific Services Computing Conference (APSCC 2009)', IEEE, pp. 279–286. 65

Kim, J. and Lerch, F. J. (1992), Towards a model of cognitive process in logical design: comparing object-oriented and traditional functional decomposition software methodologies, *in* 'Proceedings of the SIGCHI conference on Human factors in computing systems', ACM, pp. 489–498. 29, 94

Kim, T., Lee, H. and Cheon, H. (2007), Implementation of a service oriented architecture based on JXTA for new business models, *in* 'International Conference on Control, Automation and Systems, 2007 (ICCAS'07)', IEEE, pp. 2402–2406. 65

Kitchenham, B. A., Budgen, D. and Brereton, O. P. (2011), 'Using mapping studies as the basis for further research—a participant-observer case study', *Information and Software Technology* **53**(4), 638–651. Special section from EASE 2010. 45, 168

Kitchenham, B., Brereton, P., Budgen, D., Turner, M., Bailey, J. and Linkman, S. (2009), 'Systematic literature reviews in software engineering — a systematic literature review', *Information and Software Technology* **51**(1), 7–15. 70

Kitchenham, B. and Charters, S. (2007), Guidelines for performing systematic literature reviews in software engineering, Technical Report EBSE 2007-001, Keele University and Durham University Joint Report. 44, 45, 52, 53

Komoda, N. (2006), Service oriented architecture (SOA) in industrial systems, *in* 'IEEE International Conference on Industrial Informatics', IEEE, pp. 1–5. 56

Kontogiannis, K., Lewis, G. A. and Smith, D. B. (2008), A research agenda for service-oriented architecture, *in* 'Proceedings of the 2nd international workshop on Systems developmentin SOA environments (SDSOA'08)', ACM, pp. 1–6. 26, 27

Korotkiy, M. and Top, J. (2006), Onto-SOA: From ontology-enabled SOA to service-enabled ontologies, *in* 'International Conference on Internet and Web Applications and Services/Advanced International Conference on Telecommunications 2006 (AICT-ICIW'06)', IEEE Computer Society. 25

Kumar, S., Dakshinamoorthy, V. and Krishnan, M. S. (2007), Does SOA improve the supply chain? an empirical analysis of the impact of SOA adoption on electronic supply chain performance, *in* 'Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS'07)', IEEE Computer Society. 65

Kumaran, S., Chao, T., Bhattacharya, K. and Dhoolia, P. (2007), A model driven framework for it transformnation, *in* '2nd IEEE/IFIP International Workshop on Business-Driven IT Management (BDIM'07)', pp. 55–64. 65

Lane, S. and Richardson, I. (2011), 'Process models for service-based applications: A systematic literature review', *Information and Software Technology* **53**(5), 424–439. 8

Laplante, P. A., Zhang, J. and Voas, J. (2008), 'What's in a name? distinguishing between SaaS and SOA', *IT Professional* **10**(3), 46–50. 2, 3, 19

Lewis, G. (2010), Getting started with service-oriented architecture (SOA) terminology, Technical report, Software Engineering Institute. 69

Lewis, G. and Smith, D. (2008), Service-oriented architecture and its implications for software maintenance and evolution, *in* 'Frontiers of Software Maintenance, 2008 (FoSM'08)', pp. 1–10. xvii, 18, 21, 25, 26, 27

Liang, Q. and Chung, J. Y. (2007), Analyzing service usage patterns: Methodology and simulation, *in* 'Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE'07)', IEEE, pp. 359–362. 65

Liu, X. and Deters, R. (2007), An efficient dual caching strategy for web service-enabled PDAs, *in* 'ACM symposium on Applied computing (SAC'07)', ACM, pp. 788–794. 65

Locola, P. (2007), When legacy meets SOA: Achieving business agility by integrating new technology with existing software asset, *in* '1st Annual IEEE Systems Conference', IEEE, pp. 1–8. 65

