Organic Aggregation
A Human-Centred and Model-Driven Approach to Engineering Service-Oriented Systems

THE AUSTRALIAN NATIONAL UNIVERSITY

A thesis submitted for the degree of
Doctor of Philosophy
of
The Australian National University

Yuanzhi Wang
October 2010
© Yuanzhi Wang 2010

This document was produced using \LaTeX, \LaTeXe and \BIBLaTeX
I declare that the work in this thesis is entirely my own and that to the best of my knowledge it does not contain any materials previously published or written by another person except where otherwise indicated.

Yuanzhizhi Wang
10 October 2010
To my wife and my parents for your love.
Acknowledgements

Many people have offered kind help and generous support during the journey of my PhD. I want to firstly thank my supervisors Chris Johnson and Kerry Taylor. They have given me the freedom and courage to fly in the sky that I feel comfortable with, while, in the meantime, saving me from the danger of going too far off road. I feel truly grateful for their efforts of training me as an independent and capable researcher, which is what this journey really means to me.

Chris has been such a great supervisor who knows the art of supervising someone like me, who has plenty of IT industry experience in the practical world, but is also passionate about various other things in the theoretical and philosophical world. He somehow knows whether the time has come to drag me back from the exploration of theoretical knowledge and the contemplations of exciting ideas, and to get me to touch down the ground of the practical world, or vice versa. Without him, I would not have started my PhD in the first place, and battled the unexpected hardships from both work and family halfway through, and get the thesis finished when the time has finally come. I am so very grateful for everything he has done for me. I am also in Kerry's debt, for her generous help and heartfelt support throughout my PhD. Despite her many other commitments, she has been such a trustworthy person who has given me a lot of support when it was needed most. It was her experienced eyes, guidance, encouragement, and trust that enabled me to finish this journey. Thank a lot for spending a lot of time on reviewing and commenting on my papers and thesis in great detail.

I want to thank Ramesh Sankaranarayana for being so supportive and thoughtful, and for his insightful suggestions and comments on my research and the thesis. His encouragement and kind concern means a lot to me. Special thanks to Shayne Flint, who inspired and influenced me a lot and guided me through the initial exploration of various approaches to service composition. Thanks for spending much time with me, and for opening the doors to the world of Checkland and Ackoff for me. I really appreciate it.I also want to thank Clive Boughton and Lynette Johns-Boast for their valuable advice, understanding, and kind encouragement.

Moreover, I would like to thank Ramesh, Lynette, Weifa Liang, and Trevor Vickers for training me to be a confident tutor and lecturer, and for giving me the opportunities to do lectures and tutorials, which is extremely valuable experience for me.

The Australian National University, the department of Computer Science, and CSIRO Water for a Healthy Country Flagship, generously provided scholarship
and funds for my PhD, which made this research possible. I am very grateful for their financial support. Special thanks to John Colton, Cecile Paris, and Dimitrios Georgakopoulos from CSIRO ICT Centre, who kindly offered me a full-time position in CSIRO as a Research Project Officer during my PhD. It was a great experience working with you in such a wonderful research environment. I really enjoyed it. Thanks for your support and understanding.

Going back to the university for a PhD after establishing a promising industry career was not an easy decision. It was made possible because of a lot of sacrifices from my families. I owe my wife so much for all the love, patience, and support she has given me. She has always been side-by-side with me, doing everything possibly in support of my work, even though she had to go through so much suffering and torture from unexpected sickness. Without her great love, my research and this thesis would never have been possible. Finally, I want to thank my parents, who have always supported and encouraged me, unconditionally. I would like to particularly thank my mother, who has encouraged me under all circumstances, even when she had to fight with the horrible illness herself. I feel guilty for the neglect and inability to help. All I can do is to make her proud of her son.
Abstract

Owing to a widespread trend of globalisation and service economies, there are exponentially increasing demands for Software-Intensive Systems (SIS) in general, and Service-Oriented Systems (SOS) in particular. However, it presents great challenges to develop and manage these systems. Although current research and practice provide various means to attack these challenges, there are many difficult impediments to overcome. This research is motivated by such demands, challenges, and opportunities. The ultimate objective is to understand and address the critical challenges of services engineering. To do so, we develop a multi-phased and iterative research methodology that is adapted from typical applied science research methodologies, in order to suit the exploratory nature of this research.

According to the research methodology, we investigate and analyse the special characteristics of services engineering, such as a high degree of complexity, uncertainty, and volatility. Moreover, some existing approaches and related work are studied and analysed in a critical way. We conclude that the great difficulties of services engineering are fundamentally caused by a lack of disciplined engineering approaches that take into account the rapidly co-evolving socio-technical environments, where both human intellectual capacities and engineering competence need to be well understood and exploited.

To realise our vision, we derive a generic engineering framework based on generalisation of other engineering disciplines, based on which, a services engineering framework called Organic Aggregation Services Engineering Framework (OASEF) is proposed. OASEF contains a theoretical foundation that consists of complementary theories and knowledge from multiple disciplines. Some important concepts are also defined, such as services engineering, models and modelling, and Socio-Technical Environments (STE). Moreover, OASEF contains some guiding principles that provide important guidance for the design and realisation of SOS and services engineering. Based on these conceptual resources, a profound concept called organic aggregation is developed, which takes an organic and synthetic approach to grow and manage systems of any kind.

Furthermore, OASEF also incorporates: 1) a generic conceptual process model called Organic Aggregation Process (OAP) in support of organic aggregations of human intellectual and technical capacities; 2) a fully integrated model-driven method to realise OASEF/OAP activities in a systematic and automatic way; 3) a range of domain-specific and general purpose modelling languages for OASEF activities; 4) a mechanism to capture and reuse engineering capacities and to realise automatic system generation; and 5) an integrated tool environment in
support of OASEF.

Two controlled proof-of-concept case studies are conducted in real world settings, which aim to evaluate and improve OASEF concepts, methods, and mechanisms. Results show that OASEF helps to manage system complexity, agility, and productivity when engineering SOS. Some limitations and insufficiencies are also observed, which require future research. Although this research mainly focuses on SOS and services engineering, its engineering framework, or more specifically, the theoretical foundation, guiding principles, and generic process model, can be applied within a wider scope of software engineering and systems engineering.
Contents

Acknowledgements v

Abstract vii

1 Introduction 1
  1.1 Background ........................................ 3
  1.2 Fundamental problems ................................ 8
  1.3 A demand for services engineering .................. 11
  1.4 Research methodology and activities ................ 13
  1.5 Thesis structure .................................... 23

2 Engineering SIS and SOS 27
  2.1 Concepts, terms, and definitions .................... 28
  2.2 Challenges and Characteristics of SOS ............... 46
  2.3 Current Approaches .................................. 48
  2.4 Our vision .......................................... 54
  2.5 Summary ............................................. 56

3 The engineering framework and theoretical foundation 57
  3.1 A generic engineering framework .................... 60
  3.2 A theoretical foundation ............................ 63
  3.3 Guiding principles .................................. 70
  3.4 Summary ............................................. 77

4 A New Conceptual Process Model 79
  4.1 A new process model of services engineering .......... 81
  4.2 A unified model of reality and mind at the fundamental level 87
  4.3 Concepts at the first level ........................... 92
  4.4 Concepts at the second level .......................... 97
  4.5 Concepts at the third level ........................... 105
  4.6 Summary ............................................. 111

5 The OASEF approach to services engineering 115
  5.1 The overall structure of OASEF ....................... 117
  5.2 Modelling languages ................................ 120
  5.3 OASEF Modelling .................................... 132
  5.4 Reusing engineering capacities ....................... 142
  5.5 Implementation and proof-of-concept tools support .... 146
### CONTENTS

5.6 Summary .................................................. 149

6 Case studies and evaluation ................................. 151
6.1 Background ................................................ 153
6.2 Managing complexity within business domains ......... 155
6.3 Linking conceptual models with technical artefacts ...... 163
6.4 Engineering capability encapsulation and automatic generation .... 165
6.5 Change and adaptation ................................... 171
6.6 Further analysis and discussion ......................... 176

7 Conclusion ................................................... 181
7.1 Summary .................................................... 183
7.2 Contributions ............................................... 189
7.3 Limitations and future work ............................... 191

Bibliography .................................................. 195
List of Figures

1.1 An engineering research process ................................................. 16
1.2 An adapted process for organising research activities. .................... 17
1.3 A research overview within a multi-phases methodology .................... 20

3.1 Human engineering activities ................................................... 60
3.2 A generic engineering framework .............................................. 62

4.1 Organic Aggregation Process (OAP) structure - Fundamental level ....... 91
4.2 Organic Aggregation Process (OAP) structure - Level 1 .................... 92
4.3 Organic Aggregation Process (OAP) structure - Level 2 .................... 98
4.4 Organic Aggregation Process (OAP) structure - Level 3 .................... 106

5.1 The overall structure of OASEF ................................................. 118
5.2 The meta-model of Correlation Network Model (CNM) ...................... 124
5.3 The graphic notations of Correlation Network Model (CNM) ................. 125
5.4 The meta-model of Problem Situation Exploration (PSE) ................... 127
5.5 The graphic notations of Problem Situation Model (PSM) .................. 128
5.6 The meta-model of Desired End Model (DEM) ................................ 129
5.7 The graphic notations of Desired End Model (DEM) ......................... 130

5.8 An example of CNM that captures general knowledge for improving online transactions - General Abstraction ...................................... 133
5.9 An example of UML class diagrams - System Abstraction .................. 135
5.10 An example of abstract model for an online travel booking web portal - Abstraction ............................................................. 136
5.11 An example of PSM in an OTB scenario - PSE ............................... 137
5.12 An example of DEM in an OTB scenario - DEE .............................. 139
5.13 Internal logic of capabilities required for closing an existing FHSA account - realisation ......................................................... 141
5.14 An example of using epitomes to realise flexible aggregation of OAP activities ................................................................. 144
5.15 An example of database schemas generated by an epitome ................ 145
5.16 A snapshot of IPEOAP tool support environment - PSE modelling .... 148
5.17 Examples of using epitomes to automate system generation - Generating Web applications ......................................................... 149

6.1 An example of a complex CNM that captures knowledge about customer behaviour - Managing complexity ......................... 157
6.2 A subordinate CNM that captures the influential factors related to customer satisfaction in OTB case study - Managing complexity . . . 158
6.3 A complex Abstraction model in OTB - Managing complexity . . . 160
6.4 Using UML activity diagram in realisation model in OTB case study - Realisation .................................................. 164
6.5 Using UML activity diagram to model realisation II - Realisation . . 166
6.6 Automatically generating BPEL using realisation models - Implement- 
mentation ....................................................................... 167
6.7 Using epitome to capture implementation capacities: “Information-
ServiceEpitome” that generates information service for a type of entities - Epitome ........................................ 169
6.8 The subordinate realisation model of generateService activity in epi-
tome “InformationServiceEpitome” - Epitome ............... 170
6.9 The automatically generated web service interface based on nor-
malised abstract models - implementation ...................... 171
6.10 An example to make changes to high order abstract models - abstrac-
tion .............................................................................. 172
6.11 The automatically generated user interface model based on the modified abstract models - realisation .......................... 173
6.12 The automatically generated web service interface based on modified abstract models - implementation .............. 174
Introduction

The central event of the twentieth century is the overthrow of matter. In technology, economics, and the politics of nations, wealth in the form of physical resources is steadily declining in value and significance. The power of mind are everywhere ascendant over the brute force of things.

George Gilder

Contents

1.1 Background .................................................. 3
  1.1.1 The socio-technical settings of modern SIS .............. 4
  1.1.2 The general challenges ..................................... 5
  1.1.3 Research scope ............................................ 6

1.2 Fundamental problems ......................................... 8
  1.2.1 Lack of engineering processes .............................. 8
  1.2.2 Inadequate focus on human perspectives .................. 9
  1.2.3 Insufficient reuse of intellectual capacities and competence . 11

1.3 A demand for services engineering .......................... 11

1.4 Research methodology and activities ....................... 13
  1.4.1 Multi-disciplinary approach ................................ 14
  1.4.2 Research method and process .............................. 15
  1.4.3 Multiple phases and research overviews .................. 18

1.5 Thesis structure ............................................. 23
In an information era, our world is more and more depending on computer software. In support of business and social interaction among individuals, organisations and societies, large amount of interconnecting software-enabled systems have emerged. Indeed, in the past several decades, the variety, scope, and size of modern Software-Intensive Systems (SIS) have increased by many orders of magnitude. These systems have become indispensable for modern society.

This trend is far from an end. The demand for more and better software and SIS is increasing exponentially. First, a widespread social and technological trend towards globalisation and service economies have driven enormous needs for more collaboration and interactions among people, communities, and systems [Aspray et al., 2006]. Second, advances in technologies, such as networking, Service-Oriented Computing (SOC), and Cloud Computing (CC), have enabled more sophisticated systems to be built and interconnected. These emergent technological innovation greatly ease production and integration of new and existing SIS. Third, fierce business competition and accelerating change of systems and their environments demand innovative creation and agile evolution of SIS to cope with the volatility [Truex et al., 1999]. Therefore, in such a socio-technical context, there is an ever-increasing demand for more flexible and dependable SIS [Boehm, 2006b].

However, it is extremely challenging, in practice, to satisfy such a critical demand. The reality is, when engineering and operating large-scale complex SIS, people often feel unsatisfied in terms of cost-effectiveness, productivity, dependability, and flexibility [Ulrich and Hayes, 1997, Boehm, 2006b]. This undesirable situation remains unresolved today despite continuous improvements of technologies and processes over half a century, such as more advanced programming languages, better integrated development environments, and methodology improvements [Shapiro, 1997].

This research is motivated by such a fundamental problem. The main research objectives are: 1) to understand open problems and emerging characteristics of SIS in today's rapidly evolving social and technological environments; 2) to form a sound theoretical foundation to guide and support engineering of SIS in modern settings; and 3) to explore specific and practical means to address the identified critical issues.

A classic applied science research methodology has been adopted throughout the research to incrementally explore viable directions and achieve the ultimate objectives. Moreover, although this is software engineering research, a multidisciplinary approach is believed to be important for achieving our vision. Our work therefore deliberately involves syntheses of knowledge and principles from
1.1 Background

various disciplines and areas, such as philosophy of human mind and behaviour, organisational management, software engineering and systems engineering.

The purpose of this chapter is to build a big picture of the research, including the socio-technical context, motivation, objectives, methodologies, and outcomes. The remainder of this chapter is organised as follows. Section 1.1 presents some important characteristics of modern socio-technical environment, followed by an analysis of general challenges such an environment presents. The scope of this research is clarified in such a context, namely software engineering in general, and services engineering in particular. Section 1.2 presents and analyses the fundamental problems within our research scope. These problems were identified during several iterations of research. The objective of our research is to make improvements for them. Section 1.3 analyses the specific demands for an emerging engineering discipline of services engineering, which motivates the research presented here. The ultimate research objective is also made clear here. Section 1.4 introduces the research methodology we use to arrange and conduct various activities, in order to systematically achieve our objectives. Specific research activities and the questions they are supposed to answer are clarified. This section also gives an overview of what has been done during each phase of the research. Finally, Section 1.5 provides an overview of the organisation and content of the remainder of this thesis.

1.1 Background

New technologies have had significant impact on human civilisation in human history. The most recent example is the significant technological transformation from the previous focus on raw material processing towards more powerful software-enabled information and knowledge processing since the second half of the twentieth century [Arthur, 1996]. This technological transformation has brought us forward to a digital era, which is characterised by the emergence and maturation of software and its enabling concepts and realisation infrastructure, such as computers, networks, World Wide Web, service-orientation, and more recently, the cloud services in cloud computing.

In the digital era, our ability to purposively influence and interact with the outside world has been largely enhanced by, and increasingly depends on, production and use of SIS. The production, operation, and evolution of these systems enable the human society to run as a whole, and in an efficient and flexible way. Software, SIS, and larger systems they collectively form, have become pervasive in modern society [Humphrey, 2001]. Almost all human products and services today are more
or less related to SIS, such as manufacture, telecommunication, transportation, finance, education, and health care services. Our everyday lives would be totally different without them.

Such a new phenomenon opens many interesting and critical research problems and opportunities. Understanding of the fundamental emerging characteristics from a broad socio-technical point of view is essential. The following section explains the reason.

1.1.1 The socio-technical settings of modern SIS

The emergence of SIS has accelerated profound social movements. Firstly, the world is flattened [Freidman, 2005]. The use of innovative technological platform such as computers, software, and the Internet, facilitates a widespread trend of globalisation at the level of countries, organisations, and more and more, individuals [Freidman, 2005]. Globalised economies and widened horizontal value chains require every element within the social structure to compete and collaborate in an open environment. Consequently, human business, social, and technological activities have to interrelate to each other within a much wider scope [Aspray et al., 2006]. The widened interconnectivity of systems is evidenced by the emergence of large numbers of large-scale complex SIS in the last few decades, especially large web-based systems, such as e-Health systems, online travel booking systems, and e-business systems.

Secondly, the world is accelerating. We observe things changing at a pace much faster than ever. This is because of not only the exponentially increased capacity and flexibility of hardware and software, but also the volatile business and social environment caused by competition and innovation [Boehm, 2006a]. Consequently, the rapidly changing world is forcing people to change the way we perceive, orient, and behave accordingly.

Therefore, in the rapidly changing and globally competitive environments, individuals and organisations have to compete with each other globally, and at the same time, cooperate with one another in linked value chains. People must strive for continuous improvements to obtain competitive advantages, in order to survive, and to make them feel satisfied. An example of such advantages is to obtain a competitive difference to win adversaries, and to secure a collaborative position in a global value chain [Peteraf, 1993]. Moreover, in the interrelated and dynamic environment of the 21st century, such improvements for advantages need to be obtained at a fast pace while competing and cooperating with the rest of the world, because the quantity of competitors increases, whereas the resources for which they compete remain relatively stable [Freidman, 2005]. As a result, the way
people do business must change, in order to accommodate the global connectivity, competition, and rapid change. This demand has been evidenced by the way rapid growing online business in the service sector transforming the traditional industry in the past decades [Wynn, 2000].

The understanding of such a broader social context of SIS is important for organisations and individuals. It leads to a view that the evolution of technologies should respect, reflect, and respond to the social change, in order to enable processes to adapt engineering practice in accordance with emerging challenges. Therefore, we need to adapt the way we produce, operate and evolve SIS, in their new socio-technical settings.

Such a socio-technical-environment-driven and change-focused view is crucial for people to survive and grow in this rapid changing world. This research is conducted based on such a broad view. It starts with recognition of fundamental problems of engineering SIS, which violate this view. The next section discusses these problems.

1.1.2 The general challenges

To adapt to and survive in the new world, we first need to understand where we are presently. It is also important to observe and analyse the characteristics of the new environments and the challenges they present. This section discusses these issues in general, and describes the scope of the research in terms of these pertinent issues.

Software engineering and systems engineering are relatively new engineering disciplines, which are concerned with production and operation of software and related systems. Despite their rapid growth, historically, building and managing dependable complex SIS has always been difficult [Pressman, 2007]. It has been commonly agreed that humankind is far from being satisfied by the state of the art software engineering and systems engineering practice [Royce, 1990, Gibbs, 1994, Ulrich and Hayes, 1997, Boehm, 2006b].

Firstly, people are unsatisfied with not only the cost and time required to engineer SIS, but also the quality or effectiveness of achieved ends. For example, the technological capacities and human resources often fail to satisfy the business and social needs as planned, which results in unexpected delay and budget overrun [Brooks, 1987, Johnson, 1996]. Moreover, existing engineering approaches often result in fatal software failures in safety-critical systems, which had serious negative impact on business and social integrity [Lyu, 1996]. Secondly, the engineering processes or methods to build and manage systems lag behind the increas-
ing demands. For example, traditional engineering approaches cannot satisfy the needs for service-orientation in real world enterprise environments [Papazoglou et al., 2007]. As pointed out by Papazoglou et al., "an evolutionary software engineering approach" is essential to realise the full vision of service technologies, because of the insufficiency and inability of traditional approaches to building Service-Oriented Systems (SOS). In summary, the efficiency and quality of modern SIS and the engineering means to derive them are inadequate. Despite advances in technologies and engineering capacities over the past decades, the long-lasting inability to satisfy user demands still remains.

We believe this unpleasant situation is fundamentally caused by the co-evolving social and technological environments, where both user needs and engineering capabilities evolve over time. In other words, it is the emerging characteristics of today's large-scale, complex, uncertain, and dynamic environments that makes it so challenging to derive both effective means and ends pertinent to SIS. Firstly, system complexity has always challenged the limit of human cognitive and engineering capacities [MacCormack et al., 2001, Baresi, 2006]. Continuous innovation of methods and tools are necessary to assist humankind to conquer such a limitation. Secondly, in dynamic environments, it is difficult to balance the needs of addressing rapid change and maintaining desired dependability at the same time [Boehm, 2006b]. Trade-off between agility and dependability is needed, since the achievement of both is often impossible. Lastly, modern environments present a higher degree of uncertainty and emergence, which has not been paid enough attention to, or addressed effectively in the past [Boehm, 2008]. On the one hand, the emergent uncertainty and unpredictable variability cannot be managed using current rigid plan-driven approaches. On the other hand, there is insufficient methodological support for agile methods to ensure predictable dependability. Therefore, there is a lack of methodological support in today's social-technical settings.

1.1.3 Research scope

The need for recognising and addressing the critical challenges discussed in the previous section has become prominent in software engineering and systems engineering [Boehm, 2008], whose environments show an increasing degree of dynamics, uncertainty, and complexity. Such a need is especially critical for some rapidly emerging engineering areas, such as SOS and services engineering.

SOS are an emerging form of modern SIS. They incorporate a concept of service orientation from both a social and a technological point of view. Socially, these systems are organised and interconnected based on specialisation, provision and
1.1 Background

corruption of services in a value chain [Vargo et al., 2008]. Technologically, they are realised using concepts, technologies, and infrastructure from SOC, and more specifically, Service-Oriented Architecture (SOA). The former is a new computing paradigm that allows SIS to be engineered using autonomous and loosely-coupled services [Papazoglou et al., 2008]. The latter is a specific technological architecture for implementing SIS based on a set of widely-supported standards, protocols, and infrastructure [Erl, 2008]. Over years of practice and evolution, SOS have become an important, and increasingly necessary, part of modern SIS.

Accordingly, a specific branch of the engineering discipline, namely, services engineering, has emerged and grown rapidly in recent years. It is concerned with engineering methodologies, principles, and practice to develop and manage SOS. There are a number of definitions of services engineering from different perspectives [Papazoglou et al., 2008, Amyot et al., 2008]. In this thesis, we define services engineering as an engineering discipline that applies pertinent engineering knowledge, resources, and experience to derive, operate, and alter SOS in a socio-technical environment. This definition reflects our vision of SOS which involve both social and technological aspects, while maintaining the focus on engineering. The nature and meaning of SOS and services engineering will be clarified and defined in Section 2.1.2.4 of chapter 2.

The main scope of this research is therefore in the context of software engineering and systems engineering in general, and services engineering in particular. SOS inherently involve more complex, heterogeneous, and agile environments, due to the highly competitive nature of a service-intensive society and rapidly evolving SOC technical infrastructure. The openness nature of such environments is starting to become a norm for modern SIS [Ghezzi, 2007]. That is, more and more SIS would present similar emergent characteristics as SOS, in the socio-technical settings described in Section 1.1.1. Therefore, in a sense, SOS and services engineering provide a good case study for understanding and improving SIS in modern environments. It is therefore necessary to observe and analyse the emergent characteristics of such environments, and to derive and adapt engineering approaches to deal with them accordingly. The observation, understanding, improvements, and lessons obtained in services engineering could be generalised to cope with general engineering problems, in a broader scope of software engineering and systems engineering.
1.2 Fundamental problems

As discussed in the previous section, there is a critical need to analyse and understand fundamental problems of SIS in modern settings. This section investigates and discusses some important problems and issues in detail, in the scope of services engineering and SOS in particular.

1.2.1 Lack of engineering processes

Although various approaches are used in industry to build and manage SOS from both business and IT perspectives, these traditional methods show deficiencies during enterprise adoption of SOC [Chen et al., 2010].

Both engineering and management issues are involved in this undesirable situation. On the engineering side, there is a lack of suitable processes to develop and maintain dependable SOS in an efficient and agile way [Papazoglou et al., 2007]. Enterprises, in practice, either directly adopt traditional software engineering and systems engineering processes, such as structural or incremental processes, or rely solely on advanced technologies and implementation solutions without explicit or rigorous processes.

A lack of appropriate engineering processes has been a major impediment for successful SOS in today's complex and volatile environments. For example, because service environments are dynamic and volatile, business demands and capabilities cannot be derived and managed systematically and efficiently, without repeatable and justifiable processes, practical guidelines and practice, and appropriate supporting tools. Simply being agile without justification and predictable ends is not going to ensure that the right engineering decisions are made in response to rapid change. Moreover, a technological focus without methodological support is problematic when system complexity increases. For example, using implementation-centric approaches such as Business Process Execution Language (BPEL) design, it becomes tedious and error-prone to develop and maintain complex SOS that present a high degree of semantic diversity [Ouyang et al., 2006].

On the business side, understanding and capturing business needs and demands of SOS in a broad environment has not been effectively supported. As emphasised by many researchers [Sommerville, 1998, Kotonya and Sommerville, 1998, Boehm, 2006a], effective requirements engineering should take both technological and social aspects into consideration, and needs to be incorporated into the overall engineering process in a unified fashion. In the context of services engineering, although this is supported by an emphasis on business process
modelling, stakeholder and commitment modelling, and service modelling, these high level modelling processes are often conducted separately, and in isolation from service design and provision [Papazoglou et al., 2006].

There is a lack of integrated processes that align volatile business with co-evolving technologies. This problem exists in many SIS and badly affect the system effectiveness [Sauer and Yetton, 1997]. In fact, the isolation and mismatch among business and technological strategies is widely recognised as a chronic business-IT alignment problem [Tallon, 2003]. The consequence of this problem is unsatisfied business needs and waste of technological resources.

Ironically, although one of the objectives of SOC is to solve this problem, there is no holistic process in services engineering that unifies business and technology. Specifically, there is no process that integrates higher level business mission, objectives, and plans, with lower level technology architecture, design, and capacities, and creates shared understanding and vision among various stakeholders [Reich and Benbasat, 1996]. Instead, fundamental business strategies and volatile operational business goals are derived and captured in isolation from technological provision of service. Existing efforts to bring them together have not broken the barrier successfully [Ouyang et al., 2006].

Therefore, we consider a lack of suitable processes for engineering and managing SOS a fundamental problem in modern socio-technical settings. Without appropriate processes to put both technical capacities and user needs into a unified system as a whole, this problem would remain unresolved in services engineering, which, unsurprisingly, leads to useless, unusable, or unfit systems.

1.2.2 Inadequate focus on human perspectives

As discussed in Section 1.1.1, in today's socio-technical settings, individuals and organisations have to develop a global vision in order to survive and prosper. In addition to a focus on technologies and business domains, such a global vision should also incorporate another important dimension, human beings.

In general, there is a lack of emphasis on human aspects in traditional software engineering, arguably because of its initial focus on natural science and technologies in the early history of software engineering. For example, a typical software engineering process starts with pre-specified system specifications using techniques such as use cases, without taking into account the user's overall actual settings [Sommerville, 2004]. As Sommerville points out, a lack of understanding of overall user environment, such as the multiple-task environment, and the complex social and organisational settings, is often problematic in large-scale
complex systems. Similarly, the rest of software engineering activities barely respect the notion of user in a real sense.

In recent years, there is an increasing focus on user aspects in SOS, such as more emphasis on system usability and user value, in addition to product functionality and cost [Boehm, 2006a, Checkland, 1981]. There are also other areas involving intensive human and social aspects, such as requirements engineering, stakeholder and role analysis, and human-machine interaction. Moreover, recent agile methodologies also promote a focus on collaboration and communication among various stakeholders, although often being ineffective in large projects [Sommerville, 2004].

However, this is still a largely neglected area in software engineering, that is, understanding of the nature, existence, and supporting environment of human intellectual capacities during engineering processes. What we mean by human intellectual capacity here is the cognitive and purposive efforts humankind made for the purpose of solving practical engineering problems. They include the capabilities to purposefully collect pertinent resources, observe and make sense of the outside world, and to reason about relevant situations, or to make appropriate plans for physical response. Such capacities are so crucial for practical and creative problem-solving for all engineering purposes. They are the fuel of human engineering achievements and innovation. As such, this human perspective deserves through understanding and support in an engineering context.

Despite its critical position in human engineering problem-solving, there is no comprehensive research in software engineering and systems engineering related to human intellectual capacities. For example, we have very limited understanding of how human beings observe and understand the business environment, which pattern or practice would be more efficient than others when understanding and eliciting system specifications, or what is the cognitive processes involved when deriving engineering decisions with confidence, before actually implementing them. In other words, we have very limited understanding of human intellectual efforts during the entire engineering processes.

A reflection on this situation raises a significant issue. If we don’t understand the human internal process to do engineering, either as individuals or as groups, how could we control it in a good way, how could we effectively improve it? Pragmatic means might shed light on one way to improve engineering by adding feedback from downstream. However, without knowing and understanding the internal intellectual process, improvements of human engineering processes based on empirical feedback would be more exploratory and less efficient.
1.2.3 Insufficient reuse of intellectual capacities and competence

As discussed in the previous section, human intellectual knowledge and engineering capacities are the essential resources that human beings possess for attacking various engineering problems. It makes great sense to be able to capture and utilise the best of them for future challenges. However, presently, these critical resources are not systematically captured, and cannot be effectively reused in reality [Atkinson and Bostan, 2009].

It is true that various great innovation in technology have attempted to alleviate this issue. For instance, the emergence of the World Wide Web facilitates general knowledge reuse. Some mechanisms have proven to be greatly useful for sharing and reusing intellectual knowledge, such as good search engines, knowledge bases and Wikis. Service orientation in SOC also makes it possible to wrap knowledge and capabilities and to publish standard interfaces to the public, which can be later discovered and consumed by interested users.

However, in an engineering context, these forms of knowledge cannot be easily re-accessed and reused in an efficient and systematic way. For example, specific knowledge belonging to various stakeholders is captured either in people's minds as common sense, expertise and experience, or in various concrete forms such as documents, emails, web pages, regulations and laws. The nature and usage requirements of these forms of capabilities vary. As Gil et al. [2007] pointed out, it is 'impractical or even impossible' to reproduce and share information and processes within a complex environment where fragmented information exists in various forms and locations.

Therefore, in an engineering context, there is presently no holistic means to effectively represent, manage and reuse engineering knowledge and capabilities as intellectual assets. We believe improvements in this regard would be important for engineering SOS in modern settings, because important engineering capacity reuse contributes to balancing the trade-off between agility and dependability. Detailed discussion on this issue will be presented in chapter 3.

1.3 A demand for services engineering

Owing to the social and technological movements in the 21st century described in Section 1.1.1, there are exponentially increasing demands for more and better SIS in general, and SOS in particular. First, the widespread trend of globalisation requires more new systems to be built and interconnected in a broader scope [Freidman, 2005]. The process of globalisation increases the variety and
Chapter 1: Introduction

scope of new systems. Secondly, the accelerating socio-technical change demand responsive adaptation and evolution of existing systems [MacCormack et al., 2001]. The world is rapidly changing. Legacy systems could become unfit in the new environments. Traditional techniques and methods could become obsolete. Therefore, existing systems need to be improved accordingly in terms of agility, usability, and other quality attributes, in order to adapt to the changing world.

To satisfy such a critical demand, it is crucial to use disciplined engineering means. It is especially true for large-scale complex systems. For this type of systems, disciplined means are necessary to ensure effective and efficient production and management while satisfying the need for flexibility and dependability [Jones, 2003]. This principle also applies to SOS. A lack of disciplined processes can lead to either unsatisfied SOS, or unaccomplished engineering goals in terms of cost-benefit and productivity [Papazoglou and van den Heuvel, 2006]. Therefore, as the size and interaction of SOS continuously expand, the need for disciplined engineering processes and methodologies becomes critical.

Traditional software development approaches, however, are insufficient. They are mainly based on a closed-world assumption that treats the boundary between systems and their environments in a rigid, well-defined, and unchanging fashion [Baresi, 2006]. These approaches often fail to work effectively in today’s open-world environments. This is because they were designed for less open environments, and provide insufficient support for addressing new challenges of rapid, and often unpredictable, change in more dynamic and heterogeneous environments [Ghezzi, 2007]. When environments become highly complex and dynamic, these approaches cannot manage the complexity and volatility, and are unable to produce effective, flexible, and robust systems [Pastor and Molina, 2007]. More specifically, the traditional engineering approaches suffer the fundamental problems described in Section 1.2. For example, the approaches that focus on technologies and implementation detail from the solution providers’ perspectives, often lack thorough understanding of the dynamically changing user demands and essential complexities from a broader stakeholders’ perspective [Bennett et al., 2000]. They therefore often fail to understand and provide support for needs of all stakeholders.

Therefore, due to emerging characteristics of today’s environments, we cannot blindly adopt the existing engineering approaches. Instead, existing engineering disciplines need to be adapted in response to the new challenges. This applies to SOS too. A new discipline of services engineering is required to engineer various business and social systems, in support of the rapidly growing global service economies [Cardoso et al., 2008].
1.4 Research methodology and activities

That is, we cannot simply apply existing software engineering approaches in a service-oriented context, without sufficiently accommodating the nature and characteristics of services and their environments. Instead, there is a need for recognition and understanding of the problems and challenges involved in SOS, either old or emerging ones. Practical solutions are then needed to improve or address these identified issues. To do so, services engineering needs to involve much wider scope that goes beyond SOS. A wide variety of aspects, such as characteristics of SOS, the socio-technical environments in which they reside, and the engineering means to address them, should be considered and addressed. For a new, but rapidly growing engineering discipline, we need to effectively understand and address these issues, potentially based on different philosophies, views, and means, if necessary.

Moreover, it presents great opportunity for innovative techniques and methodologies, in response to the fundamental demand for more and better SOS. Such an opportunity, especially in services engineering, could provide an important means for understanding the general problems and nature of general SIS in modern environments, and for deriving and generalising effective solutions for improving and resolving them.

This research is motivated by such a critical demand and great opportunity. The ultimate objective is to understand and tackle the grand challenges when engineering SOS in particular, and modern SIS in general, and to provide a viable engineering solution to improve the critical real world problems. Research towards this objective would contribute to the emergence of a new engineering discipline of services engineering. It is also expected to contribute to advancement of the state of the art of software engineering in the new socio-technical settings of the 21st century. Simply put it, we aim to adapt traditional engineering principles and methods to form a new approach to engineering modern SIS in general, and SOS in particular.

In order to achieve our overall objective, we use a specific applied science research methodology, which is a creative synthesis of some typical research methodologies tailored for our purposes. Systematic methods help to organise research activities to gradually achieve the objective. The next section presents such a methodology.

1.4 Research methodology and activities

In the previous sections, we discussed the socio-technical environment of modern SIS and SOS, and a critical demand for mature engineering discipline to address
some fundamental problems in such environment. This, however, is not an easy

task. The fundamental problems and great challenges we are dealing with are
complex and difficult in nature. There are also unknown, or less-understood
impediments that create uncertainties. A sound research methodology is therefore
important to organise and guide research activities to systematically and purpose-
fully conquer these difficulties and achieve our objectives.

1.4.1 Multi-disciplinary approach

It is important to note that this research applies a multi-disciplinary approach. As
discussed in Section 1.1.1, the transition towards services and globalisation involve
a wide range of aspects, such as technologies, people, economics, and politics.
Therefore, the inherent complexity of SOS in such a context cannot be understood
and managed within a particular subject domain or discipline. Instead, a wider
inter-disciplinary approach is necessary to address the engineering challenges.

Therefore, in the scope of this research, it is particularly important to
strengthen, rather than weaken, synthescs of various knowledge, techniques, tools
and processes from pertinent disciplines. Due to the people-intensive nature of
services, a services engineering methodology cannot be effective without taking
human aspects seriously. The problem of a lack of human perspective has been
discussed in Section 1.2.2. What it means is that the emerging services engineering
cannot merely focus on scientific knowledge and engineering practice. It also needs
to apply knowledge of social sciences that helps to address important issues related
to human beings.