López-Sanz, M., Acuña, C. J., Cuesta, C. E. and Marcos, E. (2008), 'Modelling of service-oriented architectures with UML', *Electronic Notes in Theoretical Computer Science* **194**(4), 23–37. 136

Loy, P. H. (1990), 'A comparison of object-oriented and structured development methods', *ACM SIGSOFT Software Engineering Notes* **15**(1), 44–48. 3, 29, 137

Luthria, H., Rabhi, F. and Briers, M. (2007), Investigating the potential of service oriented architectures to realize dynamic capabilities, *in* 'The 2nd IEEE Asia-Pacific Service Computing Conference', IEEE, pp. 390–397. 65

Mahmood, S., Lai, R. and Kim, Y. (2007), 'Survey of component-based software development', *IET Software* **1**(2), 57–66. 17

Mandić, V., Markkula, J. and Oivo, M. (2009), 'Towards multi-method research approach in empirical software engineering', *Product-Focused Software Process Improvement* **32**, 96–110. 41

Massuthe, P. and Schmidt, K. (2005), Operating guidelines - an automata-theoretic foundation for the service-oriented architecture, *in* '5th International

Conference on Quality Software (QSIC05)', IEEE Computer Society, pp. 452–457. 58, 65

Mayrhauser, A. V. and Vans, A. M. (1995), 'Program comprehension during software maintenance and evolution', *Computer* **28**(8), 44–55. 30

McLeod, L., MacDonell, S. G. and Doolin, B. (2011), 'Qualitative research on software development: a longitudinal case study methodology', *Empirical Software Engineering* **16**(4), 430–459. 169, 170, 173

Mendes, J. M., Leitão, P., Colombo, A. W. and Restivo, F. (2008), Service-oriented control architecture for reconfigurable production systems, *in* '6th IEEE International Conference on Industrial Informatics (INDIN'08)', IEEE, pp. 744–749. 65

Michiorri, A. and Taylor, P. (2009), Forecasting real-time ratings for electricity distribution networks using weather forecast data, *in* '20th International Conference and Exhibition on Electricity Distribution - Part 1, 2009 (CIRED'09)', pp. 1–4. 85

Miller, J. (2008), 'Triangulation as a basis for knowledge discovery in software engineering', *Empirical Software Engineering* **13**(2), 223–228. 167

Mingers, J. (2001), 'Combining is research methods: towards a pluralist methodology', *Information systems research* **12**(3), 240–259. 41

Mingers, J. (2003), 'The paucity of multimethod research: a review of the information systems literature', *Information Systems Journal* **13**(3), 233–249. 41, 174

Moe, N. B., Dingsøyr, T. and Dybå, T. (2010), 'A teamwork model for understanding an agile team: A case study of a scrum project', *Information and Software Technology* **52**(5), 480–491. 173

Moody, D. L. (2009), 'The "physics" of notations: Toward a scientific basis for constructing visual notations in software engineering', *IEEE Transactions on Software Engineering* **35**(6), 756–779. 39, 40, 109

Muller, B. (2012), 'Energy from everywhere', *Pictures of the Future, the Magazine for Research an Innovation* pp. 68–69. 74, 172

Mykkanen, J., Riekkinen, A., Sormunen, M., Karhunen, H. and Laitinen, P. (2007), 'Designing web services in health information systems: From process to application level', *International Journal of Medical Informatics* **76**(2-3), 89–95. 57

Nakamura, M., Igaki, H., Tamada, H. and Kenichi (2004), Implementing integrated services of networked home appliances using service oriented architecture, *in* 'Proceedings of the 2nd international conference on Service oriented computing (ICSOC'04)', ACM, pp. 269–278. 65

Nestler, T. (2008), Towards a mashup-driven end-user programming of SOA-based applications, *in* 'Proceedings of the 10th International Conference on Information Integration and Web-based Applications and Services (IIWAS'08)', ACM, pp. 551–554. 65