To be effective, a multi-disciplinary approach is necessary for achieving our
overall objectives. In this research, we are influenced by knowledge and wisdom
of related disciplines during the exploration process, especially knowledge of
social sciences in support of our vision of incorporating human perspectives into
engineering. In particular, we strive for bridging the ever-lasting gap between
engineering and social sciences, by acknowledging and supporting some human
perspectives in the context of engineering. As a result, deliberately, in many parts
of this thesis, our work does not look like traditional engineering research. It
coherently incorporates some knowledge from philosophy, psychology, economics,
and management, to serve the purpose of this research, while staying within our
scope of services engineering and software engineering.
1.4.2 Research method and process

To deal with the difficult problems confronting us, we adapt a typical engineering research method generalised by Adrion [1993]. Figure 1.1 depicts the process and activities it involves. As see in the figure, it consists of a series of interconnected activities conducted in a sequence. “Analyse challenges and problems” activity is concerned with identifying and prioritising problems to be addressed. “Observe existing solutions” activity analyses current solutions in terms of advantages and shortcomings. “Propose better solutions” activity attempts to introduce different or improved ways to solve the problems. “Build or develop” activity involve realisation of the new solution in the real world settings, which is measured and analysed during “Measure and analyse” to obtain feedback and judgement in terms of its pragmatic effectiveness. “Repeat until no further improvements are possible” adds a feedback loop to the process for continuous improvements when necessary.

Due to the significance and difficulty of the problems and challenges we are dealing with, we also adapt this methodology in our context to suit specific needs. Figure 1.2 depicts this. Specifically, an activity called “Form theoretical foundation” is introduced to derive a sound conceptual foundation for following analysis and experiment activities. This activity is considered important, because that a solid foundation provides intrinsic support for critical research actions, such as exploration and determination of research directions, alternative approaches, priorities, and depth of investigation. Due to constraints on time and the high degree of uncertainty involved in the research, it is impossible to guarantee that the objective will be met beforehand. However, a good foundation ensures that the research starts from the right place, and aiming at the right direction in general. Efforts made in a solid theoretical foundation will be paid off in later stages because of the important guidance and logical soundness it provides.

Moreover, in addition to the “Form theoretical foundation”, we also emphasise gathering of the practical experience obtained from proven solutions. The idea here is not to build a better solution from scratch. “Observe existing solutions” should also generalise existing good practice, which can be reused in the new solution. In other words, our search for effective services engineering should be built on top of the generalised engineering framework that has been practised for hundreds of years.

This is the rationale of an activity called “Capture and generalise good practice”. In our scope, this activity involves analysis and generalisation of existing engineering frameworks, on top of which our approach of services engineering is designed and built. Here, by framework, we mean a coherent conceptual structure that arranges elements and resources that are required for engineering
Figure 1.1: An engineering research process

systems. Looking for such an engineering framework helps us to identify important resources for engineering and to understand the relationships among them.

To assist conducting the research activities discussed in the previous section, we explicitly identify a number of specific research questions, in accordance with the purpose of these activities. These questions helped us to stay focused on the main purpose of each activity within our scope, and to systematically analyse the effectiveness of outcomes.

I What are the nature, special characteristics, and important challenges of SOS and services engineering?

II What is nature of the existing solutions for engineering SOS?

III What body of knowledge, schools of thought, and good practice can be coher-
1.4 Research methodology and activities

Figure 1.2: An adapted process for organising research activities.

iv) What framework or pattern should services engineering be based on?

v) What conceptual and practical mechanism do we use to address each identified issue?

vi) How do these mechanisms perform in real situations? How do we assess them?

vii) Are improvements needed? If so, what are they?
The identified research questions correspond to the purpose or objective of individual research activities in our scope. It also helps us to assess the outcomes of the research at various stages. However, due to the complex nature and inter-dependency of these questions, it is difficult to organise research activities in a one-off fashion, and expect that, in the end of each sequential activity, we will be able to tick one question as mission accomplished. Instead, multiple phases were introduced to help exploration of potential answers to these questions, in other words, to help achieve our main objectives. This arrangement will be presented in the following section, which also provides an overview of what and how this research has been conducted in this structure.

1.4.3 Multiple phases and research overviews

Some questions described in the previous section are relatively easy to answer, such as question II. Some, however, are more difficult, due to the exploratory nature of the research, such as questions I, III, and V. This is because of two reasons. First, for a complex problem situation, it is often impossible to understand its nature and characteristics just by observation alone. Continuous interactions is often necessary to obtain incremental understanding. Second, investigating, choosing, and synthesising a relevant body of knowledge from the enormous source of human intelligence is requires significant time and efforts. Without a clear focus and explicit goal to be achieved, the theoretical exploration would have lost its direction in the “forest” of knowledge, and produced less value in support of practical solutions within the constrained time-frame. Third, conceiving, realising, and trying one particular mechanism to address a particular problem, out of many options, is often non-trivial, depending on the degree of complexity and difficulty of the problem. Therefore, there are many uncertainties involved while attempting to answer these questions.

For these difficult questions, exploratory and iterative approaches are most effective [Ackoff, 1999]. We therefore synthesise some aspects of Ackoff’s methodology to allow for iterative and tentative exploration of potential answers for difficult questions. This method is in good alignment with the one depicted in Figure 1.2. It includes various phases to iteratively observe and analyse the problems and their related environments, to explore and understand various options of tackling them, to design and develop possible methods that attempt to solve the problems, to evaluate the methods, re-orientate and re-learn, and to synthesise and generalise a general methodology [Ackoff, 1999]. Since gradually and continuously understanding problems and trying various solutions is the best way to address unknown and uncertain situations, it makes sense to incorporate
1.4 Research methodology and activities

iterative activities that explore different possible solutions. Therefore, the iterative and exploration aspects suit the nature of this research.

Therefore, the research activities that we conducted to achieve our visions and objectives are arranged into several iterative phases, varying in terms of size, scope, and focus. The phased arrangement helps to manage the goals, content, and schedule in our uncertain research environment [Flint, 2006]. Instead of having a big chuck of sequential activities, we designed an iterative and multi-phased research process, which consists of a number of incremental and fine-grained phases. Each iteration aims to obtain more information and knowledge about the problems to be deal with and the effectiveness of a particular potential solution.

Figure 1.3 illustrates such a multi-phased research process by showing the path that we followed during the research. There are four phases involved, namely Touching, Exploring, Focusing, and Reflecting.

Firstly, Touching is the initial phase to understand the problems and to gather relevant information about the situation and potential solutions. The main activities involve Analyse challenges and problems, Observe existing solutions, and Form theoretical foundation. We started with some initial investigation and experiment about the field under study. During Analyse challenges and problems, literature reviews on theories and practice about service, SOA, SOC, and Model-Driven engineering (MDE) were conducted to shed light on how SOS are presently engineered. Some critical problems were gathered from the literature based on reviews and analysis. Form theoretical foundation involves reflection on these problems and exploration of theoretical foundation such as systems theory. During Observe existing solutions, some experiments were conducted to try some existing solutions and potential techniques such as Web services and BPEL. We also did some experiments to get familiar with technical detail of MDE, such as modelling and model transformation techniques. In particular, we investigated eXecutable and Translatable UML (xUML) approach to build systems. These technical exercises helped to build better understanding of the strength and limitations of existing approaches of MDE. Collectively, the conducted experiments and exercises shaped concrete understanding of what these techniques are about, how can they be used to build systems, and how do they perform in practice.

Secondly, Exploring involves all research activities illustrated in Figure 1.2, which are related to exploration of new solutions from a range of potential options. These activities include Capture and generalise good practice, Propose better solutions, Build or develop, and Measure and analyse. The main objective of this phase is to identify and try various potential solutions to improve the current situation. Better understanding of the problems, challenges, and the
pitfalls can also be obtained through the experimental exploration. Change to theoretical foundation is also made to support following research activities. Specifically, a novel Aspect-Oriented Thinking (AOT) approach [Flint, 2006] was investigated to assist service composition. Due to its good alignment with the objectives of this research, a tentative solution was designed and experimented using a case study within the context of Water Resources Observation Network.
1.4 Research methodology and activities

(WRON) project [Wang and Taylor, 2008]. Some important insights and valuable experience were obtained during the experimental case study. Some other potential techniques and solutions were also investigated.

Thirdly, Focusing is a phase to assess the result of Exploring, and to make decision with regards to the strategic research direction and the specific forthcoming research activities, given available time and resource. The main task is to develop a strong vision of where the research is heading, a strong focus on particular way to achieve such a vision, and to provide the best possible solution for the fundamental problems according to outcomes of Exploring. This is possible in this phase due to deeper understanding of problem situations, richer theoretical knowledge, more experience and generalised knowledge from attempted solutions. The Focusing phase also involves all research activities as illustrated in Figure 1.2. But the depth and intensity of each activity is greater than other phases, because in this phase, a strong vision, mission, and focus have been developed. Focusing is different from Exploring, in the sense that the former has a clear and viable vision and specific focus, whereas the latter involves a lot of unknown and uncertain factors.

Specifically, the experience and knowledge obtained from the previous phases resulted in a process-driven, multi-disciplinary, and synthetic vision. We hold a system world view that takes a synthetic, as opposed to reductive, approach to growing and managing processes, knowledge, capacities, and systems, via both sensible and agile aggregations of important engineering resources.

Such a vision was continuously realised, assessed, and improved later in this phase. It is believed to address some critical challenges and fundamental problems identified from the previous phases, which are described in Section 1.2 and 1.1.2. In Form theoretical foundation, a coherent body of knowledge and theories are synthesised to guide the proposed solution to address the identified problems and challenges. Some fundamental theories and knowledge are coherently synthesised as a theoretical foundation, such as systems theory and theories about human mind and behaviour. As discussed in Chapter 3, such a foundation also incorporates some important guiding principles, which will be used to direct the design and realisation of the new solution.

Moreover, during Capture and generalise good practice, a typical engineering framework was generalised. This framework forms the basis for a new solution for services engineering. It is called Organic Aggregation Service Engineering Framework (OASEF) [Wang, 2010b], as result from Propose better solutions. It consists of a novel general process model called Organic Aggregation Process (OAP) [Wang, 2010a], which fully integrates a model-driven method and some spe-
cific mechanisms. A prototype of the new approach was realised and implemented during Build or develop, using some widely supported open source platforms, such as Eclipse Modeling Framework (EMF) and Apache Axis. Some supporting tools and techniques were implemented in an integrated development environment to facilitate the applications of proposed concepts, design, and ideas.

During Measure and analyse, the proposed solution is assessed in two proof-of-concept case studies, as opposed to proof-of-performance ones [Snyder, 1994]. These controlled case studies were designed in real-world settings to reflect the potential practical problems. The case studies aim to assess whether, in general, the research objectives are met when the proposed approach is used to build SOS. The first case study is in the context of an online travel booking business, which will be refereed to as Online Travel Booking (OTB) in the remainder of the thesis. It is a typical scenario used in the service computing community. The second case study is based on the Australian First Home Saver Accounts (FHSA) scheme that was introduced by the federal government in 2008 to help residents to purchase their first homes [Wang, 2010a]. The empirical experience from the case studies was used to assess design features and their effectiveness, and to identify strength and limitations of the proposed solution.

Fourthly, Reflecting phase aims to assess the latest states of the research after Focusing activities. Limitations, problems, and necessary improvements are to be identified. The overall research processes are iterated partially or wholly until the overall objective of the research is achieved. Another objective is to generalise widely applicable methodology by a process that Ackoff[1999] described as re-orientation and re-learning. Although the OASEF solution, particularly the OAP process, was designed within a broad scope of software engineering and systems engineering, it mainly focused on, and was tested in, services engineering in particular. The applicability of the proposed solution when engineering general SIS in modern settings will need to be assessed and verified. We made critical improvements to the theoretical foundation and general OASEF design according to feedback from peer researchers and empirical observation, and generalised the existing notions and mechanisms in a broader scope of Software-Intensive Systems Engineering (SISe). Due to constraints on time and resources, the verification of the new solution's effectiveness in broader software engineering and systems engineering will have to be conducted as future research. Empirical case studies in real world projects are necessary to achieve this goal.
1.5 Thesis structure

The remainder of the thesis is organised as follows.

Chapter 2 presents our study and analysis of research questions I and II (See Section 1.4.2). That is: I) What are the nature, special characteristics, and important challenges of SOS and services engineering? II) What is the nature of existing solutions for engineering SOS? Specifically, Section 2.1 clarifies and defines some important notions and concepts, such as SIS and SISE, services, SOS, services engineering, models, MDE, and socio-technical and human perspectives. These concepts and terms are referred to throughout this thesis. Section 2.2 presents an overview of the problems and challenges of SOS and services engineering. Section 2.3 presents the state of the art solutions to tackle these challenges. Some related work is reviewed and analysed, especially model-driven approaches to services engineering. In Section 2.4, a different vision is presented to address these challenges in the context of service engineering and modern SISE. It requires a radical paradigm shift from a closed-world view to an open-world one. In particular, it is necessary to explore important human intellectual capacities and specific means to capture and re-use existing capacities at different level of abstraction, such as cognitive activities and technical competence.

Chapter 3 aims to provide answers to research question III and IV (See Section 1.4.2). That is: III) What body of knowledge, or schools of thought, can be coherently used to form a sound theoretical foundation for services engineering, in support of addressing the identified challenges? IV) What framework or pattern should the services engineering be based on? In this chapter, we will present a coherent theoretical foundation for our new approach to services engineering. It starts with description of a generic engineering framework we developed to structure and organise our approach to services engineering, which will be presented in Section 3.1. This framework is formed based on generalisation for other engineering disciplines. Our approach to service engineering, called Organic Aggregation Service Engineering Framework (OASEF), is based on such a generic engineering framework. It has an important theoretical foundation that consists of complementary theories and knowledge from multiple disciplines, such as systems theory and the philosophy of mind. Section 3.2 briefly presents some important parts of this foundation. In Section 3.3, we will present some important guiding principles, which are in good alignment with the theoretical foundation. These principles are derived from either empirically verified engineering practices, or logical deduction from the theoretical foundation. They guide the design and realisation of our approach to tackling the identified challenges of services engineering, either explicitly, or implicitly.
Chapter 4 aims to answer the first part of research question V, that is, *What conceptual mechanism do we use to address each identified issue?* It further instantiates the proposed services engineering framework with an engineering process. Such a process is designed as a general conceptual model for addressing the challenges of open environments of modern SIS and SOS, such as a high degree of complexity and volatility. Section 4.1 analyses the demand for a new process model and our vision for such a model. In support of such a vision, we developed a fundamental concept called *organic aggregation*, which is introduced in Section 4.1.2. Based on such a concept and the theoretical foundation, we establish a conceptual model to organise engineering activities in a hierarchy. Section 4.2 defines our notions of *mind* and *reality* at the fundamental level of OAP. The internal structure of OAP is arranged in the hierarchy that link the world of human *mind* with *reality*. Section 4.3, 4.4, and 4.5 introduce the interconnected inner processes or activities of OAP at more detailed level. The concept, notions, and mechanisms pertinent to these activities are developed, defined, and elaborated. Such a conceptual process aims to provide support for engineering SIS in general, and SOS in particular.

Chapter 5 aims to address research question V: *What practical mechanisms do we use for effective services engineering?* Specifically, this chapter presents the complete picture of OASEF, by adding the missing parts to the framework. For example, it tailors the OAP process in the context of services engineering. A model-driven methods and modelling techniques for various OASEF inner-process and activities are also introduced in this chapter. A mechanism to capture and reuse engineering capacities are also introduced in this chapter. Some examples from the case studies are used here to illustrate involved concepts and mechanisms. This chapter also briefly describes the integrated supporting tool environment we developed. Specifically, this chapter is organised as follows: Chapter 5.1 describes the overall structure of OASEF that consists of a foundation, various guiding principles, an integrated OAP model, a model-driven method and various modelling mechanisms, and a supporting tool environments. Section 5.2 presents the modelling languages, meta-models, and graphical notations of various important OASEF models, especially models of higher order human intellectual effort such as *abstraction* and *rationalisation*. Section 5.3 illustrates the specific modelling methods and techniques with some simple examples from the case studies. Section 5.4 describes the mechanism to reuse intellectual engineering capacities, which is called *epitome* and *epitomisation*. Section 5.5 introduces the *implementation* processes that integrate the *epitome* with a specific model transformation technique to produce concrete systems based on engineering capacities and high order OASEF models. This section also introduces an implemented prototype tool environment, which provides a range of integrated supporting tools in support.
of OASEF engineering activities.

Chapter 6 presents our effort to answer research question VI: how does the proposed approach perform in real situations? It presents two case studies we conducted for the purpose of obtaining proof-of-concept, as opposed to proof-of-performance, evaluation of OASEF in real world settings. Business scenarios are designed in the controlled case studies according to the problems and research questions presented in Section 1.4.2 and 1.2. Specifically, this chapter is organised as follows. Section 6.1 provides background information about these case studies. Section 6.2 analyses how OASEF manages the challenge of complexity, in the context of a complex online travel business scenario. Section 6.3 presents how to utilise high level OASEF models from perception and rationalisation processes to design conceptual models of realisation, which, in turn, are automatically transformed into more concrete BPEL artefacts during the implementation activity. The BPEL artefacts are executable in runtime environment as a part of concrete systems in reality. Section 6.4 illustrates how epitomes are realised, which provide a mechanism to encapsulate capacity in OASEF process. This section also presents the detail about capturing reusable engineering capacities and how to automate the process to generate concrete systems. In Section 6.5, change is introduced in OASEF higher-level models, which illustrates how to evolve business needs towards more complex business scenario using high level models and captured engineering capacities. Section 6.6 presents some observation during the case studies, including analysis of some identified strengths and shortcomings.

Finally, chapter 7 concludes the thesis by providing a summary of the research. The main achievements and contributions are presented, followed by the identified limitations and issues, which motivate important work in the future.

Due to the multi-disciplinary approach we take to deal with SOS (See Section 1.4.1), we do not make any assumption that the readers will be familiar with all related areas of this research. However, we do expect that the readers possess some technical and engineering background. That is, we assume that the readers are accustomed to some widely-known technical terms and concepts in the context of software engineering and systems engineering, such as the principle of separation of concerns and information hiding. As a result, some basic and well-known concepts will not be explained in this thesis. In the meantime, we will discuss and present some important knowledge and concepts that are ambiguous in nature, as well as new concepts and notions that are specifically developed for the purpose of this research. Therefore, this thesis does not need be read sequentially. Certain parts can be skipped if the readers have already been familiar with them, or can be referenced in a later stage when needed.
The text on this page is not visible in the image provided. Therefore, it is not possible to transcribe the content accurately.
Chapter 2

Engineering SIS and SOS

The illiterate of the 21st century will not be those who cannot read and write, but those who cannot learn, unlearn, and relearn.

Alvin Toffler

Contents

2.1 Concepts, terms, and definitions ........................................ 28
  2.1.1 SIS and SISE .................................................. 29
  2.1.2 SOS and Services Engineering .................................. 30
  2.1.3 Models, modelling, and MDE .................................. 39
  2.1.4 Socio-technical systems and human perspectives ............. 44

2.2 Challenges and Characteristics of SOS .............................. 46

2.3 Current Approaches .................................................. 48

2.4 Our vision ............................................................ 54

2.5 Summary ............................................................ 56
As presented in the previous chapter, the research set out for a clear goal within its scope, that is, to understand and address fundamental problems and grand challenges of SIS in general, and SOS in particular. It involves two interconnected parts: to understand the characteristics and challenges, and to cope with them. The former is important as it sets the scene and determines the potential value of the overall research.

Specifically, this chapter tackles two research questions: question I and II (See Section 1.4.2). That is: I) What are the nature, special characteristics, and important challenges of SOS and services engineering? II) What is the nature of existing solutions for engineering SOS? The answers come from the results from various activities in the iterative phases. For example, continuous and incremental Analyse challenges and problems and Observe existing solutions were conducted in various phases such as Touching, Exploring, and Focusing (see Section 1.4). The observation and analysis involve general activities such as literature review, trial and tentative experiments, and case studies.

The nature and characteristics of SOS and services engineering present great challenges for services engineering. As discussed in the previous chapter, for the purpose of this research, it is important to obtain clear understanding of such nature and characteristics and the state of the art solutions. The remainder of this chapter is organised as follows: in Section 2.1, we systematically develop some important concepts, notions, and technologies for the purpose of this research, such as SIS, SISE, services, SOC, SOS, services engineering, and MDE. We also discuss the notion of socio-technical systems, and introduce our focus of human perspectives. Section 2.2 presents an overview of the main characteristics of SIS and SOS in their socio-technical environments, and discuss the main challenges they present. Section 2.3 presents some related work based on our investigation and analysis of existing approaches to services engineering. Based on the study and analysis of these research questions, in Section 2.4, we present our vision to tackle the challenges of service engineering and modern SISE. Finally, we conclude this chapter with a summary in Section 2.5.

2.1 Concepts, terms, and definitions

In this section, we will present some important and relevant concepts, notions, and technologies within the scope of this research, in order to set the stage and put the research into context. Some important terms that are used throughout this thesis will be clarified and defined, particularly SIS, SOS, and services engineering. It is important to obtain precise understanding of these concepts and notions in order
to comprehend the following chapters.

2.1.1 SIS and SISE

This section clarifies the meaning and definition of SIS, which is the broad context of this research. Throughout this thesis, “SIS” are used as an acronym to refer to general software-intensive systems in today’s socio-technical environments. According to Ackoff [1971], a system is “an entity that is composed of at least two elements and a relation that holds between each of its elements and at least one other element in the set” [Ackoff, 1971]. A system therefore consists of multiple inter-related parts. A system being software-intensive means that software is important and indispensable part of such a system. Therefore, in our thesis, the term SIS represents any type of systems that use software to inter-connect its internal elements and external environment, such as hardware, people, processes and policies.

As discussed in Section 1.1 and 1.3, today’s socio-technical environments demand more and better SIS, which need to interconnect with each other, often at a global level. As a result, more and more SIS are integrated as an overall system. The term “System of Systems (SoS)” (sometimes referred to as SOS in literature) is used to refer to such inter-connected bigger system.¹ As the size of SIS grows, there is an ever-increasing demand for Software-Intensive System of Systems (SISoS), that is, larger systems consist of various smaller SIS to work altogether [Boehm and Lane, 2006]. It represents a collection of independently-operated or managed systems that work as a whole for various purposes [Maier, 1998].

Large and complex SIS and SoS in modern socio-technical settings present some unique characteristics. Understanding of nature and characteristics of these types of systems, regardless of their scope and size, is an important goal of this research. According to Boehm and Lane [2006], the important features of these systems are “not only that they integrate multiple, independently developed systems, but also that they are very large, dynamically evolving, and unprecedented with emergent requirements and behaviours, and complex socio-technical issues to address”. In other words, these systems in modern settings present the following characteristics: autonomy, dynamics, evolution, emergent characteristics, and people and social aspects. Other researchers offer other perspectives by given different definitions and understanding of SoS. Pei [2000] holds that the development of

¹Note that, in this thesis, we use the acronym “SoS” to differentiate it from “SOS”, a term widely used in this thesis to refer to Service-Oriented Systems. The upper-case letter “O” in the latter acronym represents the word “Oriented”, whereas the lower-case “o” in the former represents “of”. The term “SOS” will be explained and defined in Section 2.1.2.
SoS needs to focus on integration, interoperability and performance. Sage and Cuppan [2001] also point out the autonomous characteristics of these systems, such as operational and managerial independence, geographic distribution, emergent behaviour, and evolutionary development. Some researchers observe the social aspects involved in these systems. For example, Luskasik [1998] emphasises the evolution of a social infrastructure within large interconnected systems. According to Gupta [2003], a good leadership is necessary to address human factors involved in the systems, such as communication, coordination, personal motivation, and interests conflict. Some of these features and their meanings will be discussed and analysed within our scope in the following sections of this chapter.

As Boehm [2006a] points out, the widespread trend of intensive SIS interoper- ation requires suitable engineering support. Due to the special characteristics of SIS and SoS, an engineering discipline is required to specifically address them. In this context, SISE, therefore, is defined as an engineering discipline concerning the development and management of SIS and SISoS, in order to satisfy stakeholders needs in an efficient, dependable, and predictable manner, in the socio-technical environments.

2.1.2 SOS and Services Engineering

As described in Section 1.1.3, this research is particularly focused on SOS and services engineering. This section provides a very brief overview of some important concepts and principles involved, and gives a definition for SOS and services engineering.

2.1.2.1 Service

The notion of service have become pervasive nowadays. However, it has diverse meaning, for different people, in a broad socio-technical setting. A range of definitions are proposed in different disciplines, such as economics, marketing, and information technology [Rai and Sambamurthy, 2006].

At the social level, the meaning of services reflects a profound social movement away from manufacturing and agriculture that used to be the centre of economies and fundamental skills that people must possess [Chesbrough and Spohrer, 2006]. But this social phenomenon has been changed since the last century. Large proportion of people have shifted to knowledge-intensive work such as providing value-added services, other than working on manufacture and agriculture. Nowadays, services, in various forms, almost exist in every industry in the world [Levitt, 1972, Quinn, 1988]. Statistics shows that, over the past few decades, service sector
has rapidly grown as the majority of economies in most developing countries, and significantly contributed to the growth of social productivity and human living standards [Bitner and Brown, 2006, Quinn et al., 1994].

In such a social and economical context, an influential definition of service is proposed by Hill [1977] as “a change in the condition of a person, or a good belonging to some economic entity, brought about as the results of the activity of some other economic entity, with the approval of the first person or economic entity”. Such a definition emphasises the purposeful and acknowledged provision of change. Another definition stresses the characteristic of service as a “time-perishable, intangible experience performed for a customer acting in the role of a co-producer” [Fitzsimmons and Fitzsimmons, 2006]. It reflects the human perception of services in contrast with manufacturing products. Other researchers also suggest understanding of service from the perspective of value exchange or request and provision of value transformation [Quinn, 1988, Riddle, 1986]. For example, services are seen as a “provider-client interaction that creates and captures value” [IBM Research Report, 2004]. There is another definition to which people with engineering background would be more accustomed, that is, services are “capabilities or competencies that one person, organisation, enterprise, or system provides for another” [Vargo and Lusch, 2004]. These definitions and understanding of service provide an overall description of what services means to us from a broad social perspective. Reflection and understanding of them help engineers to recognise the value and purposes of the service systems they produce, a necessity for fit systems.

Due to our engineering scope, it is important to look at services from a perspective of technology and engineering. It is interesting to note Levitt [1972]’s concern in relation to human perception of services. Levitt implies that too much focus on tailored value for individuals will hinder the growth of services. Instead, it is important to focus on improving the quality and productivity of services using technological and methodological support. Quinn et al. [1994] also holds that advances of IT would greatly contribute to global service economies. As the service industries rapidly grow, more understanding of, and support for, services from a technological perspective has been developed. This is evidenced by rapidly growing research and activities on IT technologies, engineering processes and methodologies [Rai and Sambamurthy, 2006], such as web services and cloud computing.

Unlike more abstract notions of services in the social settings, web services are a specific technique in software to achieve value exchange. They provide a particular form of re-usable software capabilities, which can be independently developed and published by service providers, and located and invoked by service
consumers [Erl, 2005]. A more technical definition is provided by the World Wide Web Consortium (W3C), an international standard organisation for Web technologies, that is, "A web service is a software system designed to support interoperable machine-to-machine interaction over a network" [W3C, 2004]. This definition stresses machine-to-machine interaction and software interoperability, which is one of its main design objectives.

To help those people who are not familiar with web services to obtain better understanding of the above technical definition, we will briefly describe the essence of how they work. Typically, web services can be published on the network using textual descriptive interface written in Web Service Description Language (WSDL), a format that can be processed by machine. Other parties, such as other software systems, can look up these web services from a directory using Universal Description, Discovery and Integration (UDDI), a technical mechanism similar to yellow pages phone book. The published capabilities can be consumed by parties who are interested in them, using a standard communication messaging mechanism protocol, Simple Object Access Protocol (SOAP), on top of the Web protocol, Hypertext Transfer Protocol (HTTP). The interoperability of web services among different types of software systems is achieved essentially using standard languages and protocols, such as Extensible Markup Language (XML), HTTP, and other Web-related standards. By defining a set of standardised mechanisms to describe, publish, discover, invoke, orchestrate, coordinate, and manage web services, the integration and evolution of heterogeneous SIS is greatly facilitated.

It becomes clear that web services are in good alignment with the notion of service from a social perspective, if we compare their definitions and characteristics. Essentially, Web services, as a technical means, provide a unified and flexible means to inter-connect different types of software systems, in support of transformation or exchange of information. They enable capabilities in a software and computing form, which can be provided to various parties, either machines or human beings. This is exactly what social scientists mean by services: providing capabilities, exchange or transformation of value, or intentional change of state. Articulation of this alignment helps to understand how these technologies relate to knowledge and wisdom from other social perspectives, and foster our multi-disciplinary view and approach.

The alignment between social and technical understanding of services is essential. The scope, quality, and productivity of services at the social level need to be supported by their realisation mechanisms from the technical perspective, which should maximise the value of services and service society. Such alignment enables the traceability between social needs and technical provision, and therefore allows change of the former according to latter, or vice versa, whenever agile adaptation
2.1 Concepts, terms, and definitions

is required. In an era where manufacturing is no longer the centre of human purposeful activities, advances in both technology and social movement are critical to transform the service industry and society from its primitive phase to the next level. Along with such advances, the way services are understood, defined, built, and managed should co-evolve, in order to enable an exponential boost of human production of knowledge-intensive products for social needs.

2.1.2.2 Service Composition

One of the characteristics of web services is that individual service can be aggregated to form new services. In other words, new capability can be achieved by compositions of a number of services, each of which independently provides specific capability. This is often referred to as Service Composition [Peltz, 2003]. Individual services can be changed or replaced in a flexible way, without having to unnecessarily affect the composite service, due to the loosely-coupled and contract-driven characteristics of services. There are two types of Service Composition. One is called Service Orchestration, which has some central controlling logic to coordinate the interactions among participating services. Such coordination is necessary, especially for complex interaction, in order to form new emergent value-added capability as a whole. Whereas another term, Service Choreography, often refers to a type of Service Composition that allows each participating service to take initiative, and collaborate with each other via message exchange. In this case, there is no such a role as a central coordinator.

By using various forms of Service Composition, systems can be formed in a flexible way. Individual web service can be built from scratch using literally any development platform, or from existing legacy systems of any form, because web services are fundamental based on standards to enable interoperability among various vendors. Composite services can also be used to build larger services or systems. This flexibility makes it possible to form systems using services at various level of granularity, and in a flexible, or even dynamic, way.

Due to the importance of service composition in forming systems, we systematically investigate a range of current solutions by either literature review or practical experiment [Wang and Taylor, 2008], in the Touching phase of this research (see Section 1.4.3). The objective is to help to answer research question II (see Section 1.4.2): What is nature of the existing solutions for engineering SOS? In practice, there are many technical approaches build Service Composition, such as WSFL, XLANG, BPML, WSCI, BPEL, and OWL-S [Dustdar and Schreiner, 2005, Milanovic and Malek, 2004, van der Aalst, 2003]. What they essentially do is to model, design, and implement specific composition logic to synthesise individual
services together, to form a coherent and functional whole. Some important approaches to Service Composition, in other words, forming SOS, are analysed as follows:

- **Process languages**
  These approaches use specially designed process languages to model or implement service composition, such as BPEL, Web Services Choreography Description Language (WS-CDL), Business Process Modeling Language (BPML), Business Process Specification Schema (BPSS), and XML Process Definition Language (XPDL) [Juric et al., 2006, Wohed et al., 2003, Dustdar and Schreiner, 2005, Milanovic and Malek, 2004, van der Aalst, 2003]. Among these composition languages, BPEL has obtained wide adoption in both industry and academia in recent years, due to its simplicity compared with some other approaches and comprehensive infrastructure and tool support [Pasley, 2005]. It has been developed based on a number of other process languages such as WSFL and XLANG, and is capable of capturing complex service interaction and collaboration logic. Some useful features are also provided, such as long-running business process, exception handling, both synchronous and asynchronous messaging mechanism, and execution and run-time monitoring. [Pasley, 2005].

As an open standard, BPEL can be interpreted and executed by different BPEL runtime engines on different platforms. These engines provide BPEL runtime environments such as activity execution and invocation, state storage and fault handling. Some of these engines also provide management consoles to monitor the internal process activity execution. Various tools are also available to graphically model the composition process, to automatically generate BPEL artefacts, and to deploy and control them in runtime environments.

It is also criticised by some researchers and practitioners because of its limitations. For example, van der Aalst et al. [2003] argues that it can be tedious and challenging to develop and maintain complex Service Composition based on BPEL. This is that it is error-prone for people to manage complex and dynamic situations using implementation-oriented language in detail. Moreover, it contains some ambiguous constructs of either XLANG-style or WSFL-style [van der Aalst et al., 2003]. Explicit guidance is still lacking with regards to when to use which style, which makes the language less consistent. Another example is that there is no verification mechanism to detect inconsistencies and ambiguities at the early stages of services development [Milanovic and Malek, 2004].

- **Semantic Web Services (OWL-S)**
Semantic Markup for Web Services (OWL-S) provides another way to compose web services based on ontology, sometimes dynamically [Martin et al., 2004]. It is an important exploration to tackle the dynamic nature of service environments. One of its objectives is to support automatic and dynamic service composition and validation using Artificial Intelligence (AI) reasoning and planning techniques. It is an important exploration to tackle the dynamic nature of service environment. A lot of work has been done in this area, potentially enabling automated adaptation to services change.

However, a complete automation in dynamically complex real-world environments still presents a significant challenge. The technology is not mature enough to handle various situations in practice. More significantly, in the complex socio-technical environments of large-scale complex systems, the people-intensive perspectives cannot be modelled and manipulated in an automatic way. That is, it will a present challenge to understand and incorporate human and cultural influence on services, while dynamically reasoning about, and organising capabilities in a meaningful way. There are literally endless numbers of possibilities to be reasoned about, and taken into consideration during the planning processes. These limitations perhaps explain why OWL-S has not gained popularity in practice in complex real environments. Therefore, more research in this area is necessary, especially taking socio-technical environments into account.

- Formal Methods

There are also a huge number of approaches to assist Service Composition using formal methods, such as Automata, Algebraic Process, finite state machines and Petri Nets [Hull et al., 2003, Kazhamiakin et al., 2006, Fu et al., 2004, Lazovik et al., 2005]. This work aims to support formal modelling, generation, and verification of Service Compositions by introducing rigorous specification, composition logic, and transformation rules. They have great potential to ensure completeness, correctness, and high development productivity during Service Composition [Milanovic and Malek, 2004], by using techniques that have been used in practice for many years, especially in safety-critical embedded systems. However, in the context of SOS, this work currently remains in the experimental phase, and lacks sufficient empirical support.
2.1.2.3 SOA and SOC

Advances in web services and service composition have resulted in more powerful infrastructure, architecture, and techniques, that greatly facilitate an emerging way of building SIS using services. SOA, and more recently, SOC, representing the state of the art advance in this respect.

The Organization for the Advancement of Structured Information Standards (OASIS), an open standardisation organisation actively involved in the development of SOA, gives the following definition of SOA [OASIS, 2006]:

SOA is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations.

This definition focuses on capability provision and consumption, which is essentially what services, or web services, are about. This indicates the important position of web service technologies within SOA. It, however, tells very little about what SOA really means. In fact, there are a huge number of different definitions and understandings [Erickson and Siau, 2008]. Everyone seems to understand it differently. We believe it is better to understand SOA from the perspective of its value, and purpose. That is, what it has to offer, and for what purpose? To answer the first question, SOA provides a practical, extensible, and dependable technical architecture, that enables various types and sizes of applications and information systems to interact with each other in a cost-efficient way. It also explicitly provides a range of important design principles and guidelines to help to develop and integrate systems using web service technologies [Erl, 2008]. In a sense, SOA advances the concepts and scope of web services, and wraps them into a more comprehensive framework to assist systems development. Therefore, SOA makes it practical for enterprises to utilise the technologies, and therefore obtain a better position in technology adoption.