Niemann, M., Janiesch, C., Repp, N. and Steinmetz, R. (2009), Challenges of governance approaches for service-oriented architectures, *in* '3rd IEEE International Conference on Digital Ecosystems and Technologies (DEST'09)', pp. 600–605. 26

Oates, B. J. (2005), *Researching information systems and computing*, Sage Publications Limited. 49, 77, 142, 168

O'Brien, L., Brebner, P. and Gray, J. (2008), Business transformation to SOA: aspects of the migration and performance and QoS issues, *in* 'Proceedings of the 2nd international workshop on Systems development in SOA environments (SDSOA'08)', ACM, pp. 35–40. 26

O'Brien Lero, L., Merson, P. and Bass, L. (2007), Quality attributes for service-oriented architectures, *in* 'International Workshop on Systems Development in SOA Environments, 2007 (SDSOA'07): ICSE Workshops 2007', p. 3. 25

Offermann, P. and Bub, U. (2009), Empirical comparison of methods for information systems development according to SOA, *in* 'Proceedings of 17th European Conference on Information Systems (ECIS'09)'. 37, 136

Ofgem (2012). `http://www.ofgem.gov.uk/Networks/ElecDist/Pages/ElecDist.aspx`. 196

Oikonomou, V. and Mundaca, L. (2008), 'Tradable white certificate schemes: what can we learn from tradable greencertificate schemes?', *Energy Efficiency* **1**, 211–232. 194

Oliveira, L. B. R. D., Felizardo, K. R. and Nakagawa, D. F. E. Y. (2010), 'Reference models and reference architectures based on service-oriented architecture: A systematic review', *Software Architecture - Lecture Notes in Computer Science* **6285/2010**, 360–367. 8, 68

Panahi, M., Nie, W. and Lin, K.-J. (2009), A framework for real-time service-oriented architecture, *in* 'IEEE Conference on Commerce and Enterprise Computing (CEC'09)', IEEE Computer Society, pp. 460–467. 57

Papagianni, C., Karagiannis, G., Tselikas, N. D., andI.P. Chochliouros, E. S., Kabilafkas, D., Cinkler, T., andP. Sjödin, L. W., Hidell, M., de Groot, S. H., Kontos, T., Pappas, C. C., Antonakopoulou, A. and Venieris, I. S. (2008), Supporting end-to-end resource virtualization for web 2.0 applications using service oriented architecture, *in* 'IEEE GLOBECOM Workshops', IEEE, pp. 1–7. 65

Papazoglou, M. P. (2003), Service-oriented computing: Concepts, characteristics and directions, *in* '4th International Conference on Web Information Systems Engineering (WISE'03)', IEEE Computer Society. 56

Papazoglou, M. P. and Heuvel, W.-J. (2007), 'Service oriented architectures: approaches, technologies and research issues', **16**, 389–415. 3, 25, 138

Papazoglou, M. P. and Heuvel, W.-J. V. D. (2006), 'Service-oriented design and development methodology', *International Journal of Web Engineering and Technology* **2**(4), 412–442. 36, 136

Parnas, D. L. (1972), 'On the criteria to be used in decomposing systems into modules', *Communications of the ACM* **15**(12), 1053–1058. 3, 28

Parnas, D. L. and Weiss, D. M. (1985), Active design reviews: principles and practices, *in* 'Proceedings of the 8th International Conference on Software engineering (ICSE'85)', IEEE Computer Society Press, pp. 132–136. 224

Pautasso, C., Zimmermann, O. and Leymann, F. (2008), Restful web services vs. "big" web services: making the right architectural decision, *in* 'Proceedings of the 17th International Conference on World Wide Web', ACM, New York, NY, USA, pp. 805–814. 26

Perry, D. E., Sim, S. E. and Easterbrook, S. M. (2004), Case studies for software engineers, *in* 'Proceedings of the 26th International Conference on Software engineering (ICSE'04)', IEEE, pp. 736–738. 170, 190