However, the adoption of SOA has not been smooth. It is criticised because of some limitations and unsatisfied expectation. For example, it is commonly perceived to be overly marketed by vendors [Erickson and Siau, 2008, Kaczmarek and Wecel, 2008]. Marketing hype raises users' expectation which may not be satisfied in reality, such as guaranteed improvements on productivity and system agility in all situations. On the contrary, sometimes, these desired ideals cannot be achieved due to the nature and characteristics of the system to be built or
2.1 Concepts, terms, and definitions

integrated [Erickson and Siau, 2008]. Moreover, the evolution of SOA is not always in alignment with the evolution of systems [Valipour et al., 2009]. While users need more assurance on quality attributes, the technology is not mature enough to satisfy such demands.

Despite the user dissatisfaction and disappointment, adoption of SOA, in reality, is steadily increasing. Empirical study shows that SOA is gradually emerging as the major technology for building and managing large systems in real world situations [Schulte et al., 2007]. Because web services technology takes a standardisation approach to ensure inter-operability, all elements within the architecture are built in an open, extensible, and coherent framework. This ensures that systems built with SOA technology can stay in an interoperable and open framework, which suits the globalisation demands in present socio-technical settings. Due to its valuable characteristics and features, SOA still offers a solid foundation for business agility and adaptability to some extent [Valipour et al., 2009]. Agile business change and mass customisation of products in the manufacturing industry have been achieved using SOA [Dietrich et al., 2007, Kim and Lim, 2007]. More systems have been built successfully using related technologies.

Moreover, another technical notion called SOC is also tightly related to services. Like SOA, its meaning is also vague. Again, different people would understand and explain it differently. Perhaps the only undeniable fact is that SOC is related to services, and to computing [Huhns and Singh, 2005, Papazoglou et al., 2007, Bichier, March 2006]. Because of its computing scope, presently, it is also realised by using web services, and therefore, SOA [Papazoglou et al., 2008]. One definition is proposed by Papazoglou and Georgakopoulos [2003] as “the computing paradigm that utilizes services as fundamental elements for developing applications”. We emphasise this one as it stresses the important goal of services, that is, to develop applications. Indeed, this is essentially what it is all about. This definition is perhaps intentionally kept as generic as possible to maintain broad applicability, which, however, could make it hard for people to quickly get a grasp.

We therefore further adapt this definition, for the sake of general applicability and better understand-ability. The purpose of services in SOC would be, more precisely, to engineer SIS, instead of to develop applications. In the meantime, SOC should incorporate all necessary resources and capacities to support engineering of SIS, such as theoretically or empirically justified principles, methodologies, and best practice. For example, just as Baldwin and Clark [2000] established modular design principles as the design guideline for decomposition of functional capabilities of systems, the design principles for SIS using services needs to be well understood and developed.
Since we have so far discussed the notion of services from both social and technical perspective, its potential for building and managing SIS, and the characteristics and purposes of SOA, it becomes clear what SOC really means:

SOC refers to a set of coherently related resources and capacities, such as concepts, principles, processes, methods, techniques, best practices, architecture, and infrastructure, which are used to derive, manage, and evolve SIS using services, both social and technical ones, in their socio-technical environments.

Ideally, SOC should enable SIS to be built, changed, and reused across computing platform and organisation boundaries. This enables new systems to be efficiently built and managed in a dynamic fashion. To improve the quality of SIS, and the productivity of engineering means that produce them, a mature SOC paradigm should incorporate both technologies such as web services, and social needs such as service demands and constraints. It also needs to tackle important technical aspects that have significant impact on the social side of systems, such as information integrity, security, trust, and user satisfaction.

2.1.2.4 SOS and services engineering

As discussed in the previous sections, advances in technologies, such as web services, SOA and SOC, all aim to make it easy to build agile and dependable software systems. Now, we should be able to define what do we mean by SOS and services engineering.

Maglio et al. [2006] define service systems as “value-creation networks composed of people, technology, and organizations”. This simple definition reflects our vision of a need to deal with SIS in broader socio-technical settings. It also incorporates the meaning of services from the perspective of value. However, it does not stress other important aspects of services, such as the support for wider interconnectivity and sufficient system agility. More importantly, it does not state the nature of, and internal relationship among, the value-creation networks. It is unclear what the value are and how such networks are inter-connected. To address this limitation, we give a definition for SOS as:

SOS is composite entity containing several parties, either technical or social, that possess particular capabilities, competencies, or value that are requested by, and provided for, other parties. The composition entity collectively shows emergent characteristics while interacting
2.1 Concepts, terms, and definitions

with its socio-technical environments that contain a wide range of aspects including physical resources, technologies, people and their communities, and require widened and coherent inter-connectivity and sufficient agility.

Simply put it, SOS is a type of SIS that consists of, either entirely, or partially, services, in socio-technical environments. This means that services are part of the SIS. They are also the means to build, change, and manage the SIS.

As a special form of SIS, SOS present some important and interesting emergent characteristics and behaviours. Some of these them were observed and analysed several decades ago, such as autonomy and dynamics, which were discussed in Section 2.1.1. In SOS, they become a prominent norm.

Services engineering is therefore defined as follows:

Services engineering is an engineering discipline that is concerned with the necessary resources and capacities for building, operating, and evolving SOS by using services in socio-technical environments that contain a wide range of aspects including physical resources, technologies, people and their communities.

Or simply put, services engineering is about ways to engineer SOS. Effective services engineering is essential to boost the service economies in globally dynamic environments. Like any other engineering discipline, the systems to be engineered, that is, the SOS, should be well understood; Needs for improvements should be identified and agreed upon; Suitable engineering decisions should be made and turned into actions; New states of SOS should be observed and evaluated, and should trigger another cycle of engineering process. In the end, the goal of services engineering is to satisfy various, and sometimes conflicting, needs or intention of stakeholders by using services.

2.1.3 Models, modelling, and MDE

While pursuing effective services engineering, it is important to observe the existing good practice in software engineering and SISE in general, since SOS is a special form of SIS. Our approach particularly synthesise some important concepts and techniques from MDE, an emerging means of modern software engineering. In this section, we will briefly introduce MDE and establish a definition of models and modelling in the context of SISE based on some notions and concepts of MDE. This is fundamentally important, because the pertinent terms, models and modelling, are referred to intensively throughout the thesis.
2.1.3.1 MDE

MDE evolves from traditional software engineering, such as model based development, Domain-Specific Languages (DSL), and generative programming, with an aim to continuously raise the level of model abstraction, and to encapsulate and automate lower level implementation detail [Yourdon, 1993, Czarnecki et al., 2002, Mellor et al., 2003, Atkinson and Kuhnne, 2003] Its fundamental essence is to use models as the essential way to build systems and tackle the fundamental software engineering challenges [France and Rumpe, 2007].

The emergence of MDE as a promising approach to general SISE is not without a reason. As described in Section 1.3, business and technologies co-evolve in a socio-technical setting. On the one hand, more complex SIS are demanded within a short period, on the other hand, understanding and application of diverse implementation mechanisms in a coherent manner becomes difficult. Traditional technology-focused approaches have become ineffective to manage such complexity [Schmidt, 2006]. Instead, a focus on more abstract aspects of the systems becomes critical. MDE could bridge the gap between business and technical complexity, by using models to promote abstraction, reuse and automation during engineering processes, which leads to better engineering productivity and product quality [Mellor et al., 2003, Schmidt, 2006].

There are a number of ways to realise the vision of MDE. Object Management Group (OMG)'s Model-Driven Architecture (MDA) is one of them. It aims to develop standard ways of defining and transforming models to achieve good interoperability [OMG, 2008]. The idea is to derive platform-independent specification of systems at higher level of abstraction, using modelling standards such as Unified Modelling Language (UML) and Meta-Object Facility (MOF), and to transform them into proprietary implementation platform according to well-defined transformation rules and specification using techniques such as Query View Transformation (QVT) [OMG, 2008]. More specifically, models are arranged at four level: 1) Computation Independent Model (CIM), the business models that are independent of technologies; 2) Platform Independent Model (PIM), the technical models to realise the business needs, but without dependencies on particular implementation platforms such as Java or Microsoft .NET; 3) Platform Specific Model (PSM), the models with detailed technical design that is specific to underlying target platform; and optionally 4) Implementation Specific Model (ISM), the implementation and runtime models that can be directly mapped to software code or execute in the run time. MDE specifies the standards to define, manipulate, and transform these models. Through multiple level of models, business and technologies complexity could be managed separately at different
2.1 Concepts, terms, and definitions

level of abstraction, which could facilitate the co-evolution of both business and technologies [Mellor et al., 2004, OMG, 2008]. Due to its practical value, MDA attracts a lot of industry attention and has been used in many software projects [Frankel, 2003, Pastor and Molina, 2007].

However, in our view, the effectiveness of MDA in terms of modelling the business part of systems need further justification. Although a MDA approach enables modelling of business domain using UML and a number of other high level modelling language [Pastor and Molina, 2007], it provides insufficient support for understanding and capturing general human intellectual efforts during the high level modelling processes. Arguably, MDA still has a strong focus on technologies, which is evidenced by the large portion of involvement of technology providers, as well as the technology focused content of the produced standards, especially model transformation standards. Its standards are principally built by, and arguably, for, people with more technical background and commitment. Therefore, there is a lack of important human aspects as a complement to the current technological focus.

In addition to MDA, there are also a range of other approaches to MDE, such as Product Lines, Software Factories [Greenfield et al., 2004], Generic Modeling Environment (GME) [Ledeczi et al., 2001], and DSL and Model Integrated Computing (MIC) [Sprinkle, 2004, Balasubramanian et al., 2006]. A range of tools and supporting environments have been developed to support these various approaches to MDE, such as Eclipse Modeling Project (EMP) [EMP, 2010], Microsoft DSL Tools [Microsoft DSL Tools, 2010], GME, and other MDA compliant tools. Active research from both industry and academic demonstrates the great potential of MDE. After more than a decade of evolution, MDE is steadily approaching its vision through various propelling forces, including advances in emerging standards, more comprehensive supporting tools, feature-rich development environments, and active communities [Hailpern and Tarr, 2006, Jouault et al., 2009].

2.1.3.2 Models and meta-models

The critical position of models in MDE reflects their importance in the process of human purposeful problem-solving. Indeed, human beings use the notion of models to reason about the world, make plans, and to take action to achieve the desired ends. But what do models really mean for MDE, or more generally, for SISE?

The word “model” has its origin from the Latin word modulus that means “measure, rule, pattern, example to be followed” [Ludewig, 2003]. Although being a widely used, models are very difficult to define. Generally speaking, in the context of engineering, models can be seen as “a set of statements about some systems under study”, where statements means some expression about, or
description of the system that is either true or false [Seidewitz, 2003]. That is, models relate to systems. To understand the exact relationship between models and the systems they refer to, Ludewig uses Stachowiak [1973] 's three criteria. That is: 1) "Mapping criterion": models are equally mapped to the systems in reality in a conceptual sense. Therefore models provide conceptual reference of complete systems so that human beings can think over the problems and conceive solutions; 2) "Reduction criterion": models are one form of abstraction that removes irrelevant properties of the systems, but focuses on some relevant properties. In addition, models can also provide generalisation of these properties, that is, to show the similarity of properties and hide the differentiation, among classification of various type of systems [Peckham and Maryanski, 1988]. Models can show different levels of abstraction depending on how close they reflect the systems and the position they are within the generalisation/specification process; and 3) "Pragmatic criterion": models are the systems, and can be used in real-world situations for the purposes of the systems, at least partially.

Therefore, common understanding of models is in correspondence with particular systems under consideration, which we will refer to as original systems. Generally, there are two types of models, depending on the state of the original systems that these models refer to. They are: descriptive models that describe the existing systems in reality, and prescriptive models that specify the future systems to be built [Ludewig, 2003].

Whereas the models themselves, in relation to the original systems, are different existence of systems, which we will call as artificial systems. These systems are "artificial" in the sense that they are derived for the purpose of understanding and improving the original systems. Moreover, the artificial systems can be either conceptual systems existing in human mind, such as knowledge and wisdoms, virtual systems such as software, or physical systems such as machines. They can also be combinations of various types of systems involving human, software, and hardware altogether, that is, the socio-technical systems, which will be further elaborated in Section 2.1.4.

Based on the understanding and analysis of the nature of models and their relationships with original systems, we give the following definition for models in the context of SISE:

A model is an artificial system, either descriptive or prescriptive one, that refers to, or replaces, the original system under consideration, either existing or envisioned one, either partially or in total, and either conceptually or in reality, for the purpose of obtaining better understanding and improvements of the original system of concern.
2.1 Concepts, terms, and definitions

Meta-models are a type of general models that specify common characteristics or universal rules to which particular instances of specific models conform [Pastor and Molina, 2007]. Alternatively, meta-models can also be seen as a language that is used to represent a model [Bézivin, 2005]. According to Pastor and Molina, meta-modelling is the process of finding common universals that can be used to define or represent conforming particulars. That is, modelling processes involve derivation of meta-models, based on which instances of models are created and manipulated.

Moreover, since a meta-model itself is also a model, it is represented by using a language of its own, that is, its own meta-model. The latter is sometimes called meta-meta-model. This relationship between models and meta-models forms a hierarchy of models structure. For example, the instance-class relationship forms multiple layers of models and meta-models in MOF, a unified meta-model framework to enable easy definition and integration of meta-models among various platforms [OMG, 2006]. That is, a model is an instance of a meta-model, which in turn, is an instance of meta-meta-model. Besides MOF, there are a number of other modelling architectures for defining and manipulating models and meta-models, such as EMF [EMF, 2009], KM3 [Jouault and Bézivin, 2006].

2.1.3.3 Modelling

Models form intellectual artificial systems produced by human beings to understand and manage the original systems. It is important to understand how to derive and manage such intellectual systems. The term “modelling” is for this purpose. In this thesis, we use modelling to refer to the means, or more specifically, the processes and techniques, to derive, express, and manage intellectual models. Accordingly, meta-modelling is the process of finding common rules and expressing languages that can be used to define or represent conforming models, which are particular instance of these rules or universals, or representations by these languages.

Clear understanding of models, modellings, and other concepts and principles of MDE is crucial to understand the mechanisms we used to address the complexity and volatility challenges, which will be seen in the following chapters. Essentially, models and modelling enable us to develop and exploit intellectual capacities and engineering competence to conquer the problems and challenges introduced in chapter 1.
2.1.4 Socio-technical systems and human perspectives

As discussed in Section 1.2.2, a lack of human perspectives is a fundamental problem for engineering modern SIS. By their nature, SIS have to involve people. They are built, operated, and managed by people, and for the purpose of satisfying people. From a system point of view, humans are indispensable components of SIS and SOS, which intensively interact with other parts of the system.

The recognition of the important position of human beings in complex system has been emphasised by many researchers since last century. A concept called Socio-Technical Systems (STS) is introduced to refer to complex systems that incorporate technical systems within a human world [Bostrom and Heinen, 1977]. For STS, both social and technical components have to be considered in relation to the interactions between non-human systems and human systems [Trist, 1981]. According to Trist, psychosocial aspects such as cultural values and norms or the social structure of STS are parts of the overall systems, and need to be understood collectively.

In the engineering context, we don't make a distinction between the psychosocial aspects and other socio-technical aspects of STS. Instead, we use the term STS, in a more general sense, to refer to any systems that involve interactions among both technical components such as hardware and software, and non-technical components such as people, interest group, organisations, communities, processes, laws and regulations. Therefore, STS involve two aspects that co-exist and inter-relate to each other: technical aspects such as realisation mechanisms and technologies; and human-related aspects such as cognitive and social phenomenon. Examples of these systems are public health systems, or complex financial systems. Similarly, we use the term Socio-Technical Environments (STE) to refer to the environment of SIS that involve both technical settings such as physical environments, hardware, and software, and social settings with human aspects, such as cognitive, psychological, psychosocial, political, and religious surroundings.

As Sommerville [1998] states, there is a significant consequence for a lack of acknowledgement of socio-technical aspects of SIS, such as environments that have intensive involvement of people and policies. Such a lack often leads to unfit systems and unhappy users. Many of today's engineering problem are caused by a failure to recognise and manage interactions between people and other parts of the systems [Sommerville, 1998]. Therefore, We believe SISE and services engineering need to regard human and social aspects to be as important as technical aspects.

However, it is extremely challenging to understand and manage human beings. This perhaps explains why there has been a lack of human aspects in SISE. Indeed, unlike other objects, people are complex and intrinsically unpredictable. They
2.1 Concepts, terms, and definitions

have various identities, backgrounds, believes, values, and cultures. They have emotion, feeling, and passion. They can be creative and constructive, but also make mistakes, sometimes severe ones, which impose great risks.

Our work on SISE pays attention to the human part of STE, particularly, the human mind and intellectual efforts in the context of engineering. This is because of the crucial role human mind plays in engineering.

Human mind and consciousness, as our major means to understand and interact with the outside world, is arguably the most hard-to-understand, complex, and controversial thing that human beings have discovered in the universe [Minsky, 1965, Guezeldere, 1997, McGinn, 1989]. As evidence, no widely accepted definition and understanding of consciousness and mind has been achieved during several centuries of development in philosophy, psychology, neurobiology, and computer sciences [Lewin, 1993, Dennett, 2001, Voorhees, 2000]. Crick and Koch [1990] argues that although everyone has a rough idea of what is meant by consciousness, it is often “better to avoid a precise definition of consciousness” because of its nature is not definable by human beings have consciousness of their own.

There are various understandings of mind and consciousness in philosophy [Humphrey, 1999], Dualism represents the thought that separates the physical self, the body, from the sense of self, the mind that resides in the former but remains discrete [Chalmers, 2002, Baker et al., 2002]. Materialism, dominating the modern thinking about consciousness, instead regards mind as a physical phenomenon that is identical to its causal computational partner, the brain [Ryle, 1949, Dennett and Weiner, 1993]. Computational functionalism, one modern variation of materialism, regards the cognitive process of thinking as information processing via computation function [MacCormac, 1984, Dennett and Weiner, 1993, Churchland, 1996]. The computation model of consciousness is criticised by many other researchers who emphasise the facts of human subjective feelings, intention and creativity [Penrose and Gardner, 1989]. Contrarily, mysticism believes that consciousness is rather irreducible, and essentially cannot be explained on the basis of materialist or computational models [McGinn, 1980, 1989]. McGinn [1989] believes that an understanding of consciousness is inevitably beyond the our thinking capacity because we are not cognitively equipped to understand human mind in the same way as we understand other physical phenomena. Mysticism is basically believed to be unjustifiable in the modern scientific framework, and therefore is ignored by most scientific studies of consciousness. As a result, our work mainly takes the Materialism viewpoint, while acknowledges the cognitive limitation of human mind and intellectual capacities during engineering, which justifies the importance of capturing and reuse engineering resources and intellectual capacities, a mechanism we will present in Chapter 5.
A comprehensive review of various understanding and schools of thought on human mind is beyond the scope of this thesis. But there is one conclusion we can draw from the literature review. That is, it is crucial to understand human mind, despite the difficulty and confusion it presents. As Chalmers [2003] points out, human mind should be carefully and systematically studied due to its critical role in human activities, despite the unresolved controversies, confusions, and mysteries around consciousness and mind over a long history of humankind. The importance of human mind is especially true in the context of engineering. After all, human minds are what we used for any intellectual engineering achievements. Therefore, we need to obtain better understanding of the nature, characteristics, and recognisable and reusable patterns of human mind during engineering activities, and to utilise such understanding to conduct and facilitate engineering SOS. Chapter 3 and 4 will delve into this area in detail, in pursuit of a well-founded philosophical and theoretical basis for our approach to services engineering.

2.2 Challenges and Characteristics of SOS

The ability to understand and cope with the evolving socio-technical environments is crucial to satisfy the increasing needs for effective and dependable SOS. Due to the nature of SOC and service economies, SOS inherently involve more complex, heterogeneous, and volatile environments, which is referred to as open environments [Wang, 2010a]. The special characteristics of such environments present great challenges for services engineering [Papazoglou et al., 2006].

Firstly, SOS often involve great complexity in terms of variety, scope, and inter-relationships. On the one hand, technologies such as SOC and Cloud Computing (CC) greatly facilitate development and integration of applications and systems. The concept of service-orientation involves arbitrary systems as long as they provide the required capabilities that are of value to others, regardless of what software and hardware platform they use, or what physical environment they reside in. On the other hand, economical and social globalisation processes inevitably force individuals, organisations, and communities to collaborate and compete with each other within interacting value-chains. Continuous interconnection of a large variety and quantity of services, subsystems, individuals, or organisations inevitably increases system complexity. As a result, the size, variety, and scope of inter-related systems scales up rapidly in the context of SOS.

Secondly, SOS often present a high degree of uncertainty. Autonomous services are often developed, published and managed separately and differently. Such
2.2 Challenges and Characteristics of SOS

an autonomous and loose-coupled nature of services presents a new challenge for engineers. For example, services are provided and consumed by various agents for different types of purposes, without necessarily knowing each other in advance [Chang and Kim, 2007]. Therefore, it is difficult to satisfy not only the explicit requirements of current target applications, but also the needs of envisioned future applications or unknown potential users. Defining a service contract cannot solve the problem as pre-definition at any time cannot completely incorporate unknown usage with necessary variations. As a result, special consideration needs to be made in order to derive, identify, and use suitable services for all possible types of similar, but varying, purposes, and to guarantee that suitable services are used for specific requirement specifications. For example, adequate qualitative and quantitative measurements need to be applied to ensure the right match between service requirements and service provision [Reuther and Henrici, 2008]. The uncertainty involved in service consumption in real world therefore present great challenges, which cannot be addressed by traditional build-to-specification approaches that get stakeholders to agree on clearly defined specifications about the systems-to-be.

Thirdly, SOS evolve independently in volatile environments that changes their structure and behaviour frequently and dynamically. The dynamic nature of services and their environments requires flexible methodologies that efficiently manage and respond to change [Krogdahl et al., 2005]. For example, service provision and service discovery can rapidly change in the run-time and therefore affect the overall performance of SOS that are built on top of loosely-coupled services. The dynamics of autonomous services participated in the SOS need to be managed in order to avoid impact on the overall systems. According to Ashby's Law of Requisite Variety [Ashby, 1958], in highly volatile and heterogeneous environments, the velocity and variety of appropriate responses have to match those of the environmental change. Therefore, the open environments of SOS require organisations to manage and adapt to change effectively and efficiently, and hence to maintain competitive advantage that ensures its continuous business success [Brier et al., 2006]. Since the entire ecosystem of SOS is build on top of volatile services at different level of granularities, for a given SOS, both its external environments and internal components reveal volatile change. Therefore, it presents great challenges to support and facilitate sufficient adaptation to external environmental changes while managing the internal changes in terms of structures and behaviour, in a more frequent and predictable fashion.

Fourthly, sufficient and effective communication among stakeholders becomes crucial in service environments. Service-orientation promotes specialisation of skills and delegation of services instead of building and managing everything
from scratch [Erl, 2005]. On the one hand, it requires specific focus on particular aspects of the domain of interest, on the other hand, it needs intensive cooperation and coordination among large numbers of, potentially unknown, and often vaguely-defined, stakeholders across enterprise boundaries. The nature of specialisation and integration requires wider multi-disciplinary collaboration and interoperability. Effective communication among various stakeholders is hence one of the most critical factors to ensure system fitness, in other words, the success of systems.

Fifthly, it is challenging to manage the contextual and state information regarding SOS under consideration. For example, contextual behaviour and state history are not captured within services, or anywhere else. The contextual information is important to ensure sensible and meaningful service provision. For instance, the invocation of services in a certain context has to refer to specific external information, which may affect behaviour between successive invocations [Ceponiene and Nemuraite, 2005]. Moreover, the contextual information within a certain usage context also co-evolves with the services, which makes it difficult to change the services while at the same time explicitly maintaining their external states. Therefore, it becomes challenging to manage state information explicitly and externally in a service-oriented environment, while services interact and collaborate with each other in long-running business processes.

Lastly, but not less importantly, services need to expose a higher degree of quality of service. This is especially challenging, because although different users may require the same feature or capability offered by a service, they may have different non-functional requirements, such as certain degrees of performance and security within different contexts [Wada et al., 2006]. This kind of flexibility requires careful service design and variability management. Moreover, since services are highly reusable, any mistake or risk will be duplicated and amplified in future reuses. This means that services need to satisfy greater non-functional requirements such as better quality, availability, reliability, performance, and security, in comparison with traditional distributed or localised applications.

2.3 Current Approaches

The previous section presents an answer to research question I: What are the nature, special characteristics, and important challenges of SOS and services engineering? A good understanding of these characteristics and challenges is an important precondition for the development of an effective approach to services engineering. However, as noted in Section 1.3, current practices have not
addressed these issues effectively. Existing approaches either directly adopt traditional software processes and engineering methodologies, or merely focus on applying advanced implementation technologies without sufficient support for high-level intellectual effort of humankind. This section investigates some of these approaches and discussed their insufficiency.

The emerging discipline of services engineering attract a lot of research attention. As pointed out by Papazoglou and van den Heuvel [2006], the complex environments of SOS demand a sound discipline of services engineering, which allows organisations to carry out engineering activities to understand and deal with complex situations within service value, such as new business models and relationships with business partners and customers, and emerging technologies and techniques in support of business operations in the service economies. In particular, Papazoglou and van den Heuvel analyse some methods and techniques used in services engineering and introduce a tentative methodology that involve multiple phases, including: 1) a planning phase to understand the business environment; 2) a analysis phase to understand the “as-is” system requirements and to derive the “to-be” specifications of desired business processes through various analysis activities; 3) a “service design phase” to design the interfaces of concrete services to realise the “to-be” systems; 4) a “service construction phase” to implement or provide services; 5) a “service test phase” to verify the derived SOS; and 6) “service provisioning phase” that involves “service governance, service certification, service enrolment, service auditing, metering, billing and managing operations”. While this work inspires our research to a large extent, it is at a conceptual level and requires refinement at an operational level. Moreover, it emphasises on adaptive activities such as service provisioning, deployment, execution, and monitoring, with a vision to achieve the service adaptation through dynamic adaptation in response to environmental needs and changes. While we believe it is important to realise such a vision of run-time adaptation, we believe it is also important to achieve flexibility and adaptation at another level, that is, the engineering processes level before SOS are derived and operated.

As argued by Amyot et al. [2008], model-driven approaches provide great support for services engineering. Services emerge from from analysis of business domain, and can be derived, specified, implemented, and deployed through an automated process based on modelling and model transformation. That is, SOS can be modelled at a high level of abstraction using modelling languages that are designed for the end users and domain experts. This approach therefore allows better understanding and capturing of problem space by domain users, as oppose to platform and technology providers. Moreover, these high level models can be used to derive detailed design and implementations with a high degree of automation.
This approach is in good alignment with our research, which is based on a similar vision to incorporate model-driven techniques in services engineering as a concrete mechanism to tackle the identified service engineering challenges, which will be further discussed in Section 3.3.5.

A range of such model-driven approaches have been proposed to facilitate services engineering. Higher level process modelling languages such as UML Activity Diagram and OCL [Skogan et al., 2004, Schmit and Dustdar, 2005, Orriëns et al., 2003] and Business Process Management Notation (BPMN) [Ouyang et al., 2006] are used to analyse and model service composition as well as quality attributes. These models can be translated into implementations using lower level languages such as BPEL. Some specific techniques are proposed to achieve model transformation such as deriving BPEL artefacts from XML Metadata Interchange (XMI) representations of other process models using Extensible Stylesheet Language Transformations (XSLT).\(^2\) Although this work provides some support for the design and implementation of SOS, it concentrates on technologies such as implementation and model transformation techniques. They provide no answer with regards to how these higher level models are created, what human factors influence their quality, or what mechanisms can be used to improve these high level business models.

This issue is realised by some researchers. For example, Luthria et al. [2009] propose a value-based requirements engineering method to align service and system requirements with the business strategy and core business value. Their approach incorporates a value-based decision mechanism to derive the high level business models for SOS. It, however, does not offer a concrete mechanism to link the business value with implementation. Cardoso et al. [2009] attempt to solve this issue by proposing a services engineering methodology called ISE to incorporate a wide range of service activities based on both “perspectives” and “dimensions”. The former includes various phases from different view points, including “business”, “conceptual”, “logical”, “technical”, and “runtime”. Whereas the latter consists of “service description”, “workflow”, “data”, “people”, and “rules”. Service engineering activities are hence organised as a matrix of grids based on these “perspectives” and “dimensions”. For example, service modelling from the “business” “perspective” involve all “dimensions” such as “workflow” and “data”. It then uses MDA techniques to model the outcomes of individual activities. This work understands services engineering as deriving and managing electronic services using modelling techniques, and thus focuses on how to interpret organisation business from a service perspective [Cardoso et al., 2009]. Its static structure to manage understanding of service in organisation contexts, however, does not take

\(^2\)XMI is an OMG standard for exchanging metadata information via XML [XMI, 2007].
into account the dynamics and inter-dependencies among various “perspective” and “dimensions”. For example, what relationships exists among “workflow”, “data”, and “rules”, and how does change in one perspective influence another perspective? This methodology does not model these dynamic relationships and accommodate the interconnected change.

There is also some other work that focuses on creating linkages between the business and technologies in the context of services engineering. For example, Li et al. [2008] present a platform to integrate various views from both business and IT perspectives, such as business view, process view, and service view. This approach applies both top-down business design and bottom-up service composite via modelling and model translation mechanisms, including “service meta-model”, “process model”, and “business model”. Collaboration among various stakeholders is achieved via visual development and rapid demonstration. Similarly, Adam and Doerr [2008] argue that the focus on application and systems at the business and implementation level needs to be in good alignment with the focus on service design and provision. They propose a service-oriented requirements engineering (SORE) method to identify the provision of service capabilities on the right level of abstraction and variability, which is linked with business process management through product line technology, which collectively work as SOS that suit the business needs at high level of abstraction. As another example, Lamparter and Sure [2008] propose an interdisciplinary methodology to combine a web service engineering method with marketing and ontology engineering that aims to coordinate services and customers in a collaborative environment. Its web services engineering methodology consists of System Analysis, Domain Decomposition, Component Design, Service Design, Service Binding, and Evaluation. It also incorporates a common ontology engineering methodologies containing the stages of specification, conceptualisation, formalisation, implementation, and maintenance [Pinto and Martins, 2004].

However, these model-driven approaches to link business processes with service provisions do not capture and reuse the capability to achieve coherent models from the perspective of either business or technology. For example, the approach from Li et al. lacks understanding and support for the inner processes of deriving models of business view and service view. Whereas Adam and Doerr’s approach provides no guidance to derive service capabilities at the right level of abstraction, which cannot be assured in a systematic sense. Therefore, the intellectual engineering effort of services engineering remains unexploited or gets lost over time, and cannot be accumulated to facilitate system agility and dependability for future projects and engineering experience. The success of engineering activities and collaboration solely depends on the competence and experience of people.
involved in the engineering processes, which cannot be predicted, controlled, or improved in a systematic way.

There are also a range of other approaches to services engineering. Some aspect-oriented approaches are proposed in the context of model-driven development [Courbis, 2005, Cova Suazo and Aguirre, 2005, Hmida et al., 2005]. In this work, Aspect-Oriented Programming (AOP) concepts are applied to composition languages in order to separately manage cross-cutting concerns, such as security and transactions. Although this approach facilitates the management of complexity in a flexible way, it does not address the overall challenges of services engineering for non-cross-cutting properties. There are also other formal methods, and AI approaches, such as model checking, automatic service reasoning and composition [Pistore et al., 2004, Ragone et al., 2005]. The main issue of these approaches is scalability. For large scale complex SOS, complete automation in real-world environments is unlikely to be achieved in the near future without a ‘human-in-the-loop’ to make complex decisions.

New technologies and standards emerge in engineering related areas, such as more powerful computing infrastructures and advanced means to provide them, improved human-machine interfaces, and better support for integrated social environments. Such technical advancement, such as emerging technologies of MDE, SOC, and CC, is adopted to support better SISE. However, it often involves arbitrary combinations of technologies and frameworks at the time of development, whichever are chosen to address the engineering problems at hand. Little guidance and assessment is provided to apply various independently developed technologies for different purposes. As a result, these approaches either have strong dependency on complex and quickly evolving implementation technologies, or blindly adopt the traditional processes and methods. Sometimes there is no explicit processes applied at all.

This situation is problematic. Generally, these approaches do not take into account the characteristics of services engineering described in the previous section. The history of software engineering tells us that fundamental emergent challenges cannot be resolved by merely adopting traditional software processes and engineering methodologies, or solely relying on more advanced implementation technologies [Dijkstra, 1972a, Brooks, 1987]. The implementation-focused approaches have some serious problems.

Firstly, technologies often have a strong focus on a particular aspect of the problems, and therefore often lack high-level consideration in a broad socio-technical context. Current practice of services engineering often involves arbitrary combinations of technologies and frameworks at the time of development which
is believed to be relevant to address the engineering problems at hand. However, different technologies are separately designed and maintained to solve different types, or aspects, of problems, without a common goal of helping the overall engineering productivity or the quality of software processes and their products. Some of them are not compatible with others due to various reasons, such as an intention to get exclusive market positions. There is therefore no guarantee that combination of various technologies will coherently work altogether and produce systems that work as a whole. Therefore, it presents great challenges to integrate different technologies within a cohesive development environment.

Secondly, rapidly evolving technologies increase the accidental complexities, which creates extra heavy burden for engineers who have to deal with rapidly evolving technologies, and therefore cannot focus on essential complexities [Pastor and Molina, 2007]. Without an effective means to manage increasing low-level accidental complexities and high-level essential complexities, an engineering approach would be difficult to build in the first place, error-prone, and hard to maintain during successive operation.

Thirdly, a strong focus on particular technologies often raises communication problems among various types of stakeholders, because different stakeholders are specialised in, or familiar with, different aspects and phases of the development process. Not all stakeholders have knowledge of specific technologies. It is unnecessary, and often impossible, to get them to understand the technical design and implementation details. Therefore, a process driven purely or mainly by particular technology cannot facilitate wide participation and mutual understanding among different stakeholders. The lack of communication and slow process iteration cycle also prevents the collection of user feedback in the early stage, and often makes the system ineffective and hard to change at a later stage.

Fourthly, this approach makes engineering activities inflexible and ineffective to cope with necessary and critical change. As discussed in Section 1.1.1, today’s socio-technical settings are increasingly changing. Both business and technologies are emerging, co-evolving, and disappearing, at an increasingly rapid pace. Therefore, the strong dependencies on technologies cannot cope with change on the non-technical side. In the context of services engineering, new technologies and standards emerge and evolve quickly. This change can be seen everywhere, such as service infrastructure, security and trust, and service orchestration and choreography. Therefore, processes driven by rapidly evolving technologies end up with a need for more frequent upgrade, re-factoring, re-design, and therefore more unstable systems. This type of change does not necessarily reflect the essential needs of the stakeholders, therefore is not desirable. Instead, what is desired is the change made for agile business adaptation and innovation due to intensive
competition in globalisation and service economies.

Lastly, the slow process iteration cycle and a lack of common ground for communication also prevent effective collection of user feedback at early engineering stages, which makes it difficult to make necessary change to the systems at later stages. Therefore, these approaches provide insufficient support for agile decision-making and engineering process in the dynamic economic and social environments of services engineering. This is because it is difficult to incorporate flexibility and agility into current disciplined processes that are mainly based on the rigid decomposition of system life-cycle at single level. This limitation makes these approaches ineffective to cope with necessary change in dynamic open environments, where both business and technologies are emerging, evolving, and disappearing, at an increasingly rapid pace [Ghezzi, 2007]. Without sufficient means to incorporate information, capabilities and resources to make agile decisions in specific unsatisfied situations, human beings cannot improve the engineering process to match the social needs.

Therefore, there is an urgent demand for a new approach to services engineering that enables development of complex SOS in a cost-efficient, reliable and flexible manner. After analysing the characteristics of SOS and existing approaches, we argue that the new approach in need should acknowledge and foster human intelligent activities and achievements in the world of human mind, and hide unnecessary technical complexities by enabling and facilitating reuse of important engineering capacities and knowledge. We believe this approach could be effective to tackle the special characteristics of complexity and volatility. The next section presents our vision of such a new approach.

2.4 Our vision

As discussed in the previous section, existing approaches are insufficient to address the difficult challenges of services engineering. The great difficulty indicates that a radically different approach might be needed to tackle this problem. An ideal response to these issues would be a coherent body of knowledge, theories, policies, processes, technologies and supporting infrastructure, that collectively resolve these issues in reality as desired, when they arise on the time scale. As described in Section 1.3, this research is aiming for this ideal. Therefore, the ultimate objectives of this research is to explore and exploit a practical means to address the great challenges when engineering SOS and modern SIS in their socio-technical environments.

Based on the literature review presented in the previous sections, we believe
one way to approximate this ideal is to enable and promote derivation and reuse of both higher-order human intellectual efforts and lower level technological capacities. Evidence has shown that great complexity involved in engineering problems is better dealt with by intellectual cognitive experience and capacities, which led to sensible perception of reality, logical reasoning of problems, and systematic derivation of conceptual plans [Ackoff, 1999]. Similarly, other researchers argue that raising the level of abstraction in software engineering would help to deal with increased complexity and volatility [Pastor and Molina, 2007]. Therefore, we believe that effective and efficient accumulation and reuse of important intellectual resources is a crucial factor to manage the ever-presenting complexity, uncertainty, and volatility. In the meantime, in order to take full advantage of advanced technologies and implementation infrastructures, and at the same time, to deal with their increasing heterogeneity and complexity, a discipline of services engineering should also exploit effective means to develop, encapsulate, and automate the engineering capacities and processes about lower-level realisation and implementation.

Therefore, our vision of an effective services engineering approach should enable and support efficient and flexible formation and exploitation of both higher-order intellectual resources and lower-order implementation processes. That is, the above two types of engineering resources should be explicitly identified, developed, captured, and used, as major engineering means, to produce and manage systems, in a flexible, systematic, and automatic means, at least partially if processing in its entirety is not possible.

Specifically, such an approach: 1) promotes a focus on high-level human intellectual activities within the world of human mind, such as exploring and understanding the problems at hand, and identifying and capturing higher order engineering purposes and intentions; 2) provides a means to capture, aggregate, and reuse these important engineering resources, including both higher-order and lower-order capacities, to facilitate flexible and rapid aggregations of processes and systems at implementation level; 3) links the captured higher-order engineering resources with other lower-order engineering activities by using the former to guide and shape the latter systematically and sensibly, within a coherent overall process; and 4) has a flexible process that can coherently organise engineering activities, and allow systems to be built in a systematic and efficient way; 5) facilitates sensible and agile engineering decision-making by accumulating and utilising high level intellectual resources and low level technological capacities.
2.5 Summary

To conclude this chapter, this chapter provides answers to research question I and II (See Section 1.4.2). What we did first was to systematically establish a theoretical system, in which some important notions and concepts are clarified and defined, including SIS, SISE, services, SOS, services engineering, MDE, models and modelling. We also discussed the concept of STE, and introduced our focus on human perspectives, that is, the human cognitive activities and consciousness in the engineering context. We then discussed and analysed the characteristics and challenges of SOS and services engineering, including a high degree of complexity, uncertainty, volatility, communication and collaboration, context and state management, and a demand for higher quality of services such as dependability. Some existing approaches to address these challenges were investigated and analysed. We especially discussed some existing model-driven approaches to services engineering, because of the similarity with our approach. We drew a conclusion that a different approach is necessary to tackle the services engineering challenges. Based on the analysis and insights of relevant problems, concepts, and technologies, we developed a different vision of services engineering. In our view, such a new discipline needs a radical paradigm shift from a closed-world view toward an open-world one, that is, acknowledging and accommodating the highly dynamic, autonomous, heterogeneous, varying, and decentralised nature of the STE of SOS. We believe such a discipline needs to take an open approach that incorporates multi-disciplinary perspectives. More importantly, it is necessary to explore important human intellectual capacities and specific means to capture and re-use existing capacities at different level of abstraction, such as cognitive activities and technical competence.

Based on the knowledge and insight obtained from the study, the next chapter will advance to the next two questions. That is: question III: What body of knowledge, schools of thought, and good practice can be coherently used to form a sound conceptual foundation for services engineering, in support of addressing the identified challenges? Question IV: What framework or pattern should the services engineering be based on?
Chapter 3

The engineering framework and theoretical foundation

The totality of our so-called knowledge or beliefs, from the most casual matters of geography and history to the profoundest laws of atomic physics or even of pure mathematics and logic, is a man-made fabric which impinges on experience.

[Quine, 1951]

Contents

3.1 A generic engineering framework .................................. 60
  3.1.1 Engineering and problem solving .......................... 60
  3.1.2 General engineering framework .......................... 61

3.2 A theoretical foundation ............................................. 63
  3.2.1 Philosophy of mind and action ............................. 64
  3.2.2 General systems theory ..................................... 66
  3.2.3 Change and evolution ........................................ 67
  3.2.4 The philosophy of pragmatism and human experience ... 68
  3.2.5 Process philosophy, processes, and process models ... 68

3.3 Guiding principles .................................................... 70
  3.3.1 Environment-driven view ..................................... 71
  3.3.2 Multi-disciplinary view ...................................... 72
  3.3.3 Use "soft" methods instead of "hard" ones to deal with ill-understood phenomenon ........................................ 73
  3.3.4 Maximise the reuse of engineering capacities .......... 74
  3.3.5 Everything is driven by models ............................ 75
3.4 Summary
The objective of this chapter is to provide answers to research question III and IV (See Section 1.4.2). That is: III) What body of knowledge, or schools of thought, can be coherently used to form a sound theoretical foundation for services engineering, in support of addressing the identified challenges? IV) What framework or pattern should the services engineering be based on?

As analysed in the previous chapters, there is a critical demand for effective services engineering to help to derive and manage SOS. Most current practices to build SOS directly adopt traditional engineering techniques and methods, such as structural processes, object-oriented techniques, component-based software engineering, model-driven development, and aspect-oriented programming (See Section 2.3). Others, even worse, do not use any disciplined processes and methodologies, and directly build systems in an arbitrary one-off fashion based on situations at hand.

As discussed in Section 1.4.1, in search for a new approach to services engineering in socio-technical settings, it is important to take a multi-disciplinary view. Therefore, we believe it is important for a discipline of services engineering to develop a coherent multi-disciplinary theoretical foundation, based on an open environment mindset. Moreover, because of the wide range of knowledge and resources involved, a coherent engineering structure or framework is needed to organise engineering activities at various stages, according to the nature and characteristics of participating elements of SOS and their environments.

Accordingly, this chapter presents our work on a generic engineering framework and a multi-disciplinary theoretical foundation for service engineering. It is organised as follows: Section 3.1 presents the generic engineering framework we developed to structure our approach to services engineering, which is formed based on generalisation for other engineering disciplines. Section 3.2 briefly presents the multi-disciplinary theoretical foundation, which consists of complementary theories and knowledge from various disciplines such as engineering, philosophy, and management. Section 3.3 presents some important guiding principles, which are in good alignment with the theoretical foundation. These principles are derived from either empirically verified engineering practices, or logical deduction from the theoretical foundation, which provide important guidance on design and realisation of our approach to tackling the identified challenges of services engineering. Finally, Section 3.4 concludes this chapter with a summary.
3.1 A generic engineering framework

Since services engineering is a form of SISE, or more generally, engineering, it is wise to derive an approach to services engineering based on well-established and applicable engineering framework, knowledge and principles from other engineering disciplines. Indeed, Web Services, SOA, and SOC all evolved from earlier distributed computing paradigms such as networking and Web computing, object-oriented computing, and component-based systems [Bichler, March 2006]. Its fundamental concepts, such as loose-coupling, autonomy, service contract, and decomposability, come from well-known software engineering principles such as separation of concerns, abstraction and modularisation [Parnas, 1972]. Therefore, it is natural to build services engineering based on a general engineering knowledge base. In this section, we derive a general engineering framework for services engineering, based on some existing knowledge of engineering.

3.1.1 Engineering and problem solving

The main objective of any engineering discipline is to solve practical problems and meet human needs through purposeful human activities [Ackoff, 1999]. At a high level, human beings form particular world views during continuous interactions with the real-world environment. Such understanding and impression of the outside world shapes or forms specific means that generate the desired ends and satisfy the human needs, which in turn change or influence real-world situations that people are part of. This process is generalised in Figure 3.1.

![Diagram of Human engineering activities]

*Figure 3.1: Human engineering activities*

As applied sciences, disciplines of engineering pursue effective, systematic and controlled means or process to solve problems. Evidence has shown that the quality of complex systems largely depends on the processes that are applied by people and organisations to produce them [Fuggetta, 2000]. Typically, such a process involves observations of, and interactions with, the world outside themselves,
which form an overall world view consisting of various beliefs, knowledge, expertise and experience. Based on such a mental, cultural, and social foundation, specific needs of problem-solving are identified; engineering decisions are made about how to achieve such needs; practical tools, methods and processes are utilised or developed, which, eventually, generate or realise new systems or improvements to solve the identified problems.

3.1.2 General engineering framework

When seeking practical solutions to human engineering problem, the activities to perform need to be well structured to achieve the best possible outcomes. Effective engineering activities should form a coherent system to organise and coordinate inter-related elements and resources, including conscious human actions at various stages that aim to improve recognised engineering problems [Checkland, 1981]. It is therefore useful to consider an engineering framework for this purpose. As introduced in Section 1.4.2, what we mean by framework is a conceptual structure that helps to derive and organise all necessary engineering resources and activities. A framework therefore provides structural arrangement of engineering knowledge, the previous experience, processes, and other engineering resources in a coherent and systematic way, for the purpose of guiding and conducting engineering practice in pursuit of best possible results. A well-founded engineering framework provides a good starting point and sound basis for developing an approach for services engineering.

Therefore, we first establish such a generic engineering framework through generalisation of other engineering disciplines. The framework is depicted in Figure 3.2.

The generic framework is conceptually based on a theoretical foundation consisting of a coherent system of theories, scientific knowledge and engineering experience, which are shaped or influenced by personal and social beliefs, philosophies, and culture that exist among practitioners and stakeholders. Such foundation shapes and influences every aspect of human activities, either explicitly or implicitly.

Moreover, some valuable insights and best practices that are justified in the course of continuous engineering activities also form a range of concrete and explicit guiding principles. These principles, articulated, well explained and understood, are used to guide various concrete engineering activities in practice among a group of stakeholders. For instance, in software engineering, some fundamental principles, such as information hiding, modularity, and separation of concerns, are well explained and understood by practitioners and engineers, and
provide valuable guidance to engineering software systems. It is because of the application of these principles that it is possible to build and manage large scale software, which would have been impossible otherwise.

Based on the theoretical foundation and guiding principles, practical inter-related engineering activities are arranged and conducted in a particular way, such as waterfall or spiral style engineering processes, in order to achieve better productivity and product quality. These activities have their internal structure, specific purposes and overall objectives, and can be realised and supported by a range of practical engineering methodologies. Methodologies provide detailed and practical means that realise the desired engineering objectives and solve practical human problems, in a controllable, repeatable, efficient, and effective fashion. Therefore, processes and methodologies provide a means with necessary activities to do engineering, in a particular way that provides better chance of engineering success.

Furthermore, specific languages, tools, and implementation techniques are used to enhance or facilitate these methods and processes in a more efficient or effective way. In the end, desired resources, infrastructure, events, and systems are generated, managed, and updated over time, by a group of stakeholders, or pertinent people, who communicate, cooperate, or compete with each other, in order to satisfy identified engineering needs and various interests.
3.2 A theoretical foundation

As stressed in a number of places in the previous sections, (see Section 1.2.2 and 2.1.4), human beings are an important factor for any engineering activity. Engineering activities are conducted in a purposeful manner that always involves conscious human actions at some stages, which distinguishes them from the "stimuli-provoked, goal-seeking, merely purposive behaviour" of the other species [Checkland, 1981]. In a social context, these actions and the purposes they serve need to be communicated, argued over, agreed upon, and carried out among a group of stakeholders. Any purpose within a social relation cannot be attained without mutual understanding, collaboration, cooperation, and some form of competition [Habermas, 1985]. Therefore, communicative cohesion among stakeholders needs to be enhanced to facilitate communication, cooperation and competition in the social context of engineering.

Furthermore, the effectiveness of engineering activities and results is also evaluated and judged during the interactions within the process, which generates feedback through user perception and interactions, and influences the existing theoretical and practical foundation. The feedback also helps to adapt processes and methodologies when improvements are necessary, and to derive more powerful and supportive languages, tools, and technical infrastructure, which have real value in support of the engineering processes.

Collectively, the theoretical foundation and guiding principles, cognitive and behavioural engineering activities and processes performed by various stakeholders, and corresponding supporting resources and environments, work as a whole within a coherent engineering framework.

In summary, a general framework of engineering consists of coherently aggregated foundation, guiding principles, process, methods, languages, tools, systems under consideration, and people involved, with evaluation and feedback processes incorporated. Such a framework provides a coherent structure to organise and capture necessary engineering resources and activities, which facilitate the processes to engineer systems.

3.2 A theoretical foundation

Based on the general framework presented in Section 3.1.2, we gradually designed an engineering framework for services engineerinthege, called Organic Aggregation Service Engineering Framework (OASEF). This section presents its theoretical foundation, on which all services engineering elements and resources are based. As discussed in Section 1.4.1 and 2.1.4, the socio-technical nature of SIS requires multi-disciplinary knowledge to guide and conduct effective engineering. There-
Chapter 3: The engineering framework and theoretical foundation

fore, this foundation consists of coherent aggregations of knowledge and principles from various disciplines, such as general systems theory, complexity theory, and modern philosophy of mind and knowledge. Of course, we do not intend to refer to the whole body of knowledge in these areas, which is both impossible and unnecessary. Moreover, this section will introduce some core concepts of relevant knowledge and theories in a brief manner, which set the stage for presentation of OASEF in the following chapters. This knowledge provides a theoretical foundation, and will be further expanded in individual chapters.

3.2.1 Philosophy of mind and action

As discussed in Section 2.1.4, the social and culture aspects of human perspectives of STS are beyond the scope of this research. Instead, this research focuses on human mind and human intellectual efforts in the engineering context. It is important to study the nature and characteristics of human mind, in order to understand and improve the way it works for engineering problem-solving. We particularly focus on the philosophy of mind, provides insight on human mind and consciousness, a branch of modern philosophy that studies the human mind and consciousness.

Human beings develop and present complex mental states and profound self-consciousness while observing, reasoning, planning, and manipulating the world outside ourselves [Davidson, 1980]. For example, humankind has the ability to identify rational purposes, and to intentionally act towards them, while presenting free will in the mean time. There are therefore complex relationships between mind and actions. Some modern philosophers hold that certain causal connections exist between mind and action [Sloman, 2009, Kim, 2000]. That is, specific states of mind will cause certain behaviour.

The concept of mind, its nature, order, and mechanism have been unclear and confusing in human history. Human mind has been a highly controversial subject since ancient philosophers started to inquire human beings' position in the universe thousands of years ago. There are various schools of thought in this respect. Starting from some ancient astronomer such as Nicolaus Copernicus and Johannes Kepler, physicalism holds that mental states are non-essential secondary qualities that are causally produced by physical entities with primary qualities, and therefore can be scientific studied based on the empirical measurements [Crittenden, 2009]. To an extreme, Eliminative materialism presents a thesis that uses complete neuroscience to explain psychological phenomena, and denies and eventually displaces any mental conception and theories, and their usage [Churchland, 1990]. On the contrary, some researchers believe that conscious
perception and human intentionality are fundamental and primary in our world, and are in fact presupposed by any objective knowledge [Whitehead, 1978]. That is, human knowledge intrinsically involve conscious perception and subjectiveness. Moreover, some important aspect of mind, such as psychological characteristics, cannot be causally determined and objectively observed [Crittenden, 2009, 147]. For example, it is difficult to objectively measure emotion, free will, and aesthetic and religious aspects of human mind. The value of these various schools of thought in an engineering context is to recognise the causal, observable, and measurable aspects of human mind, while acknowledging the difficult psychological aspects. Such understanding helps to determine the focus and scope of human aspects of STE (see Section 2.1.4).

It makes sense to acknowledge the difficulty to observe and understand human mind due to its complexity and uncertainty. To get around this problem, some aspects in close relation to mind are studied instead, since they are relatively easier to observe and understand. For example, Behaviorism studies mind via analysis of the observable external behaviour and their causal patterns. Functionalism theory introduces internal processes that consist of functional causal sequences, which link initial causal events inside the individual with external behavioural events [Fodor, 1985]. A variation of this thought focuses on information processing function of mind, which has two essential aspects that “must be stored, or represented, in memory”, and “be used, or processed, to perform cognitive tasks”, namely representation, and process of knowledge [Glass and Holyoak, 1988]. In other words, the mental processes are regarded as causal sequences of manipulation of certain representations of mind. Although this school of thought is criticised by some researchers because of its inadequacy to explain “the nature of some mental phenomena such as experience, qualia, sensory processes, mental images, visual and auditory imagination, sensory memory, perceptual pattern-recognition capacities, dreaming, hallucinating, etc.” [Aydede, 1999], it provides one way to visualise the working mechanisms of mind, which helps to elicit, capture, and utilise human mind, in the context of engineering, if we do not consider ill-understood psychological aspects such as emotion.

OASEF, as an engineering framework, is more influenced by the Physicalism and Functionalism school of thought, in the sense that human mind, at least pertinent part, can be observed, understood, and managed in a systematic, predictable, repeatable, and controllable way. That is, we will not consider the unpredictable and uncontrolable part of human minds such as emotion and religious believes within our scope. Moreover, the hard problems with regards to the origin of mind and its relationship to body, the controversial classic philosophical problems, is not considered essential in our engineering context.
Instead, the nature, representation, and manipulation of some, if not all, important mental phenomenon will be analysed, captured, and utilised to assist engineering activities.

3.2.2 General systems theory

Large and complex SIS and SOS involve dynamic and complex socio-technical settings. For example, SOS often comprise a large network of interacting and co-evolving services which are provided by, or consumed by, various hardware, software, and people. It is difficult to understand and manage them from a single perspective, or without taking the whole environment into consideration.

A radically different mindset of engineering driven by STE is necessary to deal with the broad system scope. The best way to understand complex systems is to take an outside-in view of systems that focuses on the understanding of their environment and their interactions [Ackoff, 1999, Checkland, 1981]. Practice has shown that ignorance of system wholeness and environments is the cause of many difficult problems, such as a lack of integration fitness and slow response to change [Checkland, 1981].

Checkland uses the term systems thinking to describe the outside-in system driven view. Essentially, a system is an organic whole that has irreducible "emergent properties" on certain conditions [Checkland, 1981]. Viewing everything from the system's perspective helps to understand things in terms of purpose, value, or anything that matters. Such understanding in turn helps to identify the most important problems and corresponding improvements. To help systems thinkers to observe, understand and solve systems problems, Checkland defines five categories of systems: natural, designed physical, designed abstract, human activity, and transcendental systems.

According to Checkland's classification of systems, a service, in its broadest sense, is not necessarily a software entity, but it can be a natural system such as available physical resources, designed physical system such as hardware, designed abstract system such as software, human activity system in a social context such as a business process, or a transcendental system that is beyond current public knowledge of human beings. It can also be a combination of them. Therefore, services engineering can be conducted in a wider scope to build SOS including not only software systems, but also systems consisting of natural, designed physical and abstract systems, as well as human activities involving complex STE.
3.2.3 Change and evolution

As introduced in Section 1.1.1, our world is rapidly changing. Change is a fundamental phenomenon that has been widely observed and admitted since the ancient time. Greek philosopher Heraclitus holds that the world is essentially a naturally changing process, which produces everything in this universe [Aurobindo, 1941, Smith, 1966]. According to him, nothing would be the same at two points in time because of the occurring change, which is eternal and rules everything. Moreover, change is the fundamental principle to understand and explain things in the world. Philosophy that focuses on process regards change as a principle law that is more fundamental than things or substance [Aurobindo, 1941]. Even theories that focus on entity or substance, as opposed to processes, also acknowledge change in terms of the nature and relations of entity. Modern sciences and philosophy also recognise the significance of change. Process philosophy holds that reality is all about changing processes that not only produce substance, but also cause or explain all other phenomena [Whitehead, 1978]. Therefore, “change of every sort is the pervasive and predominant feature of the real” [Whitehead, 1978]. Moreover, according to modern physics such as quantum theory, matter, in a micro sense, is not constituted of particles, but collections of very small changing processes showing statistical regularities [Stapp, 1982].

Evolution is a process that reflects the pervasive accumulation of change, existing in organisms, the human mind, and the universe. Whitehead [1997] states that “nature is a structure of evolving processes. The reality is the process”. Evolution is caused by either intrinsic or extrinsic force towards something new or positive, such as natural biological selection, or rationally-guided selection based on value-driven purposefulness [Whitehead, 1997].

In rapidly evolving environments, it is essential to have the ability to identify or predict change, make sense of it, decide desired ends, and to plan and act accordingly. As described in Section 2.2, services engineering needs to be conducted in uncertain and volatile environments. Such a nature of service dynamics and volatility requires agile responses to change during the development and maintenance of systems. Sufficient support for observing and reacting to change in a timely fashion is therefore necessary. Services are designed to be used in a specific situation. If the situation changes, the effectiveness of the service could change. Therefore, effective mechanisms should be provided to ensure the system to stay resilient to malfunction and inconsistencies while change occurs, and to survive the potential chaotic situations. OASEF acknowledges change and evolution in many aspects of services engineering and provides support for accommodating the need for a change.
3.2.4 The philosophy of pragmatism and human experience

While we observe and understand the human mind and system environments, there are some important questions we need to ask: why are they useful, and in what sense? How do we evaluate their effectiveness? We explore answers to these question from a pragmatic point of view, due to our engineering scope, that is, to seek practical and workable solutions for engineering problems.

According to Charles Sanders Peirce, a well-known American philosopher and scientist, the meaning of any conception is related to its practical consequences, in other words, the effect of action [Peirce, 1905]. This view, similar to that of Einstein, builds a tight connection between the nature and meaning of a thing and its effect that can be measured by an exact method [Apel, 1981]. Therefore, the meaning and the value of any conception and related intellectual assets depends on the behaviour it actually gives rise to. Such value is empirically observable and testable in specific environment, under specifiable conditions. In other words, knowledge has no real meaning unless it produces observable impact, consequences, or influence.

From a pragmatic point of view, the actual state of the achieved ends and the effectiveness of systems should play a crucial role in any discipline of engineering, including services engineering. That is, services engineering needs to value practical achievements, or the effect of services and systems that are engineered, above all else. Unlike science, which is concerned more with exploring new knowledge and its validity in answering questions, we should also look into actions that lead to a defined end, and its success in a practical sense [Checkland, 1981]. Based on such an empirical philosophy, modern sciences treat any subject matter as objects that can be explained causally and measured in a quantitative, repeatable means according to their effects within their environments. We take this view to value human intellectual capacities and other engineering capabilities, that is, whatever produces useful consequences in current or future situations is of value and should be captured and made available. This is the fundamental basis of our reuse mechanism that will be introduced in Section 5.4.

3.2.5 Process philosophy, processes, and process models

As generalised in Section 3.1.2, an engineering framework needs a process to organise and coordinate engineering activities and resources. It refers to an important notion of process. This section details what a process means and how it contributes to engineering in a pragmatic sense.

The origin of a notion of process can be traced back to ancient Greek
3.2 A theoretical foundation

philosopher Heraclitus, who holds that the essence of the world is not material substance of things but rather a continuous natural process, or "flux" in his term, which is driven by ever-presence confliction and competing forces [Aurobindo, 1941]. Whitehead defined process as "a sequentially structured sequence of successive stages or phases", which consists of temporal elements that also has a coherent structure of certain shape or format [Whitehead, 1978]. According to him, pervasive processes are the origin and sustaining force of the nature, and, in fact, what reality really is, and is therefore fundamental for the understanding of the nature. In an engineering context, Fuggetta [2000] defines a process as "the coherent set of policies, organisational structures, technologies, procedures, and artefacts that are needed to conceive, develop, deploy, and maintain a software product".

Traditional software engineering took a reductive approach to understand and analyse process includes various involved activities, participants and roles, conceptual principles and guidelines, and supporting tools. As a result, processes often involve a concept of life-cycle that divides the lifetime of systems to be engineered into various temporally separated stages [Fuggetta, 2000]. According to Fuggetta, in the context of software engineering, a process is "the coherent set of policies, organisational structures, technologies, procedures, and artefacts that are needed to conceive, develop, deploy, and maintain a software product" [Fuggetta, 2000].

Royce's influential waterfall model [Royce, 1970] is one of such reductive approaches to engineering processes, which consists of a sequence of activities such as requirements analysis, design, and implementation. After years of practice on complex engineering projects, this approach started to reveal various shortcomings, and has gradually evolved into more iterative, incremental, and evolutionary models, sometimes called Iterative and Incremental Development (IID) [Larman and Basili, 2003], such as the spiral model [Boehm, 1986], rapid prototyping [Floyd, 1984], the Rational Unified Process (RUP) [Gornik, 2004], and more recently, agile methodology [Highsmith and Cockburn, 2001].

There are also other general processes of human problem-solving such as the doxastic-conative loop which consists of environment, beliefs, evaluate the world, make plans and activity [Pollock, 2006]. Another example is Boyd's Observation, Orientation, Decision, and Action (OODA) loop [Boyd, 1986], which provides a conceptual model for human decision-making to cope with highly dynamic and competitive environments. These process models provide a general framework to guide human activities based on observation of environment, conscious sense-making and rational decision-making. OODA model also emphasises agility in order to deal with dynamic change and obtain competitive advantages. It requires
Chapter 3: The engineering framework and theoretical foundation

quicker cycling within its process loop, and, in the meantime, promotes any means to weaken or destroy opponents’ abilities to complete their loops at a faster pace. OODA model has been widely applied in critical and dynamic environments such as military battles and business competition.

Improvements on processes is a critical success factor for software engineering. This is evidenced by plenty of work. Shewhart and Deming [1986] introduces a Plan-Do-Study-Act (PDSA) cycles to develop systems that allows complexity to be understood and handled in an iterative fashion, which is supported by and further developed by some researchers [Gilb, 1977]. More recently, Capability Maturity Model (CMM) has become popular in software industry, which provides a general evaluation framework to determine the quality of a software development processes [Humphrey, 1990]. Evidence shows that improvements on maturity level of processes help to increase engineering productivity, and to manage system complexity [Clark, 2000]. Therefore, to address the problem introduced in Section 1.2.1, our approach promotes a focus on suitable processes in the context of services engineering [Wang and Taylor, 2008, Wang, 2010b].

We believe that existing knowledge about processes and sufficient adaptation to change is the key to tackle the challenge of uncertainty and volatility. A change-oriented mindset is required to adapt the processes and the systems they produce, in a timely fashion while change occurs at various paces. As discussed in Section 1.2.1, we need to synthesise and adapt existing processes to integrate both technical capacities and user needs into a unified whole, and to facilitate services engineering and SISE in a way people want them to be.

3.3 Guiding principles

Aiming at addressing the challenges discussed in Section 2.2, a number of more specific guiding principles are derived, based on the OASEF theoretical foundation and generalisation from engineering practices. These principles work as a linkage between a more abstract and general foundation, and more concrete and specific means, in support of achieving our ultimate objective of services engineering. These principles, collectively, provide useful guidance to design and implement other parts of the OASEF framework, which will be described in the remainder of this section.
3.3 Guiding principles

3.3.1 Environment-driven view

Based on a system view described in Section 3.2.2, SOS can be dealt with from an environment-driven perspective. A different mindset of services engineering can be formed with a focus on emergent system properties and environments. It helps to address the grand challenges in association with the dynamic and open system environments. For environment-driven services engineering, the focus on implementation techniques needs to be shifted towards systems' emergent characteristics and their environments. A focus on implementation often leads to development activities driven by solutions, technologies, industry standards, and their providers. This is because of a lack of clear understanding of their complex environmental nature and constraints such as critical problems, required improvements, and environmental constraints. As a result, ineffective systems are often produced that barely fit into their environment and fail to survive in the changing environments. Contrarily, an environment-driven view leads to user-driven, demand-driven, problem-driven approaches that study the environment upfront before new systems are specified and built. By understanding the environment upfront, people can obtain better understanding and specifications of real-world needs and constraints, which is widely accepted as an essential factor to ensure a success of complex software systems [Jackson and Zave, 1995, Zave and Jackson, 1997].

Taking such a view, in the context of SOS, a service should be regarded as an organic whole that has its own irreducible properties, such as end-to-end Quality-Of-Services (QoS): for example, the overall reliability, availability, security and performance. This system view of services is a necessary complement to a de-compositional or reductive view that focuses on individual services. A lack of such view often leads to the focus on the internal implementation detail of autonomous individual services, and on the mechanical composing logic or work-flow process, which is a fundamental problem. This is because although the decomposition approach can make the system complexity manageable within the well-defined and individual context by aggregating smaller and composable services to address specific parts of a bigger problem [Erl, 2005], it also eliminates some irreducible important factors such as human needs, wider environment, and dynamics. As a consequence, services and systems are often unable to produce meaningful value in a network of services and their surrounding environments.

Moreover, in the context of services engineering, clear understanding of service environments enables a meaningful and sensible service composition either at design time or at run-time. Effective development of service compositions needs not only agreement between service providers and consumers regarding the semantics
of services and their usage, but also formally captured reusable information about environments of atomic and composite services. The latter information is essential because it affects the way services are understood in terms of their behaviour and characteristics, how they can be discovered, invoked, communicated and coordinated, and therefore determines a meaningful and sensible composition. Services have diverse, agile and sometimes uncertain characteristics and must fit into their environment and satisfy dynamic and often conflicting requirements from wide range of stakeholders. In such a situation, although specific internal structures and behaviour need to be understood, a system view is also important to obtain a bigger picture containing various environmental constraints and competing forces. Such a bigger picture involves both internal and external interactions and constraints, and would help to understand what systems are needed, and why they are needed. This environmental understanding is crucial when SOS are complex and dynamic, without which, sensible and meaningful service composition and systems provision are impossible to be achieved [Wang and Taylor, 2008]. Therefore, in the context of SOS, we need to understand not only the internal structure and behaviour of individual services, and their inter-relationships, but also their external emergent properties and functions as a whole, and their socio-technical environments.

The environment view of services has another significance. It leads to the recognition of a lack of business and human perspectives in services engineering. Although current advances in technologies such as SOA and SOC alleviate the impediment of services engineering from a technology point of view, the importance of understanding business and human needs and requirements has not been effectively supported in practice. As emphasised by several researchers [Papazoglou et al., 2006, Benatallah et al., 2006], effective requirements analysis, such as business process modelling, needs to be better supported during the engineering processes of large-scale SOS.

### 3.3.2 Multi-disciplinary view

As introduced in Section 1.4.1, in order to derive effective services engineering, a multi-disciplinary approach is a must. In fact, due to the nature of services engineering, it is an important principle to take multi-disciplinary approaches to understand and conduct engineering activities. In Section 2.1.4, we discussed the socio-technical factor of today's SIS, which are built and operated in complex environments that consist of hardware, software, people, processes and regulations. Addressing critical issues of engineering in such socio-technical settings requires syntheses of multi-disciplinary knowledge from science, engineering, as well as
social sciences such as philosophy, psychology, economics, and politics. Moreover, the variety, quantity, and relationship of the elements within SIS dynamically vary in different dimensions, such as technology, psychology, business, and culture. They cannot be understood by studying a single discipline in isolation. Insufficient understanding and analysis of systems and their environments in a broader scope would lead to ineffective overall systems that cannot fit into their surroundings.

In the context of services engineering, complex SOS need to be built and managed in a way that takes a wider range of factors into consideration. It is necessary to consider the views and expertise of all applicable disciplines, including science, engineering, and social sciences, in order to ensure that complex environments and problem situations, along with the likely impact of proposed improvements, are well understood. Rai and Sambamurthy holds similar view by advocating that an interdisciplinary approach spanning areas such as technology, engineering, human behaviours, organisational strategy and design, economics and management is required to understand, design, and manage services [Rai and Sambamurthy, 2006].

3.3.3 Use “soft” methods instead of “hard” ones to deal with ill-understood phenomenon

As discussed in Section 2.3, some existing approaches to services engineering directly adopt traditional “hard” engineering methodologies, which consists of rigid and isolated activities, such as waterfall sequence of requirements analysis, design, implementation, verification and deployment. This approach is problematic due to the nature of SOS in their complex socio-technical environments.

First, service requirements are often impossible to clearly elicit and specify in advance, since services are developed and managed without knowing the consumers’ exact needs beforehand. On the contrary, the consumers dynamically locate and invoke a service after it has been developed and published, on the condition that they obtain certain degree of confidence that the service satisfies their specific needs and constraints. A perfect matching between provided service capabilities and ad hoc user needs is hard to achieve [Narendra et al., 2007].

Second, clear and complete requirements are often hard to achieve in practice. In the context of SOC, services are used to directly represent any human activities, the most complex type of systems that consists of interactions and operations involving men and machines. These activities are often of great complexity, hard to identify and understand, intangible or full of uncertainties, and dynamically evolving. For instance, services are often used to capture and represent business processes of an organisation. Step-by-step purposeful procedures are often complex
and hard to clearly define to achieve business objectives [Havey, 2005]. Service modelling and implementation in areas such as economics and politics would be even harder since they are not well understood. For such complex human activity systems with “ill-structured problems”, traditional “hard” system methods are often infeasible or ineffective [Checkland, 1981].

Third, it is often practically impossible to define engineering process in a clear-cutting fashion and to bring together every stakeholder during each engineering stage to fulfil well-defined objectives. This is determined by the nature of agility and uncertainty at every engineering stage, as well as the need of collaboration in a wider scope.

Fourth, SOS often reveal more non-technology characteristics since by definition they involve various social values in the course of service interactions. Social aspects, such as perception, trust, reputation, and experience, add another dimension to system dynamics. Experience tells us that “Hard” engineering methodologies fail to address this challenge [Checkland, 1981].

Therefore, “soft” system methodologies with fuzzy desirable goals and less rigid process are often effective when dealing with unstructured and obscure “soft” problems [Checkland, 1981], such as human activity systems represented by services. Such methodologies are often user-centric and draw attention to social aspect in addition to traditional technology-related aspects [Ross and Schoman, 1977, Bennett et al., 2000, Goguen, 1996]. In a highly uncertain a dynamic environment, such as the Internet, a more flexible development process is needed that obtains earlier feedback on performance, and allows late change during design and implementation [MacCormack et al., 2001].

### 3.3.4 Maximise the reuse of engineering capacities

Human beings continuously make improvements to themselves and their environments, by developing cognitive capacities and aggregating useful knowledge and practical mechanisms. Human knowledge and intellectual capacities are therefore crucial factors that ensure successful engineering. They represent the most important engineering resources to solve complex problems, and are only possessed by human beings, often those with specialised competence and talent. Therefore, valuable intellectual engineering efforts and achievements should be captured and reused as much as possible. These achievements include justified perception of outside world, understanding of problem situations, rationalised desired ends, successful solutions to certain type of problems, or effective means towards a desired end. Evidence shows that increases on reuse of both higher order cognitive achievements and lower level implementation techniques significantly
3.3 Guiding principles

improve system productivity, flexibility, and extensibility [Thramboulidis, 2005].

Therefore, reuse of both higher order human intellectual capacities and lower level technical competence can help to manage the increased complexity, and greatly contribute to responsive system evolution and adaptation. It therefore improves engineering productivity and facilitates agility in the uncertain, volatile, and tense situations where modern SIS and SOS often reside in.

Moreover, necessary knowledge and practical experience that enables optimal decision-making is an important intellectual assets of business, and needs to be captured, stored, and re-used systematically and effectively. As Gil et al. [2007] point out, it is often 'impractical or even impossible' to reproduce and share information and processes within a complex environment where fragmented information exists in various forms and locations. Likewise, in the context of SISE and services engineering, where agile decision-making is important, the knowledge and experience to assist decision-making is captured in a form either in people's minds as common sense, expertise, and experience, or in various concrete forms such as documents, emails, web pages, regulations and laws, and cannot be easily re-accessed and re-used [Wang and Taylor, 2008].

Therefore, in order to address the great challenges of SISE and services engineering, it is critical to form and grow organic aggregations of human intellectual capacities and lower level technical competence, and to promote the reuse of relevant capacities and knowledge to facilitate all engineering activities.

3.3.5 Everything is driven by models

As defined in Section 2.1.3, models and modelling activities provide a critical means to understand complex problems and derive practical solutions. According to Bézivin [2005], the most fundamental principle of MDE is "everything is a model". In fact, every human activity is associated with certain form of modelling, either explicitly or implicitly [Bézivin, 2005]. Without models and modelling, complex problems and technical complexities would overwhelm humankind due to human cognitive limitation.