Petre, M. (2009), Insights from expert software design practice, *in* 'Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on the foundations of software engineering (ESEC/FSE'09)', ACM, pp. 233–242. 29, 31, 38

Petter, S. C. and Gallivan, M. J. (2004), Toward a framework for classifying and guiding mixed method research in information systems, *in* 'Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004'. 41

Petticrew, M. and Roberts, H. (2006), *Systematic Reviews in the Social Sciences: A Practical Guide*, Wiley-Blackwell. 45, 52

Pichitlamken, J., Uthayopas, P., Kajkamhaeng, S. and Tippayawannakorn, N. (2007), Service-oriented architecture on a windows cluster for spreadsheet simulation, *in* 'IEEE International Conference on Industrial Engineering and EngineeringManagement', IEEE, pp. 1757–1761. 65

Pohthong, A. and Budgen, D. (2001), 'Reuse strategies in software development: an empirical study', *Information and Software Technology* **43**(9), 561–575. 29

Prinsloo, J. M., Schulz, C. L., Kourie, D. G., Theunissen, W. H., Strauss, T., Heever, R. V. D. and Grobbelaar, S. (2006), A service oriented architecture for wireless sensor and actor network applications, *in* 'Annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries (SAICSIT'06)', Vol. 204, ACM International Conference Proceeding Series, South African Institute for Computer Scientists and Information Technologists, pp. 145–154. 18, 65

Qing-Ming, W., Yong, T. and Zan-Bo, Z. (2009), Research in enterprise applications of dynamic web service composition methods and models, *in* 'Second International Symposium on Electronic Commerce and Security, 2009 (ISECS '09)', Vol. 1, pp. 146–150. 22

Rao, J. and Su, X. (2004), A survey of automated web service composition methods, *in* 'Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, 2004', Springer, pp. 43–54. 23

Reeves, A., Marashi, M. and Budgen, D. (1995), 'A software design framework or how to support real designers', *Software Engineering Journal* **10**(4), 141–155. 29

Ricci, A., Buda, C. and Zaghini, N. (2007), An agent-oriented programming model for SOA & web services, *in* '5th IEEE International Conference on Industrial Informatics', IEEE, pp. 1059–1064. 65

Roach, T., Low, G. and D'Ambra, J. (2008), CAPSICUM - a conceptual model for service oriented architecture, *in* 'Proceedings of the 2008 IEEE Congress on Services (SERVICES'08)- Part I', IEEE Computer Society, pp. 415–422. 65

Roberts, D., Taylor, P. and Michiorri, A. (2008), Dynamic thermal rating for increasing network capacity and delaying network reinforcements, *in* 'Smart Grids for Distribution, 2008. IET-CIRED. CIRED Seminar', pp. 1–4. 85

Runeson, P. and Höst, M. (2009), 'Guidelines for conducting and reporting case study research in software engineering', *Empirical Software Engineering* **14**, 131–164. 46, 72, 173, 190

Runeson, P., Host, M., Rainer, A. and Regnell, B. (2012), *Case Study Research in Software Engineering: Guidelines and Examples*, Wiley. 46, 170, 177, 190

Sabucedo, L. M. Á., Rifón, L. E. A. and Gago, R. M. P. J. M. S. (2009), 'Providing standard-oriented data models and interfaces to eGovernment services a semantic-driven approach', *Computer Standards and Interfaces* **31**(5), 1014–1027. 65

Schepers, T. G. J., Iacob, M. E. and Eck, P. A. T. V. (2008), A lifecycle approach to SOA governance, *in* 'Proceedings of the 2008 ACM symposium on Applied computing (SAC '08)', ACM, pp. 1055–1061. 65

Schroeter, J., Cech, S., Gtz, S., Wilke, C. and Amann, U. (2012), Towards modeling a variable architecture for multi-tenant SaaS-applications, *in* 'Proceedings of the Sixth International Workshop on Variability Modeling of Software-Intensive Systems (VaMoS'12)', ACM, pp. 111–120. 19

Schroh, D., Bozowsky, N., Savigny, M. and Wright, W. (2009), nCompass service oriented architecture for tacit collaboration services, *in* '13th International Conference Information Visualisation (IV'09)', IEEE Computer Society, pp. 434–442. 57, 65

Schuschel, H. and Weske, M. (2004), Automated planning in a service-oriented architecture, *in* '13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE '04)', IEEE Computer Society, pp. 75–80. 20

Seaman, C. B. (1999), 'Qualitative methods in empirical studies of software engineering', *IEEE Transactions on Software Engineering* **25**(4), 557–572. 142, 151, 168, 178, 179, 228

Sharp, J. H. and Ryan, S. D. (2010), 'A theoretical framework of component-based software development phases', *SIGMIS Database* **41**(1), 56–75. 12, 27

Shaw, M. and Clements, P. C. (1997), A field guide to boxology: Preliminary classification of architectural styles for software systems, *in* 'Proceedings of the

21st International Computer Software andApplications Conference (COMPSAC'97)', pp. 6–13. 2, 6, 68

Sinden, G. (2007), 'Characteristics of the UK wind resource: long-term patterns and relationship to electricity demand', *Energy Policy* **35**(1), 112–127. 83

Sjoberg, D. I. K., Dyba, T. and Jorgensen, M. (2007), The future of empirical methods in software engineering research, *in* 'Future of Software Engineering, 2007. FOSE '07', pp. 358 –378. 168

Skogan, D., Grnmo, R. and Solheim, I. (2004), Web service composition in UML, *in* 'Proceedings of the Eighth IEEE International Enterprise Distributed Object Computing Conference, 2004 (EDOC'04)', pp. 47 – 57. 140

Smith, G., Onions, P. and Infield, D. (2000), 'Predicting islanding operation of grid connected pv inverters', *IEE Proceedings - Electric Power Applications* **147**(1), 1–6. 196

Sommerville, I. and Kotonya, G. (1998), *Requirements Engineering: Processes and Techniques*, John Wiley & Sons, Inc. 15

Soni, A. and Özveren, C. (2006), Improved control of isolated power system by the use of feeding technique, *in* 'Proceedings of the 41st International Universities Power Engineering Conference', University of Abertay, Dundee, IEEE, pp. 974–977. 193

Soni, A. and Özveren, C. (2007), Renewable energy market potential in U.K., *in* '42nd International Universities Power Engineering Conference, 2007', University of Abertay, Dundee, pp. 717–720. 193

Stojanovic, Z., Dahanayake, A. and Sol, H. (2004), Modeling and design of service-oriented architecture, *in* 'IEEE International Conference on Systems, Man and Cybernetics, 2004', Vol. 5, IEEE, pp. 4147–4152. 136, 138

Sun, J. and Chen, Y. (2008), Intelligent enterprise information security architecture based on serviceoriented architecture, *in* 'International Seminar on Future Information Technology and Management Engineering (FITME'08)', IEEE Computer Society, pp. 196–200. 65

Sward, R. E. (2007), Using Ada in a service-oriented architecture, *in* 'ACM International Conference on SIGAda (SIGAda'07)', ACM, pp. 63–67. 65

Szyperski, C., Gruntz, D. and Murer, S. (2002), *Component Software: Beyond Object-Oriented Programming*, 2nd edn, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 4, 6, 12, 13, 15, 17

Talaei-Khoei, A., Sheriffian, A. H. and andJavad Farzaneh Verdom, M. K. A. (2005), A new approach for service-oriented architecture, *in* 'Enabling Technologies for the New Knowledge Society: ITI 3rd International Conference on Information and Communications Technology', IEEE, pp. 459–470. 56

Taylor, R. N. and der Hoek, A. V. (2007), Software design and architecture the once and future focus of software engineering, *in* '2007 Future of Software Engineering', IEEE Computer Society, pp. 226–243. 39