Therefore, we apply a principle called everything is driven by models. That is, models and modelling should become the main driving force of any SISE activities. This is because of two reasons: First, models, at various level of abstraction, help human beings to focus on particular aspects of problems of original systems, and to obtain better understanding of their nature and optimal solutions that can produce predicted improvements. Second, modelling provides a means to promote two fundamental software engineering principles: higher level of abstraction and
Chapter 3: The engineering framework and theoretical foundation

separation of concerns. It thus helps to tackle the problems of complexity and volatility, which is evidenced in MDE practice as described in Section 2.1.3. That is, conforming to these two principles helps to reduce the conceptual complexity, and lead to simple architecture, rapid development, high scalability, and good flexibility [Bézivin et al., 2005].

Therefore, in order to deal with the challenges caused by high levels of complexity, uncertainty, and volatility, it is critical to fully incorporate an effective model-driven approach to SISE. That is, various models, from different perspectives, and at different levels of abstraction, should be used not only to understand environments, problems and desired improvements, but also to specify and communicate engineering intentions, architectural and detailed design of solutions, and to implement and verify concrete systems in reality. This principle should be fostered by concrete modelling techniques and supporting tools, which should facilitate and boost model manipulation and transformation processes, in a systematic, and, if possible, an automatic way.

Such a principle has the following pragmatic value: First, it promotes a focus on higher level modelling activities. Such a focus allows developers and users to concentrate on higher order knowledge and abstraction, instead of lower level of detail such as implementation technologies [Wang and Taylor, 2008]. Second, modelling automation and model transformation helps to capture lower level technical implementation capacities, and to systematically derive systems based on high-level models. Therefore, a model-driven approach would shield engineers from the underlying technological complexity, and enable a focus on the essential complexity. As a result, an effective model-driven approach would allow developers and users to utilise pertinent domain knowledge to understand problems and the need for improvements, as well as the detailed design of such improvements, instead of spending most of the time working on lower level implementation technologies such as Java EE, .NET, or CORBA [Wang and Taylor, 2008].

Therefore, MDE promotes a focus on higher level of abstraction and effective separation of concern, the fundamental software engineering principles to deal with complexity and facilitate flexibility and extensibility. The benefit of the higher level of abstraction is better understanding of the systems and their environments, and better design of necessary improvements to the systems. The separation of concerns also enables people to use limited intellectual efforts to tackle more essential domain complexities, rather than tedious and error-prone mechanical process.
3.4 Summary

In summary, this chapter provides answers to research question III and IV, that is: III) what body of knowledge, schools of thought, and good practice can be coherently used to form a sound conceptual foundation for services engineering, in support of addressing the identified challenges? IV) What framework or pattern should the services engineering be based on?

We first developed a generic engineering framework based on generalisation of other engineering disciplines. It consists of a coherently aggregated foundation, a synthesis of a range of guiding principles, an engineering process, an integrated method in support of such a process, supporting languages and tools to realise the engineering method and process, people who involved in the engineering processes, and the system which interacts with its surrounding environment. There is also an evaluation and feedback process in relation to each constitutive elements of the framework.

Our approach to services engineering is structured based on this generic framework. It has a theoretical foundation that consists of a coherent set of multi-disciplinary knowledge, such as philosophy of mind and action, general systems theory, the theory of evolution, the philosophy of pragmatism, and process philosophy. It also includes some practical guiding principles that are either generalised from best practice of software engineering, or deduced from supporting theories. These principles are: Environment-driven view (See Section 3.3.1), Multi-disciplinary view (See Section 3.3.2), Use “soft” methods instead of “hard” ones to deal with ill-understood phenomenon (See Section 3.3.3), Maximise the reuse of engineering capacities (See Section 3.3.4), and Everything is driven by models (See Section 3.3.5). These principles provide important guidance on design and realisation of our approach to services engineering.

Based on the generic engineering framework and a solid theoretical foundation, some fundamental challenges of services engineering will be tackled in the following chapters. The next chapter will provide our answer to research question V (See Section 1.4.2): What conceptual and practical mechanism do we use to address each identified issue?. This chapter will focus on the first part of this question, by presenting a generic engineering process for general SIS in open STE.
A New Conceptual Process Model

Nature is a structure of evolving processes. The reality is the process.

[Whitehead, 1997]

Contents

4.1 A new process model of services engineering ............... 81
   4.1.1 Process models for services engineering ............... 83
   4.1.2 A notion of Organic Aggregation ..................... 84
4.2 A unified model of reality and mind at the fundamental level 87
   4.2.1 Mind and reality .................................... 87
   4.2.2 Linking mind with reality ........................... 89
4.3 Concepts at the first level ................................. 92
   4.3.1 Perception ........................................ 92
   4.3.2 Conception ....................................... 93
   4.3.3 Action ........................................... 95
   4.3.4 Summary ......................................... 96
4.4 Concepts at the second level ............................... 97
   4.4.1 Sensation ......................................... 98
   4.4.2 Abstraction ....................................... 99
   4.4.3 Rationalisation ................................... 101
   4.4.4 Realisation ....................................... 103
   4.4.5 Summary ......................................... 104
4.5 Concepts at the third level ................................. 105
Chapter 4: A New Conceptual Process Model

4.5.1 Inner structure of abstraction ........................................ 105
4.5.2 Inner structure of rationalisation ................................... 107
4.6 Summary ................................................................. 111
4.1 A new process model of services engineering

This chapter presents our effort to answer research question V, that is, *What conceptual and practical mechanism do we use to address each identified issue?*. For now, we will focus on the first part, a conceptual mechanism to deal with the challenges of SIS. In particular, we present a novel process model to derive modern SIS in their open environments, the Socio-Technical Environments (STE).

As described in Section 2.4, our vision of services engineering involves a focus on the human aspects of engineering. Any engineering process involves the interconnected world of human beings and the environments in which they reside. We aim to derive a process that promotes a focus on human perspective during the engineering activities. In particular, such a process provides support for understanding and conducting engineering activities that intensively involve human mind, such as understanding how human beings recognise problems and solve them, and improving such capacities.

In this chapter, we present our novel conceptual process model for general SISE, which is called Organic Aggregation Process (OAP) [Wang, 2010a]. Its main concepts, structure, and interacting mechanism are presented in the remainder of this chapter, which is organised as follows. Section 4.1 analyses the demand for a new process model and our vision for such a model, in the context of services engineering framework introduced in Chapter 3. In order to achieve such a vision, we developed a fundamental concept called *organic aggregation*, which is introduced in Section 4.1.2. Based on the concept of *organic aggregation* and OASEF foundation introduced in the previous chapter, a process model called OAP is designed, which organises engineering activities in a hierarchical and flexible way. In Section 4.2, we discuss and define our notions of *mind* and *reality*, at a fundamental level. Section 4.3 presents the nature and characteristics of inter-connected and iterative engineering activities, at the first level, in the world of *mind* and *reality*. At this level, OAP consists of *perception*, *conception*, and *action*, which are explained and defined in this section. Section 4.4 and Section 4.5 present OAP concepts in more detail in terms of their functions, scopes and inter-relationships. Finally, Section 4.6 provides a summary of the main content of this chapter.

4.1 A new process model of services engineering

As discussed in Section 3.2.5, effective processes have a crucial role in engineering. However, existing engineering processes are inadequate to deal with grand challenges involved in engineering of open systems [Wang, 2010a]. For example, the OODA model and its followers, in general, are too abstract and coarse-grained.
They lack some fundamental elements in the context of engineering, such as clearly defined activities at detailed levels, clarifications of boundaries and relationships, and supporting tools, according to Fuggetta [2000]'s structure of process (see Section 3.2.5). The content and boundaries of activities in each phase are vague and ill-defined. Moreover, in a concrete sense, neither guidelines about internal structures and content, nor representational formalisms for each phase are provided. Therefore, in an engineering context, the application of OODA is a challenging task [Wang and Taylor, 2008].

More importantly, after decades of evolution, these process models are fundamentally based on the same reductive concept. That is, starting from an initial scope, a process comprises pre-defined decomposition and refinement of life-cycle, at fixed levels, and with rigid dependencies, which we refer to as Rigid Reductive Processes (RRP). As introduced in Section 3.2.5, most IID processes adopt rigid pre-defined phases such as Analysis, Design, Implementation, and Test, although sometimes being accelerated, or with dynamic feedback introduced in various phases [Larman and Basili, 2003]. Similarly, RUP comprises loops of Inception, Elaboration, Construction and Transition [Gornik, 2004], whereas agile methodologies usually consist of Exploration, Planning, Iteration and Release [Beck and Andres, 2004]. So does the OODA model and its followers.

These RRP style processes, although sometimes effective in complex and safety-critical systems, suffer from limitations caused by their confining scope, as well as rigid and closed structures. For example, many rigid activities in IID processes are time-consuming to carry out in the first place, or are difficult to change over time [Shapiro, 1997]. Although agile methodologies aim to address this issue by using a lightweight process with short and iterative development cycles, quick feedback, and more flexible human involvement [Highsmith and Cockburn, 2001], these approaches often lack order and cohesion in complex situations, and therefore are usually used only for small and medium sized projects [Shapiro, 1997].

In the context of services engineering, there is also a critical lack of appropriate processes, which greatly hampers provision and engineering of SOS in their complex and dynamic environments (see Section 1.3). In the following section, we will analyse the problems involved in services engineering processes, and present a different vision of processes within the framework of OASEF, based on the theoretical foundation described in Chapter 3.
4.1 A new process model of services engineering

4.1.1 Process models for services engineering

In Section 1.3, we briefly introduced a demand for a new methods of services engineering. Such a demand is justified by current practice of services engineering as discussed in Section 2.3. There is currently a strong focus on specific technologies and techniques. These approaches often involve intensive programming and implementation activities, performed by a team of developers with necessary technological skills and expertises, such as business process modelling, service implementation and composition, and runtime management. However, as we analysed earlier, such a strong focus on, and tight coupling with, technologies and implementation is fundamentally problematic, due to various reasons (See Section 2.3).

Therefore, we argue that there is a lack of focus on non-technical aspects during services engineering processes. The understanding and specifications of human problems and desired ends are either lacking, inadequate, or merely used as documentation for reference only, and cannot be used to guide and derive other process activities in a synchronised manner. There is either a lack of high-level constructs for better expression and communication of the problems, or a notational and semantic gap between technical specifications of problems and human understanding undesired situations. Moreover, the knowledge and intellectual efforts that are used to understand the problems and involving situations, and to create sensible designs and plans, are often absent in the first place, or get discarded when they become obsolete while situations and technologies evolve [Wang and Taylor, 2008]. The inadequate understanding and modelling of the higher level problems inevitably leads to ineffective systems that are unable to solve practical problems and to satisfy desired needs, which is also a fundamental problem in modern software development process [Gibbs, 1994]. Therefore, it is insufficient to exclusively rely on existing technology-focused approaches, such as SOA, CASE tools, and component-based architectures.

Instead, in the broad socio-technical context of SOS, a radically different engineering process is necessary. Such a process needs to raise the level of abstraction in terms of the focus of engineering processes, and to promote loosely-coupled activities that can be separately captured and re-used [Pastor and Molina, 2007]. As presented in Section 2.4, our vision of service engineering should apply coherent processes to enable and support activities to explore and understand high level human intellectual engineering capacities, and to derive and utilise optimal technical solutions, for the purpose of engineering SOS that fit their complex and volatile STE. On the one hand, we need to promote higher level intelligent achievements, such as accurate understanding of complex problem situations, and
justifiable desired ends, on the other hand technical activities should be captured and automated to build usable and dependable systems in a rapid and reliable fashion.

We believe such an approach is able to attack the challenges of services engineering described in Section 2.2. This is because it promotes higher level of abstraction in areas of human mind and intellectual capacities. In addition, clear separation of concern between human minds and actions would help to reduce technological dependencies and therefore improve engineering productivity. These two well known software engineering principles, higher level of abstraction and separation of concerns, have been approved to be effective in attacking software challenge of complexity and agility, and are applied by many modern software engineering approaches, such as MDE (see Section 2.1.3).

In support of this vision, we develop a important notion called Organic Aggregation, which is fundamentally based on the theoretical foundation and guiding principles discussed in Section 3.2 and 3.3, especially an environment-driven holistic view of system and acknowledgement of continuous change and evolution. The next section describe such a notion in detail.

4.1.2 A notion of Organic Aggregation

An important notion called Organic Aggregation is developed for the purpose of attacking the grand challenges of modern SIS, such as a high degree of complexity, uncertainty, and volatility as described in Section 2.2. By aggregation, we mean a synthetic, as opposed to reductive, approach to look at the world that is aggregated by interrelated elements [Checkland, 1981]. Being organic means being coherent, ordered, logical, harmonious, or sensible, that is, all characteristics that a healthy organism possesses in order to fit into its environment.

This notion is wider in scope compared with agile software engineering, a widespread trend in current practice of SIS. We argue that being organic is far more sophisticated than being agile. The former requires all necessary efforts to enable a system to best fit into its environment that varies in different situations. An organic system therefore presents specific characteristics in its particular situation. On the contrary, current practice of agile methodologies often neglect many other important environmental factors such as rationality, priority, and human satisfaction. Existing agile practice often focuses on rapid iterative cycles and flexibility in terms of processes and products with all stakeholders involved in the engineering life cycles [Shore and Warden, 2007]. Therefore, we believe that in addition to being agile, we also need to address any other environmental dependencies and constraints, in order to achieve an organic system.
Organic Aggregation therefore is a synthetic approach to grow fit systems. It is defined as:

Organic Aggregation is creative syntheses of any form of relevant existence of resources, such as objects, events, processes, knowledge and human experience, in a coherent, sensible, and harmonious way, for the purpose of forming systems that best fit their environments.

In volatile and uncertain environments of SOS and modern SIS, Organic Aggregation requires sufficient flexibility and agility, in order to enable systems to efficiently change and evolve toward a new favourable order from the emerging chaos, at a pace faster than the changing rate of chaotic situations. More importantly, it also requires overall environmental fitness of SOS, especially aspects related to human beings, such as human perceived problems and desired ends.

At the process level, the concept of Organic Aggregation is applied to not only the structure and inter-relationships of processes, but also capabilities to carry on composing sub-processes and subordinate activities, and the products they produce. That is, a process is organic aggregations of not only inter-related internal substance, but also engineering capabilities to carry on sub-processes or internal activities based on relevant existence of reality, and the products they produce, such as various models, artefacts, systems, and events. These organic aggregations can be used, or reused, to produce and aggregate other processes or systems. This concept is incorporated into the organic structure of OAP, which is illustrated in the following sections.

The notion of Organic Aggregation is applied on the structure and inner-connections of OAP in the context of SISE. Due to the complex and dynamic socio-technical settings of SIS, it is necessary to use a more open and flexible process structure, as opposed to using fixed numbers of phases and activities at the same level, and in a rigid way. Our vision of such a flexible structure is based on an inside-out approach, as opposed to the traditional rigid decomposition approaches. It also applies the guiding principle described in Section 3.3.3: Use "soft" methods instead of "hard" ones to deal with ill-understood phenomenon. Essentially, OAP structure allows the engineering activities to be dynamically arranged within a "softly" defined and loosely-coupled hierarchy, where activities can be added, substituted, or removed in a dynamic way. Furthermore, well-established activities can be cohesively aggregated together to achieve coarse-level engineering goals.

For instance, although activities at superordinate, or more fundamental, levels are less likely to change, the nature and inner structure of activities at subordinate,
or more detailed, levels can be adjusted in different context, as long as their purposes and functions still conform to those of their superordinate activities [Wang, 2010a]. For example, rationalisation can involve different types of activities depending on the nature of problems under consideration. Similarly, depending on whether target systems are services or embedded systems, realisation can consist of either capability discovery and capability design, or functional design and simulation.

Furthermore, well-established activities can be cohesively aggregated together to achieve coarse-level engineering goal, which enables formation of a flexible hierarchical process aggregations.

Such an arrangement has two advantages: firstly, it enables classification of engineering activities at different levels of granularities, which facilitates role assignment, communication, and collaboration among stakeholders. Unlike linear decomposition, the conceptual gaps between subordinate activities of the same superordinate activity are relatively small after multiple levels of refinement, because they all share certain common characteristics. Moreover, closely-related activities at finer levels of granularity can be assigned to different roles with improved overall communication. This is because people conducting closely-related activities can communicate and collaborate in a better way, due to shared or similar domain knowledge, vision, and cultural settings [Habermas, 1985].

Secondly, the hierarchical structure facilitates capability reuse and improves process flexibility. Our view of processes abstracts the conduct of a specific activity as a capability. Such abstraction enables aggregations of capabilities of multiple subordinate activities, which collectively form an overall capability of a superordinate activity. The aggregated capabilities of activities can be reused to serve other engineering activities, which removes the necessity to conduct individual subordinate activities. Therefore, this mechanism allows capabilities and activities to be aggregated according to the same hierarchy, and provides a flexible way to abstract and conduct activities within the overall process. More detail about the flexible process aggregation by reuse will be illustrates in Section 5.4 and Section 6.4.

The following sections present the OAP conceptual model in detail. Although a constructive, as opposed to decomposition approach, was adopted to derive the overall conceptual model, the OAP model is presented in an analytical way to help readers to gradually build a cognitive path to understand its core concepts and mechanisms. Due to our multi-disciplinary belief and a strong focus on human part of services engineering, the following sections involve intensive knowledge from non-engineering disciplines.
4.2 A unified model of reality and mind at the fundamental level

In the context of engineering, human beings, either working individually, or in collaboration with each other, continuously interact with the outside world using their intellectual and physical capacities to solve problems and achieve desired purposes. It is important to distinguish two types of phenomenon in the course of the engineering interactions, that is, intellectual activities that occur in human mind and other activities that occur outside of human mind. Such a distinction emphasises different ways to understand, develop, and use specific capacities that suit the nature and characteristics of engineering activities. This section introduces two fundamental concepts of OAP: reality and mind, as general classification of human engineering activities.

4.2.1 Mind and reality

As presented in Section 3.2.1, human mind is the most critical engineering resources, and should therefore be understood and supported in the engineering context. Because of its importance, in OASEF, we divide the whole engineering world into two realms, that is, an intellectual mental world relying on human mind, and anything else that does not occur in this world. We call the former a world of mind, and the latter a world of reality. This distinction reflects our vision of SISE that promotes a focus on human part of the socio-technical engineering environment (see Section 1.2.2 and 3.2.1). All OASEF activities and resources are arranged in these two worlds according to their relationship with intellectual problem-solving capacities of human mind.

In OASEF, mind is defined as follows:

Mind is the psychic experiences and cognitive capacities of humankind consisting of organic aggregations of purposeful mental elements that can be explicitly elicited, represented, analysed or captured, utilised and reused, by either a first-person who possesses these elements, or a second-person who observes or interacts with them.

This definition therefore involves structured intentional mental phenomena that serve specific engineering purposes, such as beliefs, conscious perception, intention and desires, rational reasoning, judgements, plans and design. It, however, does not involve unstructured psychological aspects discussed in Section 3.2.1. These aspects include some highly subjective non-physical phenomena that
is not well-understood, and cannot be effectively represented and manipulated, such as feeling, attitudes, emotions, religion, dream, and irrational free-will.

What about the side: the reality? The notion of reality is commonly referred to as existence of things, but is understood and interpreted differently by different people. Some people regard reality as physical spatio-temporal occupancy that is causally produced, whereas others believe it also involve conscious and subjective experience [Crittenden, 2009]. According to Whitehead [1978], processes and change are the essence of reality, as opposed to things or substances, and therefore, "the reality is the process".

Due to the limit of human intellectual capacities, it is impossible to obtain a complete picture of reality. Reality only becomes relevant to people during continuous interactions, that is, human experience. Therefore, reality refers to what we experience, what really occurs to us, whatever is exactly as it is, or what is an "actual situation" [Whitehead et al., 1979]. According to Whitehead et al., human immediate experience, or in his term the concreteness, is the most important and basic things in our lives that connect us to the outside world [Whitehead et al., 1979].

Although it is highly controversial in regards to the origin, nature, and value of reality, it is commonly agreed in modern philosophy and science that a notion of reality must incorporate physical objects, events, and processes, which reveal recognisable, and ideally measurable qualities and properties that show relatively invariant regularity [Quine, 1973, Sellars, 1963]. The physical objects here mean any existence that occupies space and time, and can be perceived via human sensational experience. They include physical things, such as stones and vehicles, which have stable properties such as weight, size and colour, as well as repeatable functions such as moving and other change of states. Reality also includes autonomous biological objects that interact and communicate with each other, such as animals and other creatures.

Therefore, the publicly accessible world that human beings reside in form the basis of reality. Such a world can be examined at all granularities: at the microscopic levels such as molecules and electrons, at the macroscopic level such as tables and trees, and at the social level such as organisations and human communities [Sellars, 1963].

The various understandings of reality reveals its nature from different perspectives. But they do not answer an important question: what is the value of reality, or why it is important? In OASEF, reality is the source, indicator, and purpose of mind. According to Sellars, reality, especially at the macroscopic level, is an indicator of the world that is better explained by science in terms of micro-particles.
4.2 A unified model of reality and mind at the fundamental level

Other researchers believe that macroscopic reality has important pragmatic value, that is, it is the fundamental source of the origins and evolution of language, which makes human communication possible [Quine, 1953]. In the meantime, microscopic reality serves the purpose of understanding and intentionally altering macroscopic reality, although the microscopic reality is less justifiable while involving hypothetical concepts that may come and go when sciences evolve [Quine, 1953]. Moreover, reality, involving observable and repeatable processes, events, and actual occurrences, provides the best way to understand, explain, and alter natural existence [Whitehead, 1978].

In the context of SISE, OASEF coherently synthesises the existing understanding of a vague notion of reality, and establishes a clear concept of reality in relation to mind. It is defined as follows:

Reality is the recognisable, measurable, and stable presence of the world outside human mind, which consists of interconnected physical objects, events, and processes that 1) occupy space and time at various granularities, 2) can be distinguished, re-identified, and communicated in such a spatio-temporal context, and 3) becomes relevant to humankind through continuous human experience.

In this definition, we do not differentiate real objective existence from subjective representation, which is highly controversial in philosophy. Instead, we consider such a distinction non-essential from an engineering point of view. As Whitehead [1978] states, the differentiation between “mere fiction” and “truly existing” requires “a very particular and demanding mode of interpretation”, that is, to understand their value and importance in a pragmatic sense. What it means in an engineering context is that: it is non-essential whether the reality truly exists or not, what matters is if it helps to solve practical problems. Therefore, we take a pragmatic stance to simply capture any element within reality that human mind interacts with, either concrete or virtual things, such as hardware and software in SIS, and physical and conceptual services in SOS. The linkage and interaction between reality and mind is human experience, such as conscious perception and behavioural actions, which will be discussed in the next section.

4.2.2 Linking mind with reality

Having distinguishing concepts of mind and reality does not mean a postulation of two discrete realms. Instead, they form a coherent whole through continuous interconnections between them. Therefore, a distinction between mind and reality
in OAP does not mean a situation called “bifurcation of nature”, or the clash between separately developed objective nature laws and human aspects such as intentionality and responsibility, which is regarded as the source of many serious problems in modern sciences [Whitehead, 1997]. The differentiation lies in the fact that distinctive types of phenomena in reality and mind do exist, and show different characteristics. The reality involves an objective nature that can be quantitatively examined, whereas the mind shows variable regularities and patterns that are harder to observe. However, they do causally depend on one another. In other words, certain activities in one world will lead to others. Therefore, both reality and mind collectively form an organic whole. Both worlds are considered equally important, and no one is considered superior to another in any sense.

The linkage between the world of reality and mind is human experience in the continuous course of human interactions with the outside world. That is, reality interconnects with mind on the boundaries between them through human experience. On the one hand, as discussed in Section 4.2.1, the totality of reality becomes relevant in an engineering context through interaction of human experience. On the other hand, human thoughts and thinking is fundamentally based on continuous obtaining and storing information from historical and present situations, and, in the mean time, carrying out actions for particular purposes and evaluating their effects.

The human experience therefore plays an essential role to bring both worlds together. Such experience includes bodily sensation, perception, and actions or processes that change the states of the reality. According to Willard Van Orman Quine, an influential American philosopher and logician, the totality of knowledge is fundamentally the matter of human experience, and “the unit of empirical experience is the whole of science” [Quine, 1951]. Feyerabend [1993] argues that the simplified empirical approaches to aggregating knowledge via experience of science and engineering is unable to deal with the complexity and dynamics of human purposeful activities. His supporting evidence is that any single rule, including empirical conduction of science, should be violated at certain point in time for the sake of making knowledge progress, which is justified by the history of modern science. Although Quine’s view may not hold true in all situations, especially when violation of methodological rules is required for significant scientific breakthroughs, it at least holds true for human enquiries into the relevant nature of the universe. That is, the supporting role human experience plays, in forming knowledge and understanding reality in the world of human mind, is hard to deny.

The organisation of engineering resources and capacities through intercon-
4.2 A unified model of reality and mind at the fundamental level

![Diagram: Organic Aggregation Process (OAP) structure - Fundamental level]

Figure 4.1: Organic Aggregation Process (OAP) structure - Fundamental level

nected reality and mind therefore forms a foundation for a conceptual model of OAP, which is depicted in Figure 4.1. At this fundamental level, OAP consists of both world of reality and mind, with human experience connecting them altogether, which is denoted by a square bar in the centre. The dividing bar helps to distinguish intellectual resources and capacities from physical ones, and also represents the intensive human interactions between mind and reality. It, however, does not mean that there is a clear cross-cutting distinction between both worlds. Instead, according to the guiding principle in Section 3.3.3, the boundary is “soft” and involves intensive human experience, resources and capacities that exist in both worlds.

Human experience thus links reality and mind together, all of which collectively forms a organic aggregation of resources and capacities to intelligently interact with the world outside human beings. The circle in the middle represents such an organic whole, which involves particular parts of overall engineering resources, activities, and experience that are relevant in specific engineering situation. In other words, the world of mind, being reducible and understandable in principle, and the world of reality, being perceivable and accessible by humankind, are coherently inter-related to one another in the context of particular engineering interactions. Such a coherent whole therefore represents anything that humankind is involved in such a situation, or what matters to involving people from the engineering point of view. The ultimate objective of OAP engineering is to form and grow this organic aggregation in terms of its scope, accuracy, adequacy, efficiency, and coherence in the continuous course of intelligent interactions between human beings and nature.
4.3 Concepts at the first level

The previous section built a fundamental basis for a coherent conceptual process model involving interconnected worlds of both reality and mind in an engineering context. Such a simplified model also defines the scope of subordinate OAP engineering activities that involve the engineering resources, capacities, and processes in the realm of human mind and reality.

At the first level, OAP consists of three interconnected activities: perception, conception, and action, which are depicted by Figure 4.2. The lines with text besides them represent specific process activities, whereas the joint points between two adjacent lines represent direct correspondence between one activity and another one. That is, an activity facilitates, or provides information for, another one. According to the guiding principle in Section 3.3.3, the joint points between two activities do not represent rigid milestones or turnover point. Instead, it might involve a turnover activity that includes various sub-activities, such as reviewing the present state of the process, and planning the next activity to be conducted. The arrowed curve in the centre indicates that the inter-connected activities of OAP need to be carried on in an iterative fashion.

![Figure 4.2: Organic Aggregation Process (OAP) structure - Level 1](image)

The remainder of this section explains the main concepts of OAP at the first level.

4.3.1 Perception

A general notion of perception involves judgements about the objective presence of the reality [Humphrey, 2000]. According to the Causal Theory of Perception (CTP),
the causal role of material objects or events in the world of reality is essential, and it generates certain forms of objective presence of reality within human mind, such as sense impression or sense datum that lead to our perception [Grice, 1961, Hyman, 1992]. Strawson [1974] strengthens this theory by adding a normal condition to constrain the causal dependency of perception in order to explain the situation where certain perceptions cannot be causally derived. These theories emphasise the causal dependency of perception, because of the fact that perceptual experience does depend on the presence and nature of things in reality. Therefore, presence of reality is an important factor of perception. Although some researchers argue that the causal law would not always be contingent even though it works in some conditions [White, 1961], perception is influenced by what the perceivers do to change their relations to the factual things in reality [Noë, 2003]. In other words, perception is influenced by action. In OAP, perception is an inner process involving human perception of reality, which captures and projects reality in the world of mind. Based on the OAP model at the fundamental level and common understanding of human perception, the term perception in OAP is defined as follows:

Perception is a process of accessing, knowing and understanding relevant portions of presence of reality through continuous and varying human experience, which produces representational achievements in human mind that can be stored, located, and utilised to enable and influence other purposeful engineering activities that, in turn, influences or alters the reality.

Therefore, perception involves the process of acquiring any information or knowledge about the outside world that is sought by, or brought into attention to, individuals or a group of people in the engineering context, such as knowledge about existing systems, guiding principles, and best practices. Human beings use the outcomes of perception to conduct engineering activities, through which existing knowledge is updated, and new knowledge is derived.

4.3.2 Conception

In addition to obtaining understanding of the reality through perception, human mind also involves a comprehensive system of interrelated higher order elements, such as dispositions, morality, rationality, intelligence competence, cognitive operation and performances, and imagination [Ryle, 1949]. The purpose of this section is to present a notion that scoping and capturing relevant elements from
these parts of human mind, in an engineering context. Such a notion is called conception.

Firstly, such a notion should involve rationality and intentions, such as causal or intentional reason, belief, and desires, which are believed to tightly relate to, either causally or intentionally, human intentional behaviours that alter the states of reality [Brandtsstadter, 2007]. Such an important relation can be used to explain, formulate, control, or predict the effects or consequence of behavioural interactions, and provides a means to analyse indirect reason and value of intentional human behaviour in addition to its direct effect [Heidelberger, 2006]. This is valuable in an engineering context. Therefore, the processes of forming and understanding these types of mental elements, including reasons, goals, value and their production mechanism, is an essential part of construction and exploitation of our mental space, the mind. Effective incorporation of them within a process would ensure the value and consistency of our behaviour and its relationship with the environments [Thalberg, 1984, Simon, 1986].

Secondly, it needs to comprise intellectual planning and engineering decision-making processes. Human beings, confronting various types of engineering problems, have to develop and improve our cognitive capacities to solve these problems, or improve the problem situations. In addition to capturing rationality and intentions, such capacities should involve intellectual design and judgement of global plans, which leads to low-level and fine-grain sequence of local actions that in turn alter the reality [Tversky, 2003]. Moreover, at various stages, a cognitive decision-making process is necessary to make selections among alternative ways of problem-solving. Such a process “begins with the identification of a stimulus for action and ends with the specific commitment to action” [Mintzberg et al., 1976]. The intellectual planning and decision making process typically consists of activities to identify goal and problem, develop a set of required capabilities, design predetermined actions and rules, generate alternatives, and to evaluate and select between options [Schwenk, 1984]. These processes are often constrained by conflicting objectives and limited cognitive capabilities, and are influenced by other factors such political power and chances [Eisenhardt and Zbaracki, 1992].

Thirdly, since some psychological mental elements, such as disposition, morality, and free imagination, are beyond the engineering scope of this research (see Section 1.1.3 and 3.2.1), they are therefore not considered in forming a notion of intellectual mental capacities.

Therefore, we develop a notion called conception in OAP to refer to processes involving these observable, identifiable, and crucial intellectual mental capacities. These mind-related activities are based on a complete or simplified representation
of reality, the outcomes of perception, depending on the nature of the situation in terms of complexity and uncertainty. The notion of conception is defined as follows:

Conception is a mental process that is concerned with human intellectual capacities for solving or improving engineering problems, including activities to 1) identify, understand, and represent unsatisfied situations, reasons, and intentions, and 2) derive structured and functional plans and make optimal decisions for the purpose of improving such situations.

Conception is based on outcomes of perception that represent the reality in human mind. Therefore, conception produces essential intellectual achievements that link perception with human behaviour, and link reality with mind. Such a link is important because human intelligence is one of the crucial factors that determines human behaviours. Explicit study and exploitation of conception helps to understand the general working processes of the human mind, and how such processes can be improved, or be utilised to assist solving difficult and complex engineering problems.

In summary, conception concerns important intellectual activities to deal with complex situations with which humankind is not satisfied. It involves activities of understanding of interacting problems, causes, reasons, and desired ends to be achieved, as well as intelligent solutions and plans to achieve these ends. Technology-independent conceptual artefacts can be generated in the world of mind, which are used to drive development or management of systems. Therefore, conception plays a central role in engineering.

4.3.3 Action

perception and conception create intellectual responses to the situation in reality. These responses exist in the world of mind and do not change the state of the reality. Therefore, they need to be realised and turned into actions. Accordingly, we develop a notion called action to refer to the way human being intentionally alter the state of reality.

Action has been studied from different dimensions: From the perspective of human body, Davidson [2001] regards actions as “mere movements of the body”, and introduces a notion of bodily motions to explain action. According to him, action means body movements or happening events that produce effect and alter the environment upon completion of such movements, and requires human consciousness and abilities to control the motions [Davidson, 1963].

95
more general understanding regards action as “the kind of movement that can be
engaged in for a reason, something it makes sense to speak of deciding or choosing
to do” [Rundle, 1997]. This understanding involves both physical body movement
and intellectual activities such as reasoning and decision-making. Similarly, most
common understandings of action relate action directly with mental states, such as
emotion and rationale, that either cause or affect the course and effects of actions,
either causally or non-causally [Davidson, 1963, Thalberg, 1977, Mele, 1992]. This
understanding aligns perfectly with our fundamental structure of process, that is
engineering resources consists of both world of mind and reality. The relationship
between action, mind, and reality naturally fits into our framework. Action in OAP
is defined as follows:

Action is the process to bring about a coherent system of occurrences
that produce material objects or events in reality and alter its states,
which consist of both bodily movements and their emergent behaviour,
and are caused, shaped, or influenced, by rational state of human mind,
such as recognisable reason, desires and intention.

Therefore, action deals with situations in reality by interacting with it,
according to understanding of systems obtained from perception, and conceptual
plan or solutions from conception. Moreover, because our definition of mind, in an
engineering context, is within a scope that excludes ill-understood mental elements
such as emotion and free will, we would not consider these aspects in a notion of
action, such as the psychological influence of emotion on action.

4.3.4 Summary

The inner processes of perception, conception, and action form a complete con-
ceptual loop between the world of mind and reality. Such a loop links relevant
knowledge and understanding of the reality world with high level mental ele-
ments, such as understanding of situations, engineering purposes, and optimal
conceptual plans and solutions, which, in turn, lead to concrete activities that
derive or improve systems as intended. Such an organic linkage is important
because it enables rational and intelligent efforts to achieve purposeful response to
challenging situations, which is essential to ensure success and avoid failure. The
important intellectual achievements from perception and conception are critical
for organic engineering. They justify the engineering activities being necessary
and sensible in the first place, and being of pragmatic value afterwards. In other
words, a complete loop enables and justifies the necessity and meaning of the whole
process. Moreover, as described in Section 3.1.2, the complete process needs to be
continuously and iteratively forwarded by adding feedback into the loop. Moreover, the boundaries between two adjacent OAP activities are softened in the sense that they reference each other in a loosely-coupled fashion via various related models. One activity does not have to wait until the dependent activity finishes, and to generate the complete output. For example, conception can start with only a portion of the outputs from perception, as long as there are sufficient information it can start. Continuous process iterations will gradually improve the incomplete parts.

In summary, perception, conception, and action form an organic aggregation that works collectively for the overall purpose of improving unsatisfactory situations. Perception generates understanding of relevant parts of the reality. Conception uses such understanding to identify and capture problems, rationalise intentions, and to make sensible plans and engineering decisions, which is realised or carried out by action, in order to change the states of the reality to desired ones. The pragmatic effects of actions also feed back to perception, which triggers another cycle of a complete process.

4.4 Concepts at the second level

The previous section presents the OAP at the first level, which consists of a complete loop of engineering activities: perception, conception, and action. This structure reflects classification of engineering activities at a coarse level of granularities, which consists of inner structures with different nature and characteristics.

Therefore, in OAP, engineering activities and their outcomes are arranged in a hierarchy. Activities at the first level are aggregated by subordinate activities at the second level, which is depicted by Figure 4.3. In the remainder of this section, we first explain the hierarchical structure as depicted the figure, and then describe the meaning of each concepts at the second level.

In an OAP hierarchy, subordinate activities belong to superordinate ones at higher level, in the sense that the former is within the scope of the latter. That is, the superordinate activity has general characteristics that all subordinate activities have in common. For example, sensation activities and abstraction both belong to a superordinate activity: perception. Therefore they share some common characteristics that belong to the superordinate one, such as the projection of the reality world within the mind world. Superordinate activities are represented by thicker lines and larger-size fonts in the figure. Whereas activities at the subordinate level are represented by thinner lines. A superordinate activity
represented by a single line, starting from point A to B in a clockwise direction, may correspond to a sequence of subordinate activities represented by a series of adjacent lines, which also starts from point A and ends with point B, in a clockwise direction.

Figure 4.3: Organic Aggregation Process (OAP) structure - Level 2

The activities at the second level are therefore organised in accordance with their superordinate activities in the world of reality and mind. The remainder of this section introduces these activities at the second level.

4.4.1 Sensation

As discussed in Section 4.3.1, before any higher-order purposeful cognitive elements in conception and behavioural actions are formed, human beings have to first obtain, sometimes implicitly, mental representations for objective knowledge of external reality, that is, the perception.

A notion of sensation should involve the product it produces and the function it serves. Human perception of the reality is only possible because of our sensation abilities such as senses of taste, smell, light, and sound, which is called “raw sensation” [Humphrey, 1999]. Humphrey holds that the ability to sense plays a crucial role in the state of mind and consciousness. According to him,
human consciousness is all about direct sensations, such as sensory feelings of colour, hunger, pain, and happiness. Therefore, it is the direct sensation and the ability to reflect on them that differentiates the world of mind from the world of reality [Humphrey, 1999, 2000]. Ryle [1949] takes similar view by arguing that "acquaintance with sense-data which are alone directly given us to knowledge of a world of permanent things undergoing change in a single space and time". Moreover, sensation is important to guide and form complex behaviour and reinforce communication and linguistic dispositions [Dennett and Weiner, 1993]. Humphrey's notion of sensation involves both recognition of "bodily stimulation" that evokes human feeling about what is happening now, and production of mental representations of such happening [Humphrey, 1999, 2000]. However, in our opinion, such understanding does not explicitly differentiate the stimuli, processes, and products of perception. Combining mental activities of perception with the reception of physical existence reality does not help to separate these two distinct concerns. It makes the perception process ambiguous and hard to control in an engineering context. As described in Section 4.3.1, the notion of perception involves both mind and reality. However, the notion of sensation mainly deal with the latter. Therefore, in OAP, sensation is defined as follows:

Sensation is the sensory experience to obtain and capture relevant direct data and information about systems under consideration in the scope of reality, which provides the sources of higher-order understanding and knowledge about existing reality. It influences both mental conception in the world of mind, and behavioural action in the world of reality.

Human sensation therefore mainly occurs in the scope of reality, which makes it possible to link reality with its mental representations in mind.

4.4.2 Abstraction

As discussed in the previous section, our notion of perception involves aspects that mainly relate to the world of mind. We develop a notion of abstraction to refer to these aspects, which is within the scope of perception as defined earlier.

During continuous interactions with the outside world, human beings have been developing various abstract form to represent the observation and understanding of the reality world. For example, various languages are used for this purpose, which help to exchange knowledge and understanding. In Section 4.3.1, our definition of perception involves such knowledge and understanding about the
objective presence of reality. The processes to derive and represent such a mental presence, based on results from sensation, is therefore an essential part of the notion of perception.

As Dijkstra (1972b) and many others pointed out, due to our cognitive limitations, the degree of complexity in reality often exceeds what human beings can directly cope with. Aids of some sort are often needed. Abstraction is one of the major mental techniques human beings utilise to battle this cognitive insufficiency. The process of abstraction reduces problems and solutions of large software systems by dealing with highly relevant aspects only, that is, by hiding unnecessary detail of systems under consideration.

In the context of engineering, the notion of abstraction, as a means of managing complexity, often refers to a process of generalisation or removing unnecessary information and restrictions [Ward, 2006]. One way to do it is to reduce the detail, such as manifesting what a system does without exposing the specific way to do it. For example, “Abstract Data Type”, an influential software engineering concept, uses abstraction of data characteristics and function to capture the main characteristics of complex objects [Guttag, 1977]. Since representation-independent specifications are used to provide abstract interfaces between various data implementations and their usage, the complexity of the underlying data implementation does not increase the complexity of systems that use this type of data. Therefore, abstraction provides an effective means to break complex systems into manageable elements.

Moreover, the outcomes of abstraction are organised at multiple levels [Dijkstra, 1972b]. A mental state is considered more abstract than another if it captures fewer detail about the particular objects or occurrence, or if it provides more generalisations about the universal elements, such as a type of, or a group of, identifiable entities that have generalised properties, characteristics, and functions. Furthermore, using appropriate level of abstraction is important. Ignorance of some important distinction or criteria during abstraction processes, such as performance variation can lead to serious problems [Guttag, 1977].

It is important to note that the abstraction in software engineering is different from the notion of abstraction in OAP. The former often implies its role: a design and implementation mechanism to simplify problem-solving process, and to hide implementation detail [Liskov and Zilles, 1974]. Whereas the latter is a much more general form of perception or understanding of the reality world, as opposed to design or implementation techniques. The following definition illustrates such distinction.

Abstraction is a process of interpreting, analysing, and under-
standing results from sensation, in order to obtain representational achievements in the world of human mind. It captures representational knowledge and understanding about some parts of the reality through a) generalising common characteristics, b) eliminating irrelevant information or restrictions, c) narrowing scope, or d) reducing unnecessary detail. Its outcomes, clearly captured and stored, represent the reality, and can be utilised to enable and influence other intellectual and behavioural engineering activities that eventually alter the reality for specific human purposes. Abstraction is arranged at various levels in terms of the degree of generality, the degree of relevance to the part of reality under consideration, and the scope and level of detail.

Therefore, abstraction, along with sensation, organically form the perception process. The sensation process acquires direct data and information from the reality. The abstraction process uses the sensed data and information to create mental representation of the reality at different levels, and in the form of models that will be introduced in the next chapter. Altogether, these perceptive data and abstract models form the world view of the observers, which sets the context and preconditions for successive engineering activities.

4.4.3 Rationalisation

In Section 4.3.2, we established a notion of conception that captures the important human intellectual capacities in an engineering context. We introduced the most important part of engineering intellectual capacities such as reflection and planning. But what exactly these mental activities are? What are the nature and pragmatic value of them? This section and the following one will look into these questions, by introducing two notions called rationalisation and realisation.

As discussed briefly in Section 4.3.2, human engineering problem-solving should start with observing and understanding the problem first. Traditional software engineering has a focus on forming clear and complete specifications of problems to be solved. We believe this focus needs to be taken further towards justification of problems in socio-technical environments, for example, reasons and rationale. The purpose of this section is to present a notion that scoping and capturing this part of human mind, in an engineering context. Such a notion is called rationalisation.

The concept of Organic Aggregation introduced in Section 4.1.2 requires not only agile response to change, but also rational and justified actions. Reason plays a significant, although not exclusive, role that binds human mind with action.
and influences the way reality is changed. Rationality, a voluntary cognition, determines how human beings make cognitive decisions on what to believe and what to do [Pollock, 2006]. Firstly, it determines, reinforces, or influences the occurrence, direction, and effects of action through entailed intentions and desires [Stoutland, 1970]. Davidson illustrated this point in his famous essay by saying that "beliefs and desires of an agent... rationalise an action, in the sense that their propositional expressions put the action in a favourable light, provide an account of the reasons the agent had in acting, and allow us to reconstruct the intention with which he acted" [Davidson, 2001]. Secondly, human purposeful behaviour, being the major means to solve practical problems in the context of engineering, is best identified and explained by attributing definite reasons to a human being, either explicitly or implicitly [Thalberg, 1977, 47]. Although there is not necessarily causal relation between reason and behaviour, they do reinforce each other in various situations [Stoutland, 1986]. Thirdly, reason also provides effective criteria to judge success or failure of action, by comparing the difference between intention and effects [Thalberg, 1977].

Davidson’s definition of rationalisation, focused on its relationship to action, is the explanation of actions by giving the agent’s reasons for doing what has been done, or in other words, “the reason rationalises the action” [Davidson, 2001]

In an engineering context, the ultimate rational desires of any human behaviours is to solve practical problems for various human purposes. Therefore, we will only consider reasons and intentions within a problem-solving setting. Any other factors, such as personal belief and imagination, will not be addressed even though they might serve the explanation purpose well in certain situations. Rationalisation is defined as follows:

*Rationalisation* is the cognitive process to discover, understand, and represent sensible stimuli and desired intentions that explain or positively reinforce action, which reside in, or relate to, unsatisfied situations in perception and other parts of mind world that violate either causal truth and stable regularities, or desired intentions.

Therefore, rationalisation is a crucial part and plays a significant role during the entire OAP life cycle, in order to form an Organic Aggregation of processes. Rationalisation promotes a sensible process that starts with a strong focus on the cognitive rationality, instead of rushing into implementation technologies. It provides the reasons that justify successive activities, and the desired ends for which other activities would aim. Pollock [2008] holds similar view by stating that human problem-solving activities should always involve justification of the way
beliefs and actions are formed and maintained. According to Pollock, there are two types of justification processes based on rational belief and rational behaviour: "epistemic rationality" and "practical rationality". More importantly, the former should presuppose an account of the latter. To a sense, rationalisation provides a means to support an engineering paradigm shift towards understanding and improving problem situations and raise the level of abstraction in software engineering, a vision that we presented in Section 2.4. In summary, rationalisation, which recognises and captures the a cognitive justification process, is necessary to rationally explain and positively support the operational behaviours of successive engineering activities, such as realisation and action. For an engineering process to be organic, it is necessary to enable and foster effective rationalisation.

### 4.4.4 Realisation

As presented in Section 4.3.2, our notion of conception also involve another critical part of problem-solving, the intellectual capacities to solve particular problems. This section introduces a notion called realisation to understand and capture this critical portion of human mind.

Based on a good understanding of problems and unsatisfied situations, human beings strive for better intellectual solutions to identified problems that are both effective and efficient. Explicit rational intentions are crucial factors that differentiate a plan of actions from a chain of actions [Miller et al., 1960]. The intellectual efforts to derive intentional and rational plans of actions occurs in human mind as a fundamental base of behaviour actions. These plans of actions involve human processes to make judgements, choices, and predictions, and are influenced by either experience and memory, or generalised abstract principles and deductive logic [Lipshitz and Cohen, 2005].

Derivation of intellectual solutions to problems is based on either description of causal or relating factors and empirical patterns that lead to sensible design, such as motivations and circumstance, a prescription of logical plan based on guidelines, rules and deduction, or a combination of them [Nutt and Wilson, 2009]. For example, a solution to a complex engineering problem is best formed by contemplating a range of alternative options connecting to the problems and environments, and by comparing the effectiveness of their causal consequences in accordance with objectives based on empirical experience and mathematical deduction, before actually doing any of them [MacCrimmon and Taylor, 1976].

Therefore, from an analytical perspective, a complex global intellectual solution is derived from judgements, choices, and expectations of detailed local solutions. These solutions are justified ones in the sense that they are ex-
pected to best suit the human intentions, according to empirical experience and theoretical knowledge, before they are actually used to guide occurrences of actions. The application of these solutions therefore are expected to improve the well-understood problems. Moreover, it is also important to emphasise the dynamic aspect of these intellectual efforts. That is, these activities themselves are dynamic and interactive processes, as opposed to one-off and one-way events. They involve adequate response to occurring change, such as managing and altering plans while interacting with dynamic environments and various autonomous participants [Garvin and Roberto, 2003].

In summary, human intellectual problem-solving capacities are important engineering resources for purposeful human activities. It is therefore important to develop and capture these capacities. We define a process called realisation for this purpose. It is defined as follows:

Realisation is a dynamic cognitive process in the world of mind, which involves the design, formation, judgement, and evolution of conceptual plans and decisions that consist of low-level and fine-grained local activities and capabilities, which are organised in a particular way, and collectively, aim to satisfy the desired ends, and to improve the problem situations of rationalisation. These conceptual efforts are justified, either empirically, or theoretically, in the sense that occurring behaviours in accordance with them are expected to approximate the identified desired ends with a high degree of confidence.

4.4.5 Summary

We have so far presented the second-level activities for perception and conception. There is another type of activities in the world of reality, that is, action. As defined in Section 4.3.3, action involves activities that occur and directly affect the reality, such as implementation, testing, deployment, and control. At the third level, action includes aggregations of implementation and control. The former is about executing the plan, or doing things in the reality to achieve or approach desired ends, according to the conceptual achievements of abstraction, rationalisation, and realisation. Whereas control is the process to monitor, adjust, and manage the systems under consideration during continuous interactions between systems and environments. It is guided by conception and influenced by perception. Due to constraints on time and resources, in this phase of research, we do not delve further into some important parts of action such as control. Instead, only the implementation part will be investigated and supported, which will be discussed in chapter 5 and 6.
In summary, at the second level, OAP consists of some clearly defined processes and activities. Abstraction and sensation organically form perception. The sensation process acquires direct data and information from the reality, and provides these achievements to abstraction, which processes the sensed data and information to create mental representation of the reality. Conception consists of organic aggregation of rationalisation and realisation. The former concerns problem, cause, reason, and intention such as desired ends. The latter involves conceptual planning and decision-making process that determines detailed and structured actions to achieve the desired ends. It is determined and characterised by rational intentions of rationalisation, and involves organic conceptual plans of actions, that is, rational, sensible, and workable plans. Rationalisation links perception with realisation, which in turn links rationalisation with actions.

4.5 Concepts at the third level

Section 4.4 presents some critical parts of engineering resources and activities in the world of reality and mind. They provide important resources to deal with complex engineering situations. As presented in Section 2.1.4, this research aims to address the current lack of human aspects in the engineering context. We especially focus on certain parts of human mental capacities in the engineering context, such as abstraction and rationalisation. Therefore, these two processes will be further studied and supported. This section will present our work in this regard.

We aim to further raise the level of abstraction in SISE by providing better understanding and support for higher order human intellectual capacities, that is, the abstraction and rationalisation in our OAP structure. Human beings have obtained quite a lot of understanding of the human capabilities for problem-solving and decision-making, and developed a lot of techniques to assist or improve such capabilities. It is relatively easier to support realisation within the scope of this research. However, there is still a lack of understanding of how to conduct abstraction and rationalisation, and to capture, organise, and use their achievements. The remainder of this section address this issue. We will discuss OAP at the third level, mainly in terms of abstraction and rationalisation.

4.5.1 Inner structure of abstraction

Figure 4.4 depicts OAP at the third level. This section will present two notions with regards to abstraction.
Figure 4.4: Organic Aggregation Process (OAP) structure - Level 3

There are two types of abstraction in OAP: System Abstraction (SA) and General Abstraction (GA). The former is defined as the process to obtain and capture knowledge and understanding of relevant parts of system and the environment in which it resides in, which are under consideration in an engineering context. As a few examples of SA, the internal structure of a particular system, the format and meaning of exchanged data, and the specific flow of action during a business process are all understandings of particular systems, which constitute the outcomes of SA. They can be derived by observing and interacting with systems under consideration, and can be represented using various forms.

Whereas GA is defined as a process to derive and represent the experience, knowledge, understanding, valuable insights and wisdom in a general sense, which are not necessarily specific to a particular system, but can be applied to a range of, or a type of systems. Example of GA includes theories, physical laws, patterns and best engineering practice, common nature, characteristics, functions, and relationships among a range of systems.

As special forms of abstraction, GA and SA are tightly related to each other. GA provides knowledge that is more abstract than the knowledge from SA, in the sense that some knowledge and understanding of the former can be applied to derive or influence the latter. For example, in order to derive good system specific SA models
that accurately reflect the reality, it is necessary to obtain a good understanding of
a system by observing it and analysing it, with support and guidance from general
knowledge obtained from GA processes. In the meantime, knowledge in SA, which
are obtained from interactions and specific experience with specific systems, can
also be generalised, which creates generally applicable GA models that can be
used at later stages of OASEF. Therefore, GA and SA influence each other and
complement each other, and collectively form an organic aggregation of abstraction
of the reality at different levels.

4.5.2 Inner structure of rationalisation

As described in Section 4.4.3, rationalisation is a critical part of organic process
that ensures the “right” system is derived. This section will explore what is
involved in the rationalisation processes. Briefly, it involves two processes:
Problem Situation Exploration (PSE) and Desired End Exploration (DEE):

4.5.2.1 Problem Situation Exploration (PSE)

People need to make sense of complex situations in order to make right decisions
and take suitable actions. Such situations are often complex and mysterious,
and consist of many inter-connected problems, due to the complex nature of the
universe and human beings. Checkland [1981] refers to this type of complex
situations involving intertwining problems and affecting factors as “problem
situations”. According to him, such “problem situations” can be far more complex
than what humankind can cope with, because we do not have effective processes to
obtain clear understanding of such complex situations. Therefore, he emphasises
the importance of continuous learning processes to make enquiries into complex
problem situations for the purpose of making improvements [Checkland and Sc-
holes, 1999]. Similarly, Simon [1978] refers to the representation of environmental
settings as problem space, which refers to “the way a particular subject represents
a task in order to work on it”. Problem space is a subset of the whole environment
where the problem-solving process is part of. Therefore, problem space emphasises
the nature and characteristics of problems during problem-solving processes.
Whereas problem situations emphasises the environment that considered prob-
lematic. Because of good alignment with the principle Environment-driven view
we adopt (see Section 3.3.1), we use the latter term problem situation in the
conceptual framework of OAP, which refers to the complex problem situations in
rationalisation that is considered of interest, and of significance, for the overall
engineering process. Since rationalisation takes outcomes of perception as inputs,
a problem situation involves cognitive processing of the available achievements from perception.

For complex problem situations, the involving factors and their relationships that affect the sense-making and decision-making processes are complex and often uncertain. For example, studies show that complex factors were involved when U.S. Army Black Hawk helicopters were accidentally shot down by two U.S. Air Force F-15s [Snook, 2002]. A lack of systematic processes to assist making sense of the complex situations would easily lead to disasters. Understanding of this point is important for people in critical situations, where decisions need to made in a rapid way, and the consequence of such decisions needs to predictable and well-understood. Snook [2002] therefore emphasises the importance effective processes to make sense of complex situations because “good people struggling to make sense... bad ones making poor decisions”. Weick [1995] holds similar view by arguing that there is a need for effective processes to synthesise various aspects of complex problems because “...the problem is that there are too many meanings, not too few... the problem is confusion, not ignorance” [Weick, 1995].

The importance of obtaining good understanding, and cognitive representations, of problems and situations has been emphasised by many researchers [Newell and Simon, 1972] [Einhorn and Hogarth, 1981]. Einhorn and Hogarth argues that the process to understand and represent problems and affecting factors is crucial for making proper judgement and choosing the right solutions. Larkin [1985] holds similar view by stating that it is essential to have the ability to obtain better understanding, and proper representations, of problems in relevant environments, preferably at higher level of abstraction, in order to develop advanced problem-solving capacities.

Therefore, for engineering complex SOS in their volatile and uncertain socio-technical environments, it is critical to have a process to derive clear understanding of involving problem situations. Better understanding of problem situations in a broad scope is critical intellectual achievements, which determines the effectiveness of actual decisions being made. Rationalisation is designed for this purpose.

An essential part of rationalisation is to understand and represent the problem situations human beings involved in the reality according to human desired intentions. But how do we do this? According to Cowan [1986], the essence of the problem recognition process is the explicit accumulation of discrepancies between the perception of situations and the desired states, which become problems when they pass beyond certain threshold. Therefore, identification of problem situations would be based on outcomes of perception, such as understanding of GA and
SA. According to Klein et al. [2005], problem identification also involves other factors, such as expertise, stance, and attention management, that link perceivable problem situations with evolving human expectation. Therefore, problem situations are often understood by sense-making, a process that involves continuous development of mental constructs that rationalise or explain human purposeful behaviours [Weick et al., 2005]. That is, problem situations can be derived based on the difference between perception, the current states of reality, and the expected states.

Furthermore, clear understanding of problem situations needs to take into account the wider environment in association with people involved. As described in Section 3.2.2 and 3.3.1, for any engineering processes, the relevant environmental settings must be understood, represented, and made accessible.

Moreover, complex problem situations are best understood through processes that involve interactive human experience, sense-making, inference, and hypothesis. That is, we need a process to gradually understand various aspects of the problems and affecting factors, through continuous interaction with the situations of concern, such as verifying hypothesis according to the previous experience, or enumerating and trying all possibilities. We call such a process exploration.

Such an interactive process, or the exploration, is critical for understanding problem situations. This is because of two reasons. First, problem situations involve a high degree of complexity and uncertainty. They cannot be understood either because there is a lack of previous knowledge and experience for similar situations, or because the characteristics or scope of the environment and involving factors are unknown or unpredictable. A process of exploration is therefore necessary to obtain better understanding of involving factors and their significance. Second, problem situations can, and often do, change. As Weick [1995] emphasises, a process of continuous interactions is important as situations change while we tried to understand them.

Therefore, we derive a notion of PSE to refer to the processes to iteratively make inquires into the complex engineering situations that human beings are part of, in order to make purposeful improvements. We use Checkland's term: "problem situation", to emphasise the importance of recognising, analysing, and understanding complex human situations in the engineering context. In OAP, the notion of PSE is defined as follows:

Problem Situation Exploration (PSE) is an interactive and exploratory process to detect, understand, and represent discrepancies, between the perception of reality and the expected states of reality in human mind. It consists of inter-connected problems and factors that
affects or constrains it, within a particular evolving environmental setting that can be used to explain, understand, derive, influence, or predict human intentions and purposeful action.

### 4.5.2.2 Desired End Exploration (DEE)

Identifying and understanding *problem situations* is important. However, it is not the end to be achieved. The value of any engineering activities would be to make improvements to the *problem situations* and to satisfy human needs in the engineering context. What do people do after they feel unhappy about a situation? They either do something about it, or do nothing at all. Before understanding the pragmatic value of these behaviours, it is important to understand what causes different actions. Although the answer to this question is difficult and is not clearly understood, it is commonly believed that cognitive capacities play a major role. Human experience tells us that complex *problem situations* are dealt with by using intellectual capacities, as opposed to by behavioural actions or physical force [Ackoff, 1999]. There are therefore some important activities occurring within the world of human *mind*. Identifying engineering intentions is one of them. According to Checkland [1981], for any purposeful human activities, it is important to understand human purposes, in their specific environment [Checkland, 1981]. It raises a big question: how do we do determine what is to be achieved?

Human *mind* seems to involve complex mechanisms to decide what they want. There are many factors involved in such processes. Essentially, the capacities of determination and formation of what is desired and intended need to be based on achievements of *perception* of the *reality*, and intellectual reasoning. The determination processes also involve analysis of problems in accordance with higher level human purposes, taking into account various constraints and competing forces. The ultimate objective of engineering activities is to form the best view of human intentions that suits specific human purposes and interests, and to approximate such desired stated, and to remove undesired ends or constraints imposed.

Ackoff advocates a different way to improving the *problem situations*, that is, through a continuous process to speculate an idealised system, and to gradually bridge the gaps between the reality and the ideal. Therefore, the very next step after PSE would be to identify and understand human intentions in the engineering context. Such intentions could involve different levels of concreteness and softness (see Section 3.3.3). Ackoff [1999] proposes a notion called *desired end* to reflect various types of human intentions, such as ideals, objectives, and goals.

Therefore, we use *DEE* to refer to the human cognitive processes to identify and
understand human intentions in the engineering context. We define DEE within the context of OAP by putting it into the coherent relationship with other OAP inner processes, which illustrates its pragmatic value. Its definition is given as follows:

Desired End Exploration (DEE) is a process to form and represent justifiable human intentions that reinforce, rather than impede, the achievements of higher order purpose, based on outcomes of perception of reality and understanding of PSE within specific environment, which aims to achieve harmonious and desired results using available resources, and to remove or avoid undesired constraints and unwanted results, in support of derivation and execution of behavioural actions.

4.6 Summary

In this chapter, we proposed a specific conceptual process model, called OAP, to help to address the services engineering challenges. It aims to answer the first part of research question V, that is, What conceptual mechanism do we use to address each identified issue? Based on the theoretical foundation and guiding principles, some specific concepts and mechanisms are designed to battle the challenges of SISE and services engineering. An important notion called Organic Aggregation is developed, which fundamentally supports the flexible hierarchical structure of OAP.

As an important component of OASEF, OAP provides a coherent and flexible conceptual structure that comprises various carefully-categorised, well-organised, and inter-connected activities at various levels. At the fundamental level, there are two worlds: reality and mind, which distinguish human intellectual efforts from externally world. On the one hand, the world of mind involves intellectual efforts that are essential precondition of problem-solving in complex chaotic situations. On the other hand, reality involves activities that are closer to observable and justifiable existence that can be better understood and controlled, and hence can be reproduced, automated, or controlled in a relatively easy way, compared with the cognitive activities in the world of mind.

At the first level, OAP consists of three interconnected activities, namely perception, conception, and action. Perception is a set of activities that acquire information or knowledge about the reality, and form relevant representations of reality in the world of human mind. Conception, based on outcomes of perception, involves human cognitive or intellectual activities in the world of mind, such as
reasoning, rationalisation, decision-making, or plan-making sub-processes. Action, conducted in the world of reality, deals with unsatisfied situations by conducting purposeful activities, reacting to events, and interacting with the outside worlds, according to outputs from perception and conception, such as knowledge of involved situations, desired ends, rules and laws, designs, and plans. These activities are interconnected with each other and form continuous and iterative forward loops.

At the second level, perception consists of sensation and abstraction. The former involves activities that directly collect required data from reality, such as measurement of temperature and humidity, or real-time network performance. Whereas the latter, occurring in the scope of mind, is a type of activities that synthesise or interpret the data and information collected by Sensation, and produce representations of relevant reality in mind.

Moreover, conception consists of rationalisation and realisation, both of which occur in the world of mind, and are crucial intellectual engineering efforts. Rationalisation involves activities that explore, understand, and represent sensible stimuli and desired intentions that explain or positively reinforce action. Whereas realisation is defined as activities that involve design, formation, and judgement of global plans and decisions that consist of low-level and fine-grain sequence of local activities and capabilities in the world of mind.

Furthermore, action consists of implementation and control. In contrast with thinking, reasoning, or designing in the world of mind, implementation is about actually doing things in reality, in accordance with the conceptual design and plans from realisation, and in pursuit of achieving identified desired ends from rationalisation. Whereas control provides verification, run-time monitoring and management of systems during continuous interactions between systems and their environments.

These OAP inner-processes and activities can have subordinate activities at more detailed levels. For example, abstraction includes System Abstraction (SA) and General Abstraction (GA); Rationalisation consists of Problem Situation Exploration (PSE) and Desired End Exploration (DEE).

The hierarchical structure of OAP aims to form aggregations of knowledge and capabilities in both world of reality and mind in a coherent way. It also creates automated linkage between human intellectual efforts and practical effects they produce. That is, a path from human sensation of the reality to abstraction representations in the world of human mind, to derivation of sensible problem situations, to intellectual design of engineering desired ends, and to practical engineering actions that produce artefacts to form or alter systems, in order to improve the problem situations of the reality.
4.6 Summary

Based on such a conceptual process model, the next chapter will answer the second part of research question V: *What practical mechanisms do we use for effective services engineering?* The aim is to provide specific methods, languages, and supporting engineering tools, to realise OAP in the context of SOS and services engineering.
Chapter 5

The OASEF approach to services engineering

Creativity is thinking up new things. Innovation is doing new things.

Theodore Levitt

Contents

5.1 The overall structure of OASEF ............................................. 117
5.2 Modelling languages ........................................................ 120
  5.2.1 Abstraction modelling languages .................................... 121
  5.2.2 Rationalisation modelling languages ................................. 126
  5.2.3 Realisation modelling languages ...................................... 130
  5.2.4 Unified modelling framework ....................................... 131
5.3 OASEF Modelling .............................................................. 132
  5.3.1 Abstraction modelling .................................................. 132
  5.3.2 Rationalisation modelling ............................................. 137
  5.3.3 Realisation modelling and Implementation process ............. 140
5.4 Reusing engineering capacities ............................................ 142
5.5 Implementation and proof-of-concept tools support .................. 146
5.6 Summary ............................................................................. 149
Chapter 5: The OASEF approach to services engineering

In Chapter 4, a conceptual process model called OAP is presented, which is fundamentally based on a concept called *organic aggregation*, for the purpose of understanding and capturing both high level human intellectual capacities and low level engineering capacities. It consists of a flexible hierarchy of interconnected inner processes or activities that are interconnected in a holistic fashion.

Based on such a conceptual process model and the theoretical foundation presented in Chapter 3, this chapter will present our answer to the second part of research question V: *What practical mechanisms do we use for effective services engineering?* Specifically, we present a complete framework of OASEF that involves concrete mechanisms and techniques in support of OAP. In order to illustrate its concepts and methods in a concrete sense, some examples in the context of online travel booking are used in this chapter.

As introduced in Section 1.4.2, the OASEF approach to services engineering has been assessed during the later stage of the *Focusing* phase. Two controlled proof-of-concept case studies were conducted to evaluate the effectiveness of OASEF in real world settings. The first case study, which is called Online Travel Booking (OTB) in this thesis, is a typical scenario used by the SOC community. Since the business setting of online travel is popular and can be easily understood, it is helpful to illustrate OASEF concepts and processes. Moreover, since the online travel business is well known, it helps to present OASEF in a context comparable with other approaches. Another case study is called FHSA. It is a government enforced national scheme to provide financial support for first-home-buyers, which involves interactions among various stakeholders, such as the government agencies, financial institutes, and superannuation providers. More detail about these case studies will be given in the next chapter during the evaluation. In this chapter, we will use some outcomes of the case studies to illustrate the basic concepts and mechanisms of OASEF.

Specifically, this chapter is organised as follows: Chapter 5.1 describes the overall structure of OASEF. Section 5.2 presents the modelling languages, meta-models, and graphical notations of various important OASEF models, especially models for higher order human intellectual capacities such as *perception* and *conception*. Section 5.3 illustrates the specific modelling methods and techniques. Section 5.4 describes the mechanism to reuse intellectual engineering capacities. Section 5.5 introduces the *implementation* processes that integrate the capacity reuse mechanism with a specific modelling technique. This section also introduces an implemented prototype tool environment that provides a range of integrated supporting tools. Finally, Section 5.6 summarises this chapter with a brief overview.
5.1 The overall structure of OASEF

As discussed Chapter 2 and 3, OASEF aims to tackle the service engineering challenge from the perspective of human intellectual capacities and competence. The overall framework of OASEF conforms to the general engineering framework and theoretical foundation described in Chapter 3. It also incorporates the OAP conceptual model described Chapter 4. Moreover, it integrates a model-driven method that enables and facilitates various OASEF activities and inner processes. A mechanism to reuse intellectual capacities is also designed according to the guiding principle described in Section 3.3.4, that is, to maximise engineering capacity reuse. The ultimate objective of OASEF is to enable and facilitate formation and exploitation of both human higher-order intellectual capacities and lower-order technical implementation processes. As discussed Chapter 3, this approach holds the key to attack the challenges of complexity, uncertainty, and volatility.

The overall structure of OASEF and its internal relationships is depicted in Figure 5.1. As shown in the figure, OASEF conforms to the general framework presented in Chapter 3. The bottom rectangle box represents its multi-disciplinary theoretical foundation, such as systems theory and modern philosophy of mind (see Section 3.2). The vertical rectangle on the left-hand side lists the guiding principles such as Environment-driven view and Maximise knowledge re-usability (see Section 3.3). In addition to the explicit principles listed in Section 3.3, some fundamental engineering principles are also incorporated within OASEF guiding principles because of their significant role in software engineering, such as higher level of abstraction and separation of concerns.

The central portion of the Figure 5.1 illustrates how OAP conceptual model is instantiated and incorporated in the context of services engineering. More specifically, an instance of the OAP conceptual model is fully integrated in OASEF at various levels. The activity of perception acquires information about the reality world through sensation, and forms general or system-specific abstraction of knowledge. The latter is used by conception to conduct human intellectual activities in the world of mind. Specifically, conception activities involve rationalisation, the reasoning and sense-making processes that produce understanding of perceived situations, inferred problems, and desired ends. It is realised by subordinate activities such as PSE and DEE. Conception also includes high-level design, engineering decision-making, and plan-making sub-processes through realisation activities, such as Capability Exploration and Capability Design that identify and design desired capabilities in accordance with outcomes of rationalisation. Furthermore action, conducted in the world of reality, deals with concrete service
systems by finding and invoking existing services, providing new services, or composing composite services. Outcomes of realisation and perception, such as desired capability and its inner logic, knowledge of existing services and SOS, will be used to guide and derive the action activities. Moreover, Service Infrastructure provides information about implementation environment such as Enterprise Service Bus (ESB), whereas Service Management provides control and monitoring of services in run-time environments.

Therefore, OASEF manage SOS through inter-connected OASEF inner processes and activities. Achievements of these activities collectively form specific Resource Images, in various forms, representing repositories of obtained knowledge and understanding from OASEF processes. These Resource Images classify OASEF engineering resources into four categories: World Images, Rational Images,
5.1 The overall structure of OASEF

Capability Images, and Service System Images. They can be referenced by, or used as input to, successive OASEF activities. For example, the rationalisation takes a World Image from perception as input, which is the projection of relevant reality in the world of human mind. In turn, rationalisation produces a Rational Image that represents rationalised desired ends from DEE, for the purpose of dealing with identified problem situations from PSE. Realisation takes the Rational Images and produces a Capability Image that represents required capabilities for the purpose of realising desired ends. Capability Images can be used by action to eventually produce SOS that alter the state of the reality. The produced SOS form Service System Images, which constitute a part of the reality, and can be perceived by another iteration of OASEF process.

The Resource Images have to be explicitly represented using certain languages, in order to be used by third parties, and be communicated among various stakeholders. Based on the guiding principle: "Everything is driven by model" (see Section 3.3.5), we use various models, based on specific modelling languages, to capture the content of Resource Images. As discussed in Section 2.1.3, models provide an effective means to capture certain aspects of knowledge that human beings can use to understand the outside world. In an engineering context, models are essential elements to understand, develop, and manage systems. Therefore, in OASEF, a model-driven method is fully integrated with the engineering process. That is, we use models and modelling processes to explore and represent Resource Images. From a modelling perspective, an OASEF activity involves the process to derive and manage specific models and their representational formalism under a unified infrastructure. The linkage between various OASEF activities therefore involves cross references and transformation processes among these models.

Such an integrated model-driven method is depicted by the vertical rectangle area on the right-hand side of Figure 5.1. This method includes a range of modelling language and techniques. Specifically, outcomes of GA in perception are represented using Correlation Network Model (CNM) modelling language that is designed to capture some crucial general knowledge about specific business domains, which will be introduced in Section 5.2.1. Similarly, outcomes of PSE and DEE in rationalisation are modelled by Problem Situation Model (PSM) and Desired Ends Model (DEM) respectively, which will be presented in Section 5.2.2. The Realisation process involves process modelling using general purpose modelling languages such as UML activity diagram [OMG, 2009b], or BPMN [BPMN, 2008]. BPMN provides a graphical means to represent business processes, and has been developed and maintained by standard organisations such as Business Process Management Initiative (BPMI) and OMG. It is similar to an activity diagram in terms of notations and semantics, but is mainly designed for the business users.
Moreover, the structure and relationship of services and participants are modelled using Service oriented architecture Modelling Language (SoaML), the extension of UML for the modelling of services within SOA by the OMG [OMG, 2009a]. Furthermore, outcomes of implementation are represented as BPEL or lower-level programming language such as Java. These modelling processes are in correspondence with OASEF activities. They are also inter-dependent. For example, the perception modelling is referenced by the rationalisation modelling, which, in turn, is referenced by realisation modelling. The derived realisation models can be transformed into implementation artefacts using model transformation techniques, such as the Eclipse Model To Text (M2T) [M2T, 2010].

In the following section, the specific modelling languages to express Resource Images are presented. These languages are specially designed to capture the nature and characteristics of some important human intellectual achievements during various OASEF activities, according to the definitions of these activities presented Chapter 4.