Teiniker, E., Schmoelzer, G., Faschingbauer, J., Kreiner, C. and Weiss, R. (2005), A hybrid component-based system development process, *in* '31st EUROMICRO Conference on Software Engineering and Advanced Applications, 2005', pp. 152–159. 16

Tewoldeberhan, T. and Janssen, M. (2008), 'Simulation-based experimentation for designing reliable and efficient webservice orchestrations in supply chains', *Electronic Commerce Research and Applications* **7**(1), 82–92. 65

Trichakis, P., Taylor, P. C., Lyons, P. and RichardHair (2009), Transforming low-voltage networks into small-scale energy zones, *in* 'Proceedings of the ICE - Energy', Vol. 162, Proceedings of the Institution of Civil Engineers, pp. 37–46. 195

Trichakis, P., Taylor, P., Coates, G. and Cipcigan, L. M. (2008), 'Distributed control approach for small-scale energy zones', *Journal of Power and Energy* **222**(2), 137–147. 83, 85, 86, 196, 197

Tsai, W. T. (2005), Service-oriented system engineering: a new paradigm, *in* 'IEEE International on WorkshopService-Oriented System Engineering, 2005 (SOSE'05)', pp. 3–6. 35

Tsai, W.-T., Shao, Q., Huang, Y. and Ba, X. (2010), Towards a scalable and robust multi-tenancy SaaS, *in* 'Proceedings of the Second Asia-Pacific Symposium on Internetware', ACM, pp. 1–15. 19

Turner, M., Budgen, D. and Brereton, P. (2003), 'Turning software into a service', *IEEE Computer* **36**(10), 33–44. 2, 19

Valipour, M. H., Amirzafari, B., Maleki, K. N. and Daneshpour, N. (2009), A brief survey of software architecture concepts and service oriented architecture, *in* '2nd IEEE International Conference on Computer Science and Information Technology (ICCSIT'09)', IEEE, pp. 34–38. 65

Van der Hoek, A. and Lopez, N. (2011), A design perspective on modularity, *in* 'Proceedings of the tenth international conference on Aspect-oriented software development', ACM, New York, NY, USA, pp. 265–280. 28, 31

Venables, M. (2012), 'Surviving Sandy-smart technologies help the recovery', *Engineering & Technology (E&T)* **7**, 20–21. 74, 172

Visser, W. and Hoc, J.-M. (1990), Expert software design strategies, *in* 'Psychology of programming', New York, NY: Harcourt Brace Jovanovich, pp. 235–249. 7, 29, 30, 95

Vitharana, P. (2003), 'Risks and challenges of component-based software development', *Communications of the ACM* **46**(8), 67–72. 16

Vitharana, P., Zahedi, F. M. and hemant Jain (2003), 'Design, retrieval, and assembly in component-based software development', *Communications of the ACM* **46**(11), 97–102. 17

Wada, H., Suzuki, J. and Oba, K. (2006), Modeling non-functional aspects in service oriented architecture, *in* 'IEEE International Conference on Services Computing, 2006 (SCC'06)', IEEE, pp. 222–229. 136

Wagner, S. and Deissenboeck, F. (2008), Abstractness, specificity, and complexity in software design, *in* 'Proceedings of the 2nd international workshop on The role of abstraction in software engineering', ACM, New York, NY, USA, pp. 35–42. 28

Waguespack, L. and Schiano, W. (2004), 'Component-based is architecture', *Information Systems Management* **21**(3), 53–60. 13

Wang, X., Hu, S. X., Haq, E. and Garton, H. (2007), Integrating legacy systems within the service-oriented architecture, *in* 'IEEE Power Engineering Society General Meeting.', IEEE, pp. 1–7. 65, 66

Weinberg, G. and Freedman, D. (1984), 'Reviews, walkthroughs, and inspections', *IEEE Transactions on Software Engineering* **10**(1), 68–72. 49, 144, 152, 224, 225, 226