5.2 Modelling languages

As introduced in the previous section, OASEF incorporates a model-driven method to model various engineering activities, based on the guiding principle: Everything is driven by models, which is presented in Section 3.3.5. As defined in Section 2.1.3, modelling activities produce models as an artificial system in correspondence with the original systems, for the purpose of making improvements and satisfying human desire. Essentially, an OASEF activity is a modelling process that produces particular models to capture the OAP concepts described in Section 4. Models are represented using their meta-models, the modelling languages to express the models. Therefore, in OASEF, various types of models are derived and represented using certain modelling languages to capture and utilise the various understanding of reality and mind.

This section presents the specifically designed modelling languages and graphic notation to express and capture some important aspects of the higher order human cognitive achievements in OASEF Resource Images, such as understanding of problem situations and desired ends. Although general purpose modelling languages such as UML class diagrams and activity diagrams can be used to capture some parts of OASEF resource image, such as SA, they are often less effective in expressing domain specific knowledge. Instead, it is necessary, and often more desirable, to use DSL to represent some higher order cognitive knowledge. DSL provides an effective means to capture the knowledge of problem space using
domain-specific concepts and notions, with which users are familiar while making sense of the problem situations and expressing their needs [Czarnecki et al., 2002, Sprinkle, 2004]. Therefore, there is a trend towards more usage of DSL for the modelling of domain knowledge, in addition to the traditional use of general purpose languages in the context of software engineering [Bézivin et al., 2007]. In our cases, the domain is the human cognitive activities in the scope of OASEF. We therefore design a range of OASEF modelling languages that are close to human cognitive systems within the world of mind, according to the understanding and definition of pertinent components of the OAP conceptual model presented Chapter 4. These modelling languages allow engineers to derive and represent OASEF Resource Images, and use them to conduct successive OASEF activities.

Therefore, some OAP-specific modelling languages are designed for some important activities in the world of mind. This design reinforces one of our design objectives, that is, to promote higher order human intellectual efforts. We particularly emphasise abstraction and rationalisation in the early stage of the life cycle, which produce general perception knowledge and conception achievements.

5.2.1 Abstraction modelling languages

As defined in Section 4.3.1, the objective of perception modelling is to derive, capture, and manage relevant portions of presence of reality in the world of human mind. It produces knowledge and understanding of reality, in the form of models, which can be located, referenced, and used by other OASEF activities. This understanding clarifies the purpose of perception modelling. But what exactly should perception capture in the reality world? As discussed in Section 4.4.1 and 4.4.2, there are two types of perception: sensation and abstraction. The former collects required data directly from the reality, such as temperature or humidity data collected by sensors. Whereas the latter further forms higher order impression of reality in human mind, such as properties or characteristics of an entity being observed. Together, they form representations of the relevant reality that is projected in the world of mind, which are used to direct and influence other OASEF activities.

For the sensation part of perception, a range of modelling formalisms, either general purpose modelling languages, or DSLs, can be used to capture data directly collected from the reality. For example, structured data can be modelled using well-known Entity Relationship Diagram (ERD) [Chen, 1976] or Data Dictionary (DD) within Data Flow Diagram (DFD) [DeMarco, 1978, Gane and Sarson, 1977]. Moreover, UML class diagram, an industry standard that evolved from DFD, can be used to model the properties and relationship of collected data during Object-
Oriented Analysis and Design (OOAD) [Rumbaugh et al., 2004]. These well-known modelling techniques will not be explicitly presented in this thesis. Their usage in OASEF will be illustrated in Chapter 6.

Therefore, the outputs of perception modelling are mainly representations of cognitive abstraction based on analysis and understanding of the sensation. It represents the reality that is projected in the world of mind. As we mentioned earlier, some general purpose modelling language can be used to represent some parts of OASEF resource images, such as the System Abstraction. For example, OOAD modelling techniques such as UML can be adopted to represent System Abstraction. Specifically, general purpose UML structure diagrams such as class diagrams and object diagrams [OMG, 2009b] are used to capture information models, which represent structure, attributes, and relationships of information flowing through systems. Moreover, UML behavioral diagrams such as activity diagrams, sequence diagrams, and state machine diagrams [OMG, 2009b] could be used to represent behavioural interactions of systems.

Compared with SA, outcomes of GA capture more general knowledge about the reality. Although UML could be used to achieve this, we consider a modelling language that could better express the general knowledge that is not specific to a particular discipline, which we call Discipline-Independent Language (DIL). Using DIL to capture GA helps to form commonly understandable abstraction, and therefore facilitates sharing of multi-disciplinary knowledge by people with different background knowledge and skills.

The design of a DIL for GA considers an important cognitive mechanism that humankind has used to query into the universe: the means-end relationship. Its importance has been recognised in science and philosophy, and provides a fundamental means to understand the reality. According to Whitehead’s “ontological principle”, “there is nothing which floats into the world from nowhere” [Whitehead, 1978]. That is, whatever happens must be related to specific reasons, or a means that leads to it. Therefore, means-end analysis has been widely used as a general approach to understand problems and their solutions [Newell and Simon, 1972]. By observing and analysing the causal means and their results, human beings could develop cognitive capacities to reason about and solve general problems of various kinds. For example, means-end analysis in the realm of computer machines results in an evolving “complex web of causal connections”, which forms an essential basis to understand and design operations on hardware and software to solve complex problems [Sloman, 2009].

Therefore, a specific type of model called Correlation Network Model (CNM) is introduced in the context of GA. It is designed to capture the critical causal
relationships that the reality often reveals. A CNM captures two types of causal correlations that make abstraction and rationalisation of reality possible, namely hard correlation and soft correlation. The former specifies the “strong” causal correlation between a means and an end, where such a means “definitely” leads to the end, according to theoretical rules or empirical experience. Whereas the soft correlation refers to the “weak” correspondence between two elements of a system. Such a notion makes it possible to analyse certain complex situations, where no direct causal relations can be identified or justified. For example, a successful academic career for an individual depends on the talent, commitment, and intellectual capacities of the person, as well as the socio-technical environments in which he or she resides. There are also other irregular factors involved, such as opportunities and luck. We cannot use the hard correlation to reflect the inter-relation between the success and other causal factors because of the complex relationships among them and other unknown influential factors. According to the guiding principle: Use “soft” methods instead of “hard” ones to deal with ill-understood phenomenon (see Section 3.3.3), the soft correlation represents such a situation better. The causal factors are necessary, but not sufficient, means that influence the result of a successful career. Some of these factors influence the result in a more certain sense compared with the others, such as opportunities and social environments. Therefore, by incorporating both hard correlations and soft correlations, the causal correlations can incorporate all levels of causality. They are not given quantitative weight to indicate their significance because it is difficult to quantify the level of significance in terms of causality in all situations.

Based on the notion of causal correlations, we design a language for expressing CNM. Its meta-model (see Section 2.1.3.2 for the definition) is depicted in Figure 5.2. Such a meta-model consists of a Context that describes the environmental knowledge of the model, such as its scope and high level purpose, which conforms to the guiding principle: Environment-driven view (see Section 3.3.1). Context specifies target, the intended high order purpose of the modelling; constraint, known confining conditions or competing forces that influence the relationships; scope, boundary and relevance within or beyond this modelling activity; and priority, the level of importance and urgency.

Therefore, the construction of a CNM mainly involves a process of eliciting and exploring relevant causal correlations, including both hard correlation and soft correlation. A correlation links means with consequence. The former represents events, mechanisms, capabilities, processes, or actions that can occur, be performed, and are able to change the states of reality. The latter represents facts, attributes, or states of the reality that can be observed or sensed. The causal correlation between them means that the former causally leads to the
latter. That is, an action, in the form of means, is causally correlated to a set of states, in the form of consequence, in either a soft or a hard way. For example, demanding a creative employee to do a lot of routine tasks could cause frustration and inefficiency, where the means is the “demand”, and the consequence is the “frustration and inefficiency”. This correlation is a hard one. Moreover, a particular means can also lead to another means. For example, accepting more service requests could cause the service provider to drop existing scheduled tasks. Therefore, we distinguish another two types of causal correlations, namely action correlation and state correlation. The former refers to causal correlations between two means, whereas the latter refers to the correlation between means and consequence.

Moreover, consequences can relate to value, which is the pragmatic meaning of the consequences in the context of systems under consideration (see Section 3.2.4). Value is contextual because it is subject to the purposes of the observers and the states of the environment at certain point in time. It can be implicit and non-obvious because higher order value often needs explicit and conscious analysis. However, the incorporation of value is important to determine the relevance of causal correlations, according to the foundation described in Section 3.2.2 and
3.2.4. The meaning and fitness of the CNM depends on the understanding of values in terms of observed consequence.

Furthermore, means can relate to a range of aspects, or factors which affect how means can be conducted. Unlike means and consequence, aspects can neither be conducted nor observed as state of the reality. Instead, they are various perspectives that influence how means, consequence, and their relationship are understood. For example, going out for a picnic in a sunny weekend could lead to the happiness of the family. The means: “going out for a picnic”, can be influenced by multiple aspects, such as budget, location, length of time. Depending on the state of these aspects, the nature and characteristics of the means varies. Therefore the consequence and value would be different. Therefore, going too far and spending too much time might not lead to desired happiness. The value of aspects therefore lies in the way it directs, guides, and influences, the way causal correlations are understood and derived. The aspects can be categorised into a hierarchy, based on the subordinate relationship among them. They can also relate to each other. Chapter 6 illustrates this with an example in a real world situation.

![Diagram of Correlation Network Model (CNM)](image)

**Figure 5.3: The graphic notations of Correlation Network Model (CNM)**

Based on such a meta-model, a graphic language is designed for CNM, in order to graphically represent and manipulate corresponding CNM models. A range of graphical notations are introduced, as depicted in Figure 5.3. Specifically, elements
pertinent to involved *causal correlations* in CNM are represented by coloured rectangles within which their names are displayed. Blue rectangles represent *Aspect*. White, yellow, and green rectangles represent *Means, Consequence*, and *Value* respectively. Causal relationships among these elements are represented by lines that connect them. For example, a dotted line connects an *Aspect* with its subordinate one.

Therefore, *abstraction* modelling produces coherent aggregation of knowledge about the *reality*, which enables or supports successive engineering activities; Because of this mechanism to separate concerns between system specific and discipline-independent knowledge, we achieve a higher level of modelling abstraction.

### 5.2.2 Rationalisation modelling languages

As discussed in Section 4.3.2, *conception* involves human activities to reason about the perceived *problem situations*, identifying intentions or *desired ends*, as well as making or choosing conceptual plans or designs that provide specific means to achieve these *desired ends*. The objective is to use information and knowledge acquired during the *Perception* activities and to make sensible plans and designs that lead to effective action.

In Section 4.4.3, we argue that *rationalisation* is a critical part of *conception* and makes *organic* processes possible. It provides the reasons that justify successive activities, and the *desired ends* for which other activities would aim. Due to its critical role, we designed specific DSLs for *Rationalisation*, according to the analysis and definitions presented in Section 4.4.3 and 4.5.2.

According to its definition presented in Section 4.4.3, *Rationalisation* involves important intellectual activities in *mind*, such as understanding and reasoning of problems, causes, reasons, and desired intentions and actions. It generates implementation-independent cognitive artefacts that are used to direct and drive development or management of systems in the right way. As presented in Section 4.5.2, *Rationalisation* consists of *PSE* and *DEE*.

Therefore, The objective of *Rationalisation* modelling is to analyse and justify the reasons and needs of engineering processes toward sensible and rational decisions of actions. Two types of modelling formalism, namely Problem Situation Model (PSM) and Desired Ends Model (DEM), are designed to capture the crucial higher order cognitive achievements of *PSE* and *DEE*. For example, the output of *PSE* is captured in terms of complex inter-relationships between various problems, facts and constraints, and high-level purposes. Whereas the outputs of *DEE* is
5.2 Modelling languages

captured in terms of relationships between various desired ends, such as Ideal, Objective, and Goal. Together, PSM and DEM provide an important means to explore and represent problem situations and desired ends, which justify successive engineering activities and their produced engineering results as being organic.

- PSM

As presented in Section 4.5.2.1, PSE provides a means to understand the problem situations through human cognitive experience during the interactions between reality and mind. Such a process is supported by a specially designed DSL in the context of OAP conceptual model. According to Larkin [1985], the nature of the representations of problems determines how problems are addressed and whether they can be addressed successfully. Larkin argues that more abstract representations such as mathematical or scientific ones are better than what he called naive representations composed of entities familiar to everyday life. Therefore, the objective of the PSM is to formulate such an abstract formalism in the world of mind to capture understanding of problem situations. The PSM design is based on foundation presented in Section 3.2.1 and 3.2.2.

![Diagram of Problem Situation Exploration (PSE)](image)

Figure 5.4: The meta-model of Problem Situation Exploration (PSE)

In Section 4.5.2.1, we introduced the notion of problem situation in the context of OAP conceptual model, which is used to describe a complex situation consisting of inter-connected problems. The complex problem situations in PSE are captured and expressed using a PSM meta-model, which is depicted by Figure 5.4. It consists of a number of Problems, Purposes, and Facts.
Chapter 5: The OASEF approach to services engineering

*Problems* have various attributes such as *name, description, priority*, and the degree of *confidence* that stakeholders believe it is a problem. It can lead to, or be caused by, other *problems*. A *Problem* can also be caused by some *Facts* that are constrained by the reality, such as physical laws or legislation. Furthermore, a *Problem* can be justified as a problem if it is believed to violate specific *Purpose*. The latter describes the higher level aim or intention of stakeholders, and can have subordinate *Purpose*s. This meta-model can be used to analyse the *problem situations* and generate models to capture such understanding, which will be illustrated by examples from the case studies in Section 5.3.2.

Similar to the CNM graphic notation presented in Section 5.2.1, a range of graphical notations for *PSM* are designed to graphically represent *PSM*, which is depicted by Figure 5.5. Specifically, elements of *PSM* are represented by coloured rectangles within which their names are displayed. Blue rectangles represent *Purpose*, whereas white and green ones represent *Problem* and *Fact* respectively. Relationships among these elements are represented by connecting lines among these elements. Specifically, an arrowed line links a *Purpose* with its subordinate *Purpose*. A *Problem* is linked to a *Purpose* via a thick arrowed line. A dotted arrowed line symbolises causal factors of a *Problem*, such as another *Problem* or *Fact* that causes this *Problem*.

![Diagram of Problem Situation Model (PSM)](image)

Figure 5.5: The graphic notations of Problem Situation Model (PSM)

- *DEM*
  As discussed in Section 4.5.2, DEE is an important process to deal with identified *problem situations* expressed in *PSM*. It captures both high-order
human intentions and low-order goals and requirements in a soft sense (see Section 3.3.3). To apply the unified model-driven method within OASEF, we design a DLS to express the DEM produced by the DEE.

Figure 5.6 depicts the meta-model of DEM that captures the nature and relationships of various elements involved in DEE. The meta-model consists of a number of instances of desired end, which has a name, description, precondition that specifies the expected conditions and environments, a successTarget that describes the desired successful outcomes to be achieved, a failureTarget that describes the undesired or failure outcome to be explicitly avoided, as well as priority that specifies to what degree the end is desired. A desired end is categorised into three types according to Ackoff’s classification [Ackoff, 1999], namely Goal, Objective, and Ideal. Goal is a desired end that can be explicitly specified, easily achieved and evaluated; Objective can only be achieved via successive milestones and subordinate Goals; Ideal contains a number of Objectives, and has vaguely defined vision that can only be approximated and subjectively evaluated. The Goal has the smallest granularity compared with other two, and can have subordinate goals. A Goal also has specific correspondence with desired Capability that provides a means to achieve the desired goal.

![Diagram](image.png)

Figure 5.6: The meta-model of Desired End Model (DEM)

The graphical notations of DEM are depicted in Figure 5.7. Coloured rectangles with names inside represent DEM elements. Light yellow rectangles represent Ideal, whereas light green and dark green ones represent Objective and Goal respectively. White rectangles are used to represent higher
order Capability. Relationships among these elements are represented by connecting arrowed lines. Specifically, an arrowed line links a Ideal with its subordinate Objective, a Objective with its subordinate Goal, or a Goal with its subordinate one. A Goal is also associated with a Capability via an arrowed line.

Figure 5.7: The graphic notations of Desired End Model (DEM)

5.2.3 Realisation modelling languages

Realisation involves conceptual planning and decision-making towards a more concrete plan or design of action in order to achieve the desired ends. The DesiredEnd and Capability in DEM are further elaborated with more detail, which leads to models that contains lower level of abstraction towards the concrete reality. The process includes two types of elaboration: top-down and bottom-up. The former analyses the higher level capability, and designs necessary lower level capabilities and their connections, whereas the latter aggregates the existing services from the Perception models in a certain manner to meet the desired ends. These two approaches can be used separately or together to form a hierarchy of desired capabilities that work as a whole to achieve the desired ends. The UML Activity Diagram [OMG, 2009b] is used to capture the complex internal structure of a detailed level of capabilities.

The Realisation process includes two types of elaboration: top-down and bottom-up. The former analyses the higher level capability, and designs necessary lower level capabilities and their connections, whereas the latter aggregates the existing service from the perception models in certain manner to meet the desired
5.2 Modelling languages

ends. These two approaches can be used separately or together to form a hierarchy of desired capabilities that work as a whole to achieve the desired ends. The UML Activity Diagram meta-model [OMG, 2009b] is used to capture the complex internal structure of a higher level capability.

The realisation process stops on two conditions: 1) All subordinate capabilities can be provided by existing services captured in perception models; otherwise, 2) the subordinate capabilities can be provided by creating a new service, and the knowledge to do so is captured.

5.2.4 Unified modelling framework

In the previous sections, various modelling languages, for either domain-specific or general-purpose models, are introduced to facilitate OASEF activities. It is important to note that these modelling languages are used in a unified way in OASEF to ensure the organic interconnection among different OASEF modelling activities. That is, every OASEF inner process and activity produces specific models that all conform to a unified meta-meta-model. As a result, modelling activities in OASEF can be conducted and inter-connected in similar way, within a unified modelling environment. In this research, we examined two general modelling environments of this kind, namely AOT [Flint, 2006], and EMF [EMF, 2009]. The latter was used to support OASEF because of its technological maturity and wide support from an active development community.

As a result, although OASEF activities and inner processes use various modelling languages, they are all based on a unified meta-meta-model. This ensures that different types of models are manipulated and utilised in the same way. For example, UML models based on EMF meta-meta-model are used to capture information and knowledge in sensation and abstraction, in the same way that CNM, PSM, and DEM are manipulated. DSL are built on top of EMF modelling framework, which are supported by any EMF-compliant tools. Other modelling languages such as the UML activity diagram, SoaML, State Machine diagram, and BPMN can also be easily integrated with EMF framework using a wide range of tools, for example, tools provided in EMP [EMP, 2010]. Moreover, some model manipulation and transformation techniques based on the unified modelling framework are also integrated into OASEF to manage and transform various OASEF models in a unified fashion, and to generate desired systems in an automatic or semi-automatic way. These techniques and tools are developed and provided by various open source projects, such as Graphical Modeling Project (GMP) [GMP, 2010] and Model Development Tools (MDT) [MDT, 2010].
5.3 OASEF Modelling

As mentioned in Section 5.1, according to the guiding principle: *everything is driven by models* in Section 3.3.5, all OASEF activities are essentially modelling processes. The outcomes they produce are models, which is expressed and communicated using the modelling languages described in the previous section. In this section, we illustrate how OASEF activities are conducted using a model-driven method.

Some of these methods are illustrated via an online travel booking scenario, which normally involves integrated services that enable customers to either access travel-related information such as timetables, fares, and booking conditions, or reserve or purchase selected travel products from various underlying providers. A typical example of an online travel scenario is to orchestrate interactions among various participants to provide integrated travel packages to customers, such as enquiries and booking of airline tickets, hotel rooms, and rental cars [Curbera et al., 2002].

5.3.1 Abstraction modelling

As described in Section 5.2.1, abstraction processes form cognitive representations of relevant parts of the reality in the world of human mind. In Section 5.2.1, we described some modelling languages to represent abstraction models, including the general purpose modelling language such as UML, and the CNM, a DSL we designed for GA modelling. In this section, we will illustrate how abstraction modelling is conducted in OASEF. We start with GA modelling, a way to capture general knowledge.

5.3.1.1 GA: Capturing general knowledge

Abstraction does not necessarily have to be specific to the system to be built. It can be generalisation of system knowledge according to existing best practice or previous experience. CNM provides one way to capture such generic knowledge. According to the CNM modelling language specified in Section 5.2.1, CNM can be derived, using the supporting tools we developed. Figure 5.8 depicts a simple example of a CNM model, which captures the general knowledge to improve online transactions.

The context of such a CNM provides the contextual information about this model. It specifies a target: “Showing how online transactions can be increased
in the context of online travel booking business.". Its scope is “online purchasing made by non-corporate customers for personal travelling”. The constraints include “Limited budget and a lack of technical expertise in web applications”. This CNM contains an Aspect that is relevant to the context of this model, which contains three subordinate Aspects, namely “Marketing”, “Pricing”, and “User Interface”. It indicates that the performance of online transactions are associated with these three factors in the context of this model. In association with these factors, there are three Means, respectively “Apply aggressive marketing”, “Use real-time pricing in association with supply chain partners”, and “Improve website User Interfaces”.
The last Means also has a causal correlation with another three Means, that is: "Simplify booking process", "Set effective focal point", and "Apply consistent look & feel". This exemplifies some design principles that lead to improved User Interfaces. Moreover, Means also relate to Consequences. For example, in the figure, "Use real-time pricing" leads to "Increased rate of transactions"; "Apply aggressive marketing" causes "Increased Website hits"; "Improve website User Interfaces" results in the fact that "customers stay with the site longer than before". Note that the last two consequences also contribute to the first one. Unlike means, which represents particular action that can be performed, consequence are state of the reality that can be observed or measured, such as website traffic, the number of transactions per hour, and average length of time a user stays within a website. Finally, the consequence of "Increased rate of transactions" leads to higher-order business Value: "Increased revenue".

As illustrated by this example, the general knowledge of increasing online transactions is captured in the CNM which reveals the relationships among various influential factors and specific means. Such a model therefore can be used to guide or derive other OASEF inner processes, such as SA models for specific systems, in this case, to improve the online travel booking system. The knowledge captured in CNM can also be used to communicate with various stakeholders, and guide the rationalisation and realisation inner processes, to identify the key problems in specific situations, to understand the interrelated complex problem situations, and to clarify and design rational and feasible desired ends. A more complex example will be given in Chapter 6, to illustrate how this modelling mechanism can help to manage the complexity. In the next section, we will briefly describe how SA, the knowledge about the specific system under consideration, is realised in OASEF.

5.3.1.2 SA: Understanding the systems

In addition to capturing and utilising general knowledge during the GA activities, it is also important to understand the systems under consideration or pending development, and to communicate such understanding with critical stakeholders. This is what SA is mainly concerned with. As mentioned in Section 5.2.1, SA is modelled using general purpose modelling languages such as UML. This is because, traditionally, software engineering and systems engineering pay close attention to the specifications of systems under development. These general purpose modelling languages were designed for this purpose, and have been accepted and practised by most practitioners. These modelling languages are therefore effective to capture the system specific knowledge, which is easily
understood by many stakeholders. For example, UML class diagram provides a simple and widely-used formalism to capture static information, which reveals the attributes and behaviours of observable objects, and shows the relationships between such objects and other elements of the system. On the other hand, UML activity diagrams and state machine diagrams show dynamic behaviour, processes, and state transition patterns that are involved in the systems.

To illustrate what we mean by SA modelling, we first present a simple information model using a UML class diagram. As presented in Section 5.2.1, Information models are used in SA to represent the structure and attributes of information that flows among various participants and components of the system. An instance of Information model captures the abstraction of some information and data used in the OTB case study. The model is depicted by Figure 5.9. It contains two classes of information: FlightInformation and RoomInformation. The former represents information about a flight, which contains a range of attributes describing the flight, such as flight number, the departure and arrival city, and the departure and arrival time. Similarly, RoomInformation describes information about hotel rooms such as available date, price and type of the room.

![Figure 5.9: An example of UML class diagrams - System Abstraction](image)

Note that this model is deliberately made simple, and lacks some important attributes such as hotel name for RoomInformation, and contains incorrect data types such as departureDate. These kinds of less desirable or wrong initial designs are a normal phenomenon during software engineering. It will be improved in an example provided in Section 6.5, in order to demonstrate how OASEF supports change and evolution over time.

As illustrated by this example, in OASEF, UML class diagrams are used to represent a relevant portion of the reality, in this case, the online travel system. It provides the information about the flights and hotel rooms. This type of abstraction is used in many software engineering projects, and provides a good way to specify the existing systems, or systems to be built.
Chapter 5: The OASEF approach to services engineering

Besides class diagram, other UML modelling languages can be used during the SA process. We use an example to illustrate how to use UML language to express the internal structure of the online booking systems. Figure 5.10 depicts the structure of such a system using a UML use case diagram. It includes web interfaces, in the “TravelPortal”, to interact with “Customers”, and provides travel-related information services to accommodate information inquiries such as availability of flights, hotel rooms, car rental, and local weather. This information is obtained from various service providers. For example, various instances of “FlightProvider” provide inquiry, reservation, and purchasing services that allow “Customers” to book flights and make payment. Similarly, “HotelProvider”, “TaxiProvider”, “WeatherProvider”, and “RentalProvider” provide other types of services that are of value to the “Customers”.

Figure 5.10: An example of abstract model for an online travel booking web portal - Abstraction

The models derived from SA and GA collectively form the World Image, which is a bigger picture of the reality that is considered relevant for the engineering purpose. In other words, World Image is the relevant knowledge about the reality. It is the critical intellectual achievements obtained by continuously observing and interacting with the reality. The organic aggregations of these achievements in various abstraction models therefore become valuable intellectual assets that can be accessed and reused by other engineering activities. For example, the World Image is used for successive OASEF activities including the Conception and Action processes.
5.3 OASEF Modelling

5.3.2 Rationalisation modelling

As described in Section 5.2.2, PSM in Rationalisation provides a means to observe, analyse, and understand complex problem situations. In Section 5.2.2, we presented a DSL modelling language and introduced the graphic notations of PSM. To illustrate how PSM is represented using such a language, an instance of PSM is presented here to analyse the online travel booking problem situation, which is depicted by Figure 5.11.

![Diagram of PSM example](image)

Figure 5.11: An example of PSM in an OTB scenario - PSE

This PSM contains three higher level Purposes, including “Ease of use”, “Quick Responsiveness”, and “Effective User interfaces”. These PSM elements are generated systematically in accordance with general knowledge captured in GA models within World Images. These contextual purposes represent customers’ expectation of the online travel system. This PSM also contains a range of higher level Problems that violate these high level Purposes: 1) “The booking process takes
too long” violates the Purpose of “Quick Responsiveness”; This problem reflects the observation that the efficiency of travel enquires and booking does not satisfy the customers’ expectation. 2) “Booking process is complex” violates the Purpose of “Ease of use”; and 3) “Too many Web forms and widgets” violates the Purpose of “Effective User interfaces”. Moreover, some Problems have causal Problems: Problems 1 mentioned above is caused by five other Problems, including a) “Some partner service providers are less efficient”, b) “The interaction interfaces with partners are complex and diverse”, c) “Too many Web forms and widgets”, d) “Insufficient network bandwidth”, and e) “Booking process is complex”, which in turn is affected by other Problems. Furthermore, some Problems are determined by the Facts: “Insufficient network bandwidth” is determined by the Fact of “physical network traffic constraint”, whereas “The interaction interfaces with partners are complex and diverse” is determined by the Fact that, by nature, there are diverse types of service providers in the travel booking business, with different capacities and priorities. Altogether, this PSM reflects the internal dependencies of various influential factors within a problem situation. Exploration of problems, purposes, and their relationship reveals their nature and characteristics, and helps to identify, understand, and capture important problem situations.

The PSM also helps to systematically generate other models during successive engineering activities. That is, other models can be derived in accordance with every single problem that has high priority. For example, in accordance with a major problem “The booking process takes too long” that violates the higher priority purpose “Quick Responsiveness”, a DEM is created during the DEE process, in order to derive and capture the desired improvements for the problems. The construction process of a DEM therefore is systematically guided by, and makes reference to, elements in the PSM we presented earlier.

Figure 5.12 depicts such a DEM, which is represented using the DSL modelling language we introduced in Section 5.2.2. To address the Problem: “The booking process takes too long”, a desired higher level DesiredEnd is considered highly relevant. Due to the fact that such a desired end is too general and lack specific targets, it is created as higher level Ideal, called “Fast booking”. Further analysis of the DesiredEnd reveals that such an Ideal is achieved via a number of more specific Objectives, including “Service provider filtering”, “Caching”, “Simplify booking process”, and “Parallel processing”. These Objectives aim to, respectively, 1) filter out slow service providers, 2) provide a cache to store repeated information locally, 3) simplify the processes of booking by reducing the number of coordinated partners and automating some manual processes, and 4) interact with service providers in a parallel and asynchronous fashion. Objective 4, in turn, contains some lower-level Goals that are more specific and focused, which aim to achieve particular aspects
or parts of the DesiredEnd, and to contribute to the overall Objectives and Ideals. These Goals include “Parallel inquiries” and “Parallel booking”, which makes inquiries and reservation in a parallel and asynchronously fashion. Moreover, since the more specific Goals have enough information to specify the desired states in the reality, it provides clear external interface for service consumers to use a Capability to achieve the specified Desired End. Therefore, these Goals are associated with Capabilities, namely “Parallel Travel Enquiries” and “Parallel Travel Booking”.

The identification and derivation of these improvements and desired ends at different levels, namely Ideal, Objective, and Goal, helps to discover the achievable goals and desired capabilities, which collectively contribute to addressing the overall problem situation. The inter-relationship between the PSM and DEM creates a direct link between well-understood problems and rational solutions that can be reused in future situations. Altogether, they form an organic aggregation of Rational Images, which represents the reusable intellectual achievements about desirable and achievable intentions to improve complex problem situations. Therefore, Rational Image is an important cognitive resources to facilitate engineering problem-solving process.
5.3.3 Realisation modelling and Implementation process

As presented in the previous sections, in OASEF, higher order models such as CNM, PSM, and DEM are used to derive and capture important intellectual achievements within the world of mind. The World Images and Rational Images they form are critical engineering assets that can be utilised, or reused, to enable and facilitate other engineering activities.

Another important human cognitive activity in the world of mind is related to realisation in the conception process. As defined in Section 4.4.4, realisation is a cognitive process involves design and formation of conceptual plans and decisions that aim to satisfy the desired ends, and to improve the problem situations. Therefore, Realisation is associated with intellectual design of conceptual solutions.

The design of a solution has been a major concern in traditional software engineering and systems engineering. There are various approaches that aim to capture, model, and assist this process using specific modelling languages. For example, UML activity diagrams are often used to describe the internal structure and behaviour of business processes. UML sequence diagrams can be used to model the interactions among various participating parties. In OASEF, we use these UML models to conduct realisation and capture its outcomes. Specifically, the objective of OASEF realisation is to derive Capability Images, which are organic aggregations of capabilities in support of the rationalised desired ends. That is, the higher level Capabilities and DesiredEnds in the Rational Images need to be realised by lower level capabilities in a specific way.

The derivation of detailed level capabilities are conducted in Capability Design process of realisation. It is derived systematically according to the higher level Capabilities and DesiredEnds in DEM, which were derived in accordance with the problem situations in PSM. That is, for every higher level Capability and DesiredEnd in the DEM, there will be a corresponding Capability Design model that captures the required local capabilities that are organised in a specific way to realise such a desired end.

We use a scenario in FHSA case study to illustrate the realisation modelling process. In this case, a specific Capability Design model is created to achieve the DesiredEnds: “Close an existing FHSA account”. It contains interconnected lower level capabilities within an orchestrated structure, for the purpose of providing higher level capability to close a FHSA account. The local capabilities include a sequence of local capabilities such as “Check Eligibility”, “Request FHSA Closure Form”, and “Validate FHSA Closure Form”. Depending on the age of the applicant, remaining account balance, and whether the applicant is purchasing a house as the first home, one of the following three capabilities are required, that is, 1)
Figure 5.13: Internal logic of capabilities required for closing an existing FHSA account - realisation

"Send Account Closure Request to Business Operation Service" and then "Wait for Process Result", 2) "Withdraw Money", and 3) "Decline request". Finally, the processing result will be sent to the applicant, via capability: "Notify Customer Result". Note that this Capability Design model contains not only the participating capabilities, but also the orchestration logic that reflects the business process.

This Capability Design model is still at the conceptual level within the world
of mind. A collection of these models in accordance with Rational Images form Capability Images. These Capability Design models will be transformed into a more concrete form that is implemented via a specific technical platform. The Service Composition activity in the Implementation process is designed for this purpose. It systematically transforms every capability in the Capability Design model into concrete services, and orchestrates these services together according the specified composition logic. Using existing model transformation techniques and supporting tools, this process can be performed automatically. For example, the Capability Design models expressed by UML activity diagrams can be transformed into a more concrete formalism, such as BPEL artefacts. The latter can be directly executed in a run-time environment, in our case studies, the Apache ODE BPEL engine. The transformation process from UML Activity Diagrams to BPEL uses the MDD4SOA tools developed by Mayer et al. [2008]. The model transformation process in the implementation process is based on a capacity reuse mechanism that will be introduced in the next section. Examples of how this is achieved will be presented in Chapter 6.

5.4 Reusing engineering capacities

In the previous sections, we presented the overall structure of OASEF, and its modelling mechanism for internal processes and activities. The objective of this process is to form organic aggregations (see Section 4.1.2) of process resources including the interconnected inner processes and their outcomes. It means that the OASEF approach to services engineering needs to promote environmental fitness to address the STE challenges, that is, a high degree of complexity, uncertainty, and volatility. To achieve such an objective, the design of OASEF is guided by the foundation and guiding principles described in Section 3.2 and Section 3.3. In this section, we will present a specific design of OASEF in accordance with the guiding principle Maximise the reuse of engineering capacities specified in Section 3.3.4.

OASEF provides a mechanism to reuse engineering capacities to conduct its various inner processes and activities, which is called epitome. An epitome is a typical and justified means or capability to achieve a specific type of engineering purpose. It is generalised from proven examples, and is captured as a service that can be used at any time for the same type of purpose. For example, if a specific problem situation that is captured in a PSM is resolved by applying particular implementation, we can regard the latter as a working example to generate specific ends in reality to improve that type of problem situation. Such a working example is captured in an epitome that provides the capability to improve the problem situation. It can be identified and reused by anyone who is allowed to use this
Reusing engineering capacities

capability for solving the same type of problem at hand. Therefore, applying an existing epitome directly produces the outcomes without having to go through the particular activities. In the above example, the epitome generate specific artefacts of implementation that are able to improve the problem situation.

By associating means with ends, epitomes create Strong Correspondence between them. The Strong Correspondence here means that two elements are tightly associated with each other in a way that one would significantly influence or affect the other, and that the occurrence of one is often accompanied by the appearances of the other. The associations could be 1) causal relationships, such as a solution in implementation that leads to the resolution of a problem in rationalisation, 2) intentional relationships such as an intention to do something for particular purposes, or 3) perceptual relationships, such as projection of certain aspects of the reality in mind. This notion to deal with reuse is in accordance with the guiding principle: Use “soft” methods instead of “hard” ones to deal with ill-understood phenomenon (see Section 3.3.3). That is, we don’t focus on precise and determined patterns to realise reuse. Instead, reuse is in accordance with examples that may or may not work in the new situations. For the latter case, the epitomes create pseudo ends for reference only, on top of which the real ends can be derived. Nevertheless, the capabilities and knowledge of working examples are captured in epitomes, which could be of use partially or entirely.

The Strong Correspondence of epitome therefore can be used to link two OASEF activities together. In the example presented earlier in this section, the epitome provides the capability of creating the previously proven implementation to solve the same type of problem situation. Therefore, there is a Strong Correspondence between the outputs of implementation and the outputs of PSE.

An epitome is created by generalising a proven example of Strong Correspondence between two elements of OASEF engineering resources, which is a process called epitomisation. That is, the process of epitomisation creates epitome that links two OASEF element by working examples.