Wieringa, R. (1998), 'A survey of structured and object-oriented software specification methods and techniques', *ACM Computing Surveys* **30**(4), 459–527. 6, 31, 33, 39, 96, 137, 139, 186

Wong-Bushby, Egan, R. and Isaacson, C. (2006), A case study in SOA and re-architecture at company ABC, *in* '39th Annual Hawaii International Conference on System Sciences (HICSS '06)', Vol. 8, IEEE Computer Society, pp. 179b–179b. 65

Wood, M., Daly, J., Miller, J. and Roper, M. (1999), 'Multi-method research: an empirical investigation of object-oriented technology', *Journal of Systems and Software* **48**(1), 13–26. 41, 167, 171, 174

Yau, S. S. and Liu, J. (2007*a*), Functionality-based service matchmaking for service-oriented architecture, *in* '8th International Symposium on Autonomous Decentralized Systems (ISADS'07)', IEEE Computer Society, pp. 147–154. 65

Yau, S. S. and Liu, J. (2007*b*), A situation-aware access control based privacy-preserving service matchmaking approach for service-oriented architecture, *in* 'IEEE International Conference on Web Services (ICWS'07)', IEEE Computer Society, pp. 1056–1063. 65

Yin, R. K. (2008), *Case Study Research: Design and Methods*, Vol. 5, 4 edn, Sage Publications, Inc. 42, 46, 72, 77, 168, 169, 177, 190

Yourdon, E. (1989), *Structured walkthroughs*, 4 edn, Yourdon Press. 177

Yourdon, E. and Constantine, L. L. (1979), *Structured Design: Fundamentals of a Discipline of Computer Program and System Design*, Prentice-Hall. 97

Yue, P., Dia, L., Yanga, W., Yua, G. and Zhaoa, P. (2007), 'Semantics-based automatic composition of geospatial web service chains', *Computers and Geosciences* **33**(5), 649–665. 65

Zeid, A. and Elswidi, M. (2005), A peer-review based approach to teaching object-oriented framework development, *in* '18th Conference on Software Engineering Education and Training', pp. 51–58. 174

Zhang, J., Cheng, M., Chen, Z. and Fu, X. (2008), Pitch angle control for variable speed wind turbines, *in* 'Third International Conference on Electric Utility Deregulation and Restructuring and Power Technologies, 2008 (DRPT'08)', IEEE, pp. 2691–2696. xviii, 220

Zhang, T., Ying, S., Cao, S. and Jia, X. (2006*a*), A modeling framework for service-oriented architecture, *in* 'Proceedings of the Sixth International Conference on Quality Software (QSIC'06)', IEEE Computer Society. 57, 65

Zhang, T., Ying, S., Cao, S. and Jia, X. (2006*b*), A modeling framework for service-oriented architecture, *in* 'Sixth International Conference on Quality Software, 2006 (QSIC'06)', IEEE, pp. 219–226. 136

Zhang, X. and Gracanin, D. (2008), Streaming web services for 3D portal applications, *in* 'Web3D '08: Proceedings of the 13th international symposium on 3D web technology', ACM, pp. 23–26. 65

Zhao, H. and Tong, H. (2007), A dynamic service composition model based on constraints, *in* 'Sixth International Conference on Grid and Cooperative Computing (GCC'07)', IEEE Computer Society, pp. 659–662. 23

Zhu, F., Turner, M., Kotsiopoulos, I., Bennett, K., Russell, M., Budgen, D., Brereton, P., Keane, J., Layzell, P., Rigby, M. and Xu, J. (2004), Dynamic data integration using web services, *in* 'Proceedings of the IEEE International Conference on Web Services (ICWS'04)', IEEE Computer Society, pp. 262–269. 18

Zhu, X. and Zheng, X. (2005), A template-based approach for mass customization of service-oriented e-business applications, *in* '7th international conference on Electronic commerce (ICEC'05)', ACM, pp. 706–710. 65