Although the OASEF activities are organised as a hierarchy, it is important to note that the composition of process activities is not fixed, but flexible, which can be bridged, bypassed, or altered, because of epitomes. Because the epitome creates a linkage between two OASEF activities, an instance of OASEF does not have to contain all of the inter-connected inner processes and activities to make a complete loop from the reality to the mind, and to the reality again. Instead, various activities can be flexibly inter-connected: some might be bypassed, some can be aggregated at higher level of OASEF process. Two activities can be potentially linked together, if there is an epitome that provides Strong Correspondence between

143
these two activities. Therefore, the actual activities that form a particular engineering process can be dynamically selected, from activities at different levels within the hierarchy, as long as the activities can be conducted manually, or there are existing epitomes that provides required capabilities to achieve the goal of such activities. The desired capability can also be selected among a number of epitomes provided by different stakeholders, who can perform the same activity separately, or in parallel, as different ways of doing things, or as backups. Furthermore, two epitomes can also be aggregated as a single one as long as the correspondences they provide are adjacent.

Figure 5.14: An example of using epitomes to realise flexible aggregation of OAP activities

Figure 5.14 demonstrates an example that uses epitomes to bypass certain activities. The process has Perception, Control, and an aggregated epitome called “epitome5”. The latter is an aggregation of several adjacent epitomes, from “epitome1” to “epitome4”, which creates a shortcut, or reuse, between a problem situation and the implementation of a solution.

Using this mechanism, an organic OASEF process can be formed as flexible aggregations of OASEF inner processes and activities that are either linked directly, such as DEE linked with PSE, or indirectly via epitomes, such as implementation linked directly with rationalisation. For example, since Conception consists of Rationalisation and Realisation, reusable capacities of Conception can
5.4 Reusing engineering capacities

```
PHBS_db.sql
--DROP TABLE T_CUSTOMER;
CREATE TABLE T_CUSTOMER(
CUSTOMER_ID INT NOT NULL GENERATED ALWAYS
AS IDENTITY,
EMAIL VARCHAR(500)
FIRSTNAME VARCHAR(500)
LASTNAME VARCHAR(500)
LOGIN VARCHAR(500)
PASSWORD VARCHAR(500)
)
PRIMARY KEY (CUSTOMER_ID)
)
DROP TABLE T_BHSCACCOUNT;
CREATE TABLE T_BHSCACCOUNT(
PHSAccount_id INT NOT NULL GENERATED
ALWAYS AS IDENTITY,
TITLE VARCHAR(500)
CREATIONDATE date
' BALANCE
```

Figure 5.15: An example of database schemas generated by an epitome

be encapsulated in an epitome, which generates required models, without having to go through an instance of either Rationalisation or Realisation to achieve the same effect. Therefore, this mechanism enables flexible aggregations of OASEF processes. Moreover, just like other intellectual organic aggregations, such as the Rational Images, a repository of epitome form valuable engineering resources that can be located and reused by other stakeholders.

Therefore, this mechanism provides a flexible way to conduct OASEF process, and to capture, store, and reuse proven engineering capacities.

To exemplify this, we present a scenario in the FHSA use case. A database schema epitome, which physically exists as a service, captures the specific capability to persist a customer information into a database. It takes a business information model from SA of the abstraction as an input, such as a class diagram for Customer, and produces a specific database schema.

The internal logic of providing such a epitome is based on a code generation template language provided by the Acceleo project [Acceleo, 2010], which conforms to the OMG MOF Model to Text Language (MTL) standards, and is developed as a part of the Eclipse M2T project [M2T, 2010]. The template language is used to retrieve required information from OASEF models and put it into a template for
creating database schema. The template is generated via the known examples of creating a Derby database schema for a particular class. It is then used to incorporate a given information model in the form of UML class diagram, and to generate appropriate database schemas. Figure 5.15 depicts an example of generated database schema based on a class diagram in FHSA case study. Given the class with various attributes such as email, first name, last name, the database schema was automatically created accordingly. Therefore, the technical competence to generate database schema is captured, and can be reused through epitomes.

In OASEF, a similar approach is used to realise epitomes to generate required outputs, which exist as services that can be accessed and used by other parties. Because of our focus on promotion high-level intellectual capacities (see Section 2.1.4), we will not elaborate the technical implementation of the epitome. Instead, we present our work from the perspective of incorporating non-technical concerns within the context of SOS.

5.5 Implementation and proof-of-concept tools support

In the previous sections of this chapter, we presented our approach to organise inter-connected engineering activities during services engineering, including the practical modelling mechanism and capacity reuse mechanism. These mechanisms are supported by proof-of-concept tool support, in order to implement the OASEF design and use it to assist services engineering. A range of supporting tools are organically aggregated within an integrated supporting tools environment, which we call Integrated Process Environment for Organic Aggregation Process (IPEOAP) [Wang, 2010a,b]. It contains a range of complementary tools, either from third parties, or developed by ourselves. It is developed using Java programming language on top of Eclipse IDE and EMP. A range of graphical model editors are developed to view, create and modify models for various OASEF activities, such as CNM, PSM, and DEM.

IPEOAP uses EMF framework as its unified modelling environment. It supports all OASEF meta-models for its inner processes and activities, such as the DSL and general-purpose UML and BPMN modelling. Various OASEF models, especially those at a higher level of abstraction in the world of mind, are fully supported in IPEOAP, which enables modelling of GA, PSE, DEE, and realisation in a graphically supported environment. IPEOAP not only supports model manipulation as other MDE approaches do, but also provides a formalism to promote the notion of organic aggregation by realising rationalising and justifying
high-level engineering design decisions.

Moreover, as described in Section 5.4, the linkage between high-level models and lower-level implementation is derived, captured, and automated through universally accessible repositories of *epitomes*, the engineering capacity reuse mechanism introduced in the previous section. Through *epitomes*, OASEF provides a practical mechanism to implement systems according to the higher level intellectual achievements, such as CNM, PSM, and DEM.

Therefore, with support from IPEOAP, the integrated tool supporting environment, OASEF inner processes and activities can be conducted in a unified fashion. That is, every OASEF model is graphically manipulated within the tool sets, and is stored according to the same meta-meta-model. These models can reference one another because of the same meta-meta-model they use. Furthermore, these high level OASEF models are systematically, and in our case studies, automatically, transformed into implementation code or documentation.

As mentioned in the previous section, *epitomes* and modelling techniques are integrated within our tool support environment. Specifically, *epitomes* are supported by EMP model transformation techniques, in our case study, the Acceleo tools that conform to the OMG MOF MTL standards, developed by the Eclipse M2T project [M2T, 2010]. IPEOAP also incorporates a range of other tools such as UML2 Tools and BPMN2 from Eclipse MDT project [MDT, 2010], BPEL editor from BPEL Designer Project [BPEL, 2010], and MDD4SOA tools developed by Mayer et al. [2008]. Therefore, a wide range of tools have been coherently integrated with our OASEF modelling mechanisms based on EMF. Altogether, these tools allow easy manipulation of OASEF models and automatic generation of concrete systems based on them. From our experience in the case study, the majority of engineering efforts is to design and manage the abstract model such as PSM and DEM, and general-purpose UML or BPMN models. However, with the tool support from IPEOAP, the modelling tasks have been greatly facilitated.

Figure 5.16 shows a snapshot of the IPEOAP tool support environment. It includes project navigation pane on the left-hand side, which can be used to manipulate various OASEF resources, such as OASEF models and transformation engines. These resources can be opened in corresponding graphic editors on the right-hand side of the figure. Within the editor, models and models elements can be manipulated accordingly, including creating, editing, moving, and deleting model elements.

The case studies are supported by IPEOAP. As shown in the previous sections of this chapter, various OASEF models were created, which systematically led to generation of concrete systems. We use some figures to illustrate what was
Figure 5.16: A snapshot of IPEOAP tool support environment - PSE modelling

achieved using this tool environment.

Figure 5.17 depicts two generated web pages for the FHSA bank institution. The page on the left-hand side is used to collect customer information, and to create a new customer account. It was generated by `createCustomerAccountEpitome`, which takes an instance of customer information, and generates the web pages and database entry to support the functionalities pertinent to creating a new account. The web page on the right-hand side was generated by `displayCustomerAccountEpitome`, which generates the web pages and business logic to return a list of available customers.

Appropriate web services are generated to realise a range of services during implementation. This is conducted according to the Capability Design from the realisation process. In this case, these services provide functional capabilities to get database access to query the existing customers, generate a user interface to display customers, generate business layer control logic to determine the availability of customer information, as well as wrapping Axis 2 web services and deploying them to run-time environments. Eventually, Java code and database
scripts are generated and deployed to an application server that physically hosts the software. As a result, a functional web system is systematically derived, which can create a new customer account after a user submits an online form.

5.6 Summary

In summary, the specific design and mechanisms used in OASEF to facilitate services engineering are presented in this chapter, which provides an answer to research question V: **What practical mechanisms do we use for effective services engineering?** in the context of concrete SOS. As an overview, OASEF cohesively synthesises some inter-connected elements, such as the theoretical foundation, the conceptual OAP process model, and a more concrete model-driven method. An integrated environment of supporting tools, called IPEOAP, is also developed to facilitate various OASEF activities such as manipulation of models and generation of desired systems.

More specifically, a particular model-driven methodology of MDE is organically integrated with OAP at all levels. It provides an effective means to capture knowledge and outputs of various OASEF activities, and to utilise or transform them to generate SOS.

A range of meta-models for other OAP activities such as CNM, PSM, and
DEM are designed in an integrated process environment. We also use a general purpose modelling language such as UML to model Abstraction and Realisation. For example, UML Class Diagrams and Activity Diagrams are used to capture information and behaviour models within a business domain. More specifically, UML diagrams are used to store information of sensation and system abstraction. PSM and DEM are used to capture outcomes of Rationalisation. Extended UML diagrams, UML activity diagrams, state machine diagrams, or BPMN can be used to conduct Realisation. Moreover, these models can be translated, transformed, or converted to code and other artefacts such as BPEL, Java code, or other documents.

The next chapter will evaluate OASEF via a couple of case studies. It aims to answer the last two research questions: question VI: How do these mechanisms perform in real situations? How do we assess them? Question VII: Are improvements needed? If so, what are they? These questions are important, because, after all, the ultimate objective of this research is to design and prototype a services engineering approach that works in practice.
Chapter 6

Case studies and evaluation

All technology should be assumed guilty until proven innocent.

David Brower

Contents

6.1 Background ........................................... 153
6.2 Managing complexity within business domains .......... 155
  6.2.1 GA models: modelling complex general knowledge ...... 156
  6.2.2 SA models ........................................ 159
  6.2.3 Rationalisation models ................................ 161
6.3 Linking conceptual models with technical artefacts ....... 163
  6.3.1 Realisation ........................................ 163
  6.3.2 Action ............................................. 165
6.4 Engineering capability encapsulation and automatic generation ............ 165
  6.4.1 An example of epitome to generate information service .... 168
  6.4.2 Generate Flight Information Service using epitome ....... 168
6.5 Change and adaptation .................................. 171
6.6 Further analysis and discussion .......................... 176
  6.6.1 Observed strength ................................. 177
  6.6.2 Identified issues .................................. 178
Chapter 6: Case studies and evaluation

We have so far presented our new approach to services engineering, called Organic Aggregation Service Engineering Framework (OASEF). Its structure, theoretical foundation, and conceptual process model were presented in Chapter 3 and Chapter 4. In Chapter 5, we presented the overall structure of OASEF and its specific mechanisms. Briefly, OASEF incorporates a conceptual process model called OAP, and mechanisms to model and reuse both high level intellectual knowledge and capacities, and lower level technical competence. Moreover, a notion of organic aggregation is applied to form a coherent process that is arranged in a flexible way, and to form valuable aggregations of knowledge and engineering capacities. A specific mechanisms is designed to model engineering knowledge and capacities, and to re-use them. This was achieved by integrating a specially designed modelling technique with a Epitomision process, our capacity reuse mechanism (see Chapter 5). As introduced in the previous section, a prototype of an integrated tool environment has been developed to implement and facilitate the OASEF design.

This design aims to realise our research objective, that is, to address the challenges of STE in the context of services engineering. To achieve such an objective, OASEF design is based on the foundation and guiding principles tailored for these challenges (see Chapter 3 and Section 5.1). Specifically, OASEF is based on these principles: Environment-driven view (Section 3.3.1), Maximise the reuse of engineering capacities (Section 3.3.4), and Everything is driven by models (Section 3.3.5). Moreover, it aims to promote another two fundamental software engineering principles: higher level of abstraction and the separation of concerns. By applying the empirically-proven engineering principles, the design of OASEF aims to address the fundamental problems introduced in Section 1.2, that is, a lack of suitable processes, inadequate focus on human perspectives, and insufficient reuse of engineering knowledge and capacities.

These guiding principles and practical mechanisms would help to address the challenges of complexity, uncertainty, and volatility. However, would our approach based on these principles actually work in practice? Does it achieve the design objectives to address identified problems and challenges discussed?

This chapter presents our research efforts to answer these questions, which are in association with the research question VI (see Section 1.4.2): how does the proposed approach perform in real situations? Although specific technologies and techniques are used during the case studies, it is not the main purpose of this thesis to report the technical and implementation detail. Therefore, we will refrain from describing too much technical terms and implementation detail here. Simply put it, we will stay focused on presenting the higher level engineering issues we are dealing with, that is, exploiting high level human intellectual capacities and
encapsulation of implementation capacities.

The remainder of this chapter is organised as follows. Section 6.1 provides background information about these case studies. Section 6.2 analyses how OASEF manages the challenge of complexity, in the context of the online travel business domain. Three OASEF activities are examined here, namely the GA, SA, and rationalisation. Section 6.3 presents how to utilise high level models from perception and rationalisation processes to design conceptual models of realisation, which, in turn, are automatically transformed into more concrete BPEL models during the implementation activity. The latter are executable in a runtime environment as a part of concrete systems in reality. Section 6.4 illustrates how epitomes are realised, which provide a mechanism to encapsulate capabilities in the OASEF process (see Section 5.4). This section presents the detail about capturing reusable engineering capacities and how to automate the process to generate concrete systems. In Section 6.5, changes are introduced in OASEF higher-level models, which illustrates how to evolve business needs towards a more complex business scenario using high level models and captured engineering capacities. Finally, Section 6.6 presents some observation during the case studies, including analysis of some identified strengths and shortcomings.

6.1 Background

The importance of evaluating engineered systems based on formulation of measures of effectiveness has been well recognised and practised for a century [Hall, 1962, Chestnut, 1967, INCOSE, 2008]. As a fundamental engineering principle, pragmatic effects of systems need to be monitored and assessed, in order to provide valuable feedback to make continuous improvent to the systems. This is supported by Krogdahl et al. [2005], who states that the evaluation and feedback processes form continuous loops within operational, economic, and environmental constraints, and would enable and support continuous system improvements. Therefore, this section aims to form such a feedback loop through the research activity of measure and analyse in the Focusing and Reflecting phase (see Section 1.4.2).

Therefore, we develop two proof-of-concept case studies in order to assess the design of OASEF. Specifically, the objective is to understand and evaluate whether OASEF helps to derive and manage SOS, and to tackle the identified challenges presented in Chapter 2. The IPEOAP tool supporting environment described in presented in Chapter 5 are used to facilitate this process.

According to Section 1.4.3, these case studies are Measure and analyse
activities in the *Focusing* phase. Our main concern is not about how well OASEF performs, but whether it works in principle according to our research objectives. Therefore, the case studies focus on proof-of-concept assessment, as opposed to proof-of-performance [Snyder, 1994]. Moreover, due to a lack of access to real world projects in our research setting, they are controlled case studies that are based on some purposefully designed scenarios, for the purpose of assessing particular aspects of our design. These scenarios are designed in the context of real world project, and the use cases reflect the present industry practice and real user requirements that are publically accessible and verifiable. The complexity involved in the case studies matches that of a medium-size enterprise project according to our previous industry experience.

Specifically, as briefly introduced in Chapter 5, the first case study is in the context of an online travel booking business, which is referred to as Online Travel Booking (OTB) in this thesis. It is a typical scenario used in the SOC, e-business, marketing, and management research communities [Curbera et al., 2002, Park et al., 2007, Nepal et al., 2009, Hewett and Kijsanayothin, 2010]. Therefore, it can be easily understood by people with either technical background or business and management one. Moreover, since the online travel business is a typical scenario used in SOC community, using it as a test bed helps to present OASEF in a context comparable with that of other approaches. Another reason we choose this business domain is because of the widely available resources that are made public to anyone. It presents less significant barrier compared with other business domains such as finance and banking. Because of the availability these resources, we were able to design a complex business scenario in terms of capturing complex knowledge of business domain, which will be presented in Section 6.2.

The second case study is based on the First Home Saver Accounts (FHSA) scheme that was introduced to Australia by the federal government in 2008 to help residents purchase their first homes with the aid of government payment contributions and tax benefits [Wang, 2010a]. The empirical experience from the case studies was used to assess design features and their effectiveness, and to identify strengths and limitations of the proposed solution. This business domain is chosen because it provides a good business scenario with business processes that show a high degree of complexity. There are complex interactions among various stakeholders, including residents, various government agencies such as the treasury department and the taxation office, commercial bank institutions, superannuation organisations, and their customers. As an example, a commercial bank institution provides FHSA services to eligible customers, and enables them to create and manage their unique FHSA accounts. The treasury department and taxation office manage the government contributions and taxation aspects
6.2 Managing complexity within business domains

of such accounts. Customers can switch over to a different banking service provider as they wish, and can also transfer their funds designated superannuation institution under certain conditions. The FHSA business settings also provide a socio-technical environment that is rich enough for us to investigate the nature and characteristics of such an environment, and the inter-relationships among both social and technical factors, which is useful for the purpose of this research (see Section 1.1.1 and 2.1.4). Moreover, the legislation requires that all regulatory information should be made accessible to the public. It enables us to easily set the stage and to design the experiment, which is based on our previous industry experience on online banking applications.

Therefore, by working on concrete case studies in the context of real business settings, the features and value of this research is observed and understood; its effectiveness in various situations, such as increased complexity and evolving changes, is analysed and assessed; the problems and limitations are also identified, which provides a feedback look within the framework of OASEF.

6.2 Managing complexity within business domains

As described in Section 2.2, one of the great challenges of services engineering is to manage the increased complexity in the socio-technical environment. Therefore, we design a controlled scenario in order to assess whether OASEF is able to address this challenge in real-world settings. In particular, we want to understand whether OASEF helps to manage a high degree of business complexity. This knowledge domain is chosen because the online travel business can involve a high degree of complexity.

One aspect of the complexity comes from the rapidly evolving business models. Technology has quickly changed the status of tourism industries, such as airline travel, ground transportation and accommodation [O'Connor and Murphy, 2004]. Electronic means have become popular for people to discover and purchase travel products and services [Park et al., 2007]. Travel services, which used to be provided through traditional off-line channels, are increasingly generated from service providers' own websites or online travel intermediaries, such as Expedia and Travelocity [Carroll and Siguaw, 2003]. According to marketing research, for example, 66 percent of US Internet users have experience of making online travel reservation or buying travel service online [Pew Research Center, 2010].

However, the emergent characteristics of online travel business are poorly understood. In order to understand the nature of such complexity, we deliberately delve deeper into the online travel domain. As a result, the knowledge captured in
this section might go beyond the scope of engineering.

For the purpose of this case study, we developed some imaginary scenarios, which involve an online travel portal that provides web interfaces for various services from different service providers. This portal provides travel-related information services including airlines, hotels, car rental, and weather reports. It also allows customers to make online reservation and payment for the chosen products and services. The engineering tasks we want to achieve include understanding the complex *problem situations* and identifying *desired ends* through OASEF PSE and DEE in the process of *Rationalisation* (see Section 4.5.2). Although this web portal is not real, the business domain we investigated involve the state of the art research in online travel business. Therefore, these tasks deal with the real world problems instead of imaginary one.

### 6.2.1 GA models: modelling complex general knowledge

We start with GA modelling of general knowledge about the *reality*. Online travel business is a complex social activity that involves interacting factors in multiple disciplines such as e-commerce, marketing, psychology, and customer behaviour. Therefore, it is important to obtain better understandings of the nature, characteristics, general rules or regular patterns of such systems in *reality*, in order to obtain accurate *abstraction* models during *perception*.

In order to understand the behaviour of online customers when they purchase a product or service online, it is important to understand the correlation of relevant factors that influence the customer decision-making processes and behaviour patterns. Exploration of domain knowledge in online consumer behaviour and marketing reveals that customer behaviour in an online context is determined by: 1) pre-purchase perception of attributes such as “perceived usefulness”, “perceived ease of use”, “perceptions of value”, and “perceived risk”; 2) pre-purchase user attitudes such as “trust”, “privacy concern” and “Internet affinity”; and 3) post-purchase attitudes such as “customer satisfaction” and “loyalty intentions” [Taylor and Strutton, 2009]. Moreover, according to Parasuraman et al. [2005], “perceived value” includes “Price”, “Overall value”, “perceived control” and “perceived convenience”, whereas “loyalty intentions” include “Positive word of mouth”, “Recommend to others”, “Encourage others to use”, “First choice for future”, and “Do more business in future”. Therefore, customer intentional behaviour is associated with a range of influential factors that inter-relate to one another. This type of relationships can be modelled as *Aspect* in the CNM, as introduced in Section 5.2.1. Accordingly, we create a CNM for capturing online customer behaviour factors, which is depicted by Figure 6.1.
6.2 Managing complexity within business domains

![Diagram showing CNM (Complex Network Model) for customer satisfaction and behaviour]

Figure 6.1: An example of a complex CNM that captures knowledge about customer behaviour - Managing complexity

This CNM contains a network of interconnected Aspects. The essential ones are "pre-purchase perception of attributes", "pre-purchase user attitudes", and "post-purchase attitudes". Each of these Aspects has subordinate Aspects, according to Taylor and Strutton's observation. For example, "post-purchase attitudes" has a subordinate Aspect called "satisfaction" that refers to the factor of customer satisfaction. Moreover, Parasuraman et al. [2005]'s insights are also incorporated in the CNM. That is, another two Aspects are considered essential to affect customer purchasing behaviour, that is, "perceived value" and "loyalty intentions". This is a good complement to Taylor and Strutton's analysis. As the user perception of the value they expect to obtain from the online behaviour would determine whether they are going to feel satisfied or not. On the other hand, customer intention for repeating transactions would affect their future purchase decisions and behaviour. These top level Aspects are hence justified either by evidence, common sense, or empirical data, which are represented in the research paper. Moreover, the "perceived value" Aspect contains subordinate ones such as "price", "overall value", "perceived control". So does "loyalty intentions". This network of influential factors of online customer behaviour captures the knowledge and wisdom about what affects a customer's purchase behaviour. This understanding is important to guide other OASEF modelling, such as rationalisation, which will be illustrated later in this section.
Moreover, CNM can be broken down into subordinate model if the number of elements within one model increase to a level that people cannot easily comprehend. That is, one element of a CNM can have its subordinate elements in another CNM, instead of putting them altogether within a single model. We call the second CNM a subordinate CNM. To demonstrate how this is achieved, we further explore the subordinate factors of the “satisfaction” Aspect in Figure 6.1. The exploration process in search for influential factors of customer satisfaction reveals important Aspects that lead to improved customer satisfaction. The captured knowledge is depicted in Figure 6.2.

Figure 6.2: A subordinate CNM that captures the influential factors related to customer satisfaction in OTB case study-Managing complexity

Specifically, construction of this subordinate CNM starts with exploration of correlation with satisfaction. Research reveals that service quality and customer perceived value are two major factors that have positive impact on customer satisfaction [Zeithaml et al., 1996, Bai et al., 2008, Park et al., 2007]. Service quality in the online context is regarded as “the extent to which a website facilitates efficient and effective shopping, purchasing and delivery” [Zeithaml et al., 2000]. Evidence shows that, in order to survive the competition and prosper in increasingly competitive business environment of online travel, it is crucial to understand and improve the online service quality, and therefore to improve customer satisfaction [Zeithaml et al., 2002]. Parasuraman et al. [2005]
generalised seven dimensions relating to online service quality, such as ease of use, efficiency, problem handling and compensation, and system availability. Other researchers also proved that service quality can be captured and measured through some perceptual dimensions, including ease of use, information content, responsiveness, fulfilment, security privacy, and visual appeal [Law et al., 2009, Park et al., 2007, Ho and Lee, 2007, Scharlr et al., 2003, Long and McMellon, 2004]. More recently, trust is increasingly regarded as an important concern affecting customers’ decision-making with regard to whether use a service or not [Urban et al., 2009, Kim and Stoel, 2004, Milne and Boza, 1999].

As shown in Figure 6.2, the new CNM captures the above knowledge about correlated factors that influence or determine customer satisfaction, which provides critical perception achievement that contribute to other OASEF activities, such as rationalisation and realisation in the online travel context as shown in Section 6.2.3 and 6.3.1. As general knowledge that applies to other online business, it also serves as reusable intellectual assets that can be utilised to serve other engineering activities, such as identifying problems in online B2C business. In the context of OASEF, these abstract artefacts are used and referenced in successive OASEF activities, which will be illustrated in Section 6.2.3.

6.2.2 SA models

In Section 5.3.1.2, we presented a simple information model during SA process. Such a model only shows the information structure of individual class of objects. In practice, information model can be complex and consists of hundreds or thousands of different classes, with complex relationships among them. Therefore, the UML class diagram can deal with such complexity in practice. To examine how OASEF deals with this type of complexity, we designed a more complex online travel system. In this section, a more complex information model is shown in Figure 6.3. It represents related messages that flow among various participants such as Customer, TravelPortal, FlightProvider, and HotelProvider. For example, it describes what type of information Customers need to provide in order to make an enquiry on flight and hotel. It also provides a representation of information returned by the service provider that describes the service item. A class called HotelInfo represents a type of messages provided by HotelProvider. It has attributes that describe the queried hotel, such as the name of the hotel and its address. The model also reveals the relationship between HotelInfo and the rooms it provides that is modelled as RoomInfo.

As UML class diagrams have been used in industry for reasonably long time, especially in large-scale SISE projects, their effectiveness to manage complex
Figure 6.3: A complex Abstraction model in OTB - Managing complexity

information models in SA has been empirically verified. The abstraction provided in SA helps to understand and communicate information about complex systems.
6.2 Managing complexity within business domains

6.2.3 Rationalisation models

Before we start to analyse the problem situation, we first look at the environmental information of our online system. That is, the high-level expectation and purpose of customers. These contextual purposes often give good starting points for PSE. Analysis reveals that the rationale of customers to use online travel services is to save money and time, that is, to reduce the cost of information acquisition and business processing. Moreover, information and business processing is expected to be accurate and up-to-date due to utilisation of electronic means. From the service providers’ point of view, the volume of acquired information and accomplished business transactions per time unit is expected to increase. Ultimately, the purpose of providing the online booking systems is to maximise the business profits by providing required value to customers with satisfied qualities, whereas the expectation of service consumers is economical, convenient, efficient, and reliable travel service.

This analysis identifies a number of contextual Purposes in PSM, which are identified based our own knowledge and experience. However, such a discovery process is not systematically supported, and is subject to individual’s analysis capacities. A better approach is to utilise the general knowledge from GA pertaining to customer behaviour in online business settings, which was illustrated in Figure 6.1. That is, high level PSM Purposes can be derived based on the Aspects within the CNM described in Section 6.2.1. For example, to analyse the problem situation, one needs to understand that a customer’s purpose would be based on pre-purchase perception of attributes Aspect, as well as its subordinate ones, such as “perceived usefulness”, “perceived ease of use”, “perceptions of value”, and “perceived risk”. Moreover, other factors related to pre-purchase user attitudes, such as “trust” and “privacy concern” could also be considered in the PSM. Therefore, the Purposes in PSM could be systematically examined and derived according to the CNM. This approach is better than the one we present initially, as it is based on justified general knowledge captured in the GA, instead of on subjective speculation of one’s own.

By systematically identifying violation to all higher-level purposes according to observation in sensation, Problems are identified in PSM. For example, online travel booking takes too long is identified as the observation that violates the purpose Quick Responsiveness.

There is another way to identify and analyse problems, that is, to reuse general knowledge from other systems. For example, in practice, many issues have been found in the online travel business which impede greater success towards its full potential. Practitioners found out that although progress in infrastructure
technologies such as SOA has made it relatively easy to build and manage online travel services, the customer satisfaction levels of generated systems remain low. According to American Customer Satisfaction Index, a national economic indicator of customer feedback of service quality in the United States, the online travel category of e-commerce has not made any progress in terms of service quality and customer satisfaction since 2002. The satisfaction level has remained 77 percent for 8 years, compared with much higher satisfaction level through offline channels [Freed, 2010]. Therefore, it makes sense to start to look at customer satisfaction in our imaginary system.

The low level of customer satisfaction level is identified as a key problem in the system. The next step is to discover what other problems are related to it. The best way to conduct this is according to GA in Section 6.2.1. As the root reason for unsatisfactory online travel service is often related to non-technical issues such as poor service quality and privacy [Park et al., 2007], we further analyse this problems according to various non-technical service qualities, as depicted in Figure 6.2.

The high level analysis of problem situations is modelled in rationalisation as PSM, which was depicted by Figure 5.11 in Section 5.3.2. As demonstrated, PSM is derived according to abstraction models, and reflects the internal dependencies of various influential factors within a problem situation. In OASEF, PSM helps to discover and capture problems during business analysis, and is used to drive successive engineering activities such as generating DEM, which is demonstrated Section 5.3.2.

It is important to note that the application of rationalisation differs from traditional requirements engineering in terms of scope, mechanism, and "softness". Traditional requirement elicitation, analysis and management usually takes a rigid and "hard" approach, which is based on an assumption that there are requirements, clearly-definable and stable, out there to be collected in all situations. However, we believe such an assumption is not always valid in an open environment, where situations are complex, volatile, and uncertain. The common concept of "requirement" does not involve or imply "soft" intentions and desires that involve certain degree of uncertainty in problem situations, which, undeniably, also influence the way engineering plans and design are derived and carried out. On the contrary, our approach provides a specific means to enable and support discovery, understanding and utilisation of these aspects of engineering. That is, it incorporates the concept of desired end in the context of identified problem situations, and systematically provides sensible targets for realisation such as engineering planning and design. Its aim is to improve the problem situations to a satisfied degree, as opposed to achieving a "we-must-do-THIS" type
of “requirements” [Wang, 2010a].

In situations where requirements are clear, relatively stable, and well-understood, the purpose and function of this approach would be the same as traditional requirements engineering, although represented differently.

### 6.3 Linking conceptual models with technical artefacts

The abstract models in abstraction and rationalisation are further linked with more detailed design in realisation. Using the concept of service-orientation, the activities in realisation model can be automatically transformed into an executable implementation model in the form of BPEL. This approach simplifies the management of coarse level design and implementation, and is suitable for a design which has all its detailed requirements satisfied by service providers. This section demonstrates how it is achieved in the online travel booking scenario.

#### 6.3.1 Realisation

realisation involves conceptual planning towards more concrete and more detailed design of action to achieve the desired ends. The Capability and DesiredEnd in DEM are further elaborated with more detail here, which leads to models that contain lower levels of abstraction towards the concrete reality. The process includes two types of elaboration: top-down and bottom-up. The former analyses the higher level capability, and designs necessary lower level capabilities and their connections, whereas the latter aggregates the existing service from the perception models in certain manner to meet the desired ends. These two approaches can be used separately or together that forms a hierarchy of desired capabilities that work as a whole to achieve the desired ends. The UML Activity Diagram meta-model [OMG, 2009b] is used to capture the complex internal structure of a higher level capability.

For example, the Goal and Capability “Parallel Enquiries” from rationalisation are associated with a realisation model. It is modelled using the UML4SOA extension to UML Activity diagram that is depicted in Figure 6.4. It contains interconnected activities including “TravelEnquiries”, “searchFlight”, “searchHotel”, “searchTaxi”, and “sendEnquiryResults”. More specifically, “TravelEnquiries” activity has a receive stereotype, which indicates that it waits for an incoming enquiry message to get travel package information; “searchFlight” activity will interact with the “FlightProvider1”, which provides the flight information “searchFlightResult” according to the request information “searchFlightRequest”; “search-
Hotel" interacts with the HotelProvider1, which provides the hotel information “searchHotelResult” according to the request information “searchHotelRequest”; whereas “searchTaxi” interacts with the “TaxiProvider1”, which provides the taxi information “searchTaxiResult” according to the request information “searchTaxiRequest”; “sendEnquiryResults” activity returns the aggregated travel information to the “Customer”. Note that the “searchFlight”, “searchHotel”, “searchTaxi” activities are conducted in parallel, and have a stereotype “send&receive”, which indicates that they all have input and output messages.

Figure 6.4: Using UML activity diagram in realisation model in OTB case study - Realisation

An activity in a model in realisation can be further elaborated with subordinate activities, which forms a hierarchy of models. It also allows exception handling and event handling, which means if something of significance happens during these activities, some specific handling activities can be invoked.
6.3.2 Action

Implementation is about actually doing things or carrying out plans according to realisation models, in pursuit of achieving DEM from rationalisation. As long as the realisation models are well designed and fully elaborated, there is always a way to generate a concrete implementation that can be carried out in action to realise the previously derived abstract models. If all activities in realisation models have corresponding concrete service providers as modelled in abstraction, the transformation between realisation and implementation can be simplified as automatic transformation between UML and the BPEL models that can be executed on a BPEL engine, due to the semantic alignment between them.

Figure 6.5 depicts a realisation model for travel booking. It waits for an incoming booking request from the Customer, who provides the detailed request information bookingRequest. Once a valid request is made, the system interacts with PaymentService to confirm if the provided payment information is valid. It then continues to book flight from a FlightProvider, providing the request information bookFlightRequest. If the flight booking is successful, it then continues to book the hotel room from HotelProvider by providing roomBookingRequest information. After getting confirmation information FlightBookingResult and RoomBookingResult, it notifies the customer about the booking status. In this scenario, we assume that all activities have corresponding service providers, such as Quantas and Wotif booking services.

During the action phase, the above realisation model are transformed into a more concrete implementation model in the form of BPEL. Figure 6.6 shows the generated BPEL model. Which contains concrete service interactions between designated PaymentService, FlightService and HotelService with physically accessible address. Such a BPEL model can be executed in a BPEL runtime engine such as Apache ODE, which takes input message from customer, executes the successive activities until receiving all require response from the service providers, and replies the customer with booking status information.

6.4 Engineering capability encapsulation and automatic generation

In OASEF, the realisation models are sometimes elaborated as detailed engineering tasks, which have no corresponding service providers to satisfy their needs. In this case, new services need to developed to achieve the specific engineering tasks. These new services are generated by well-captured generation capabilities,
Chapter 6: Case studies and evaluation

Figure 6.5: Using UML activity diagram to model realisation II - Realisation
called epitome. It is a mechanism to encapsulate capabilities, and is able to generate predictable and previously verified implementation based on an input of a type of abstract model. Its internal logic represents a typical way or process to generate engineering artefacts for a type of realisation task. For example, let’s suppose that we have a realisation task “generate flight information service” to achieve a DEM goal called “provide flight enquiry service”, which is used for provision of a flight information service that can be used to maintain the flight information by
6.4 Engineering capability encapsulation and automatic generation

Figure 6.6: Automatically generating BPEL using realisation models - Implementation

the information providers, and to query flight information by the customers. This realisation task can be achieved by an epitome called “InformationServiceEpitome” that encapsulates typical process and logic to generate necessary code to provide the required information service for a type of entity, in this case, the flight information. The internal structure and interaction of this epitome is presented in the next section.
6.4.1 An example of epitome to generate information service

The epitome also has correspondence to an intention in DEM, in this case "provide information service", which means that it has to serve this particular purpose. In addition, it is itself a realisation model and contains internal activities that collectively describe the typical process to achieve such a purpose, for any type of things, such as flight and hotel. These entities are represented as static information models as exemplified by Figure 5.9 in Section 5.3.1.

Figure 6.7 depicts realisation model of the epitome "InformationServiceEpitome". It encapsulates the capability to create information service such as Flight service. Specifically, it involves a process that starts with receiving a request to generate an information service, followed by an activity to create the static information models using a UML class diagram, such as a model for Flight information. If such a model already exists in the abstraction phase, it can be referenced directly instead of creating a new one. Once the information model is created, it needs to be normalised so that it is ensured to be valid and contains required structure. Then the normalised model will be used to trigger two streams of actions. One is to derive a user interface model and generate the user interface artefacts to access the service. Whereas another stream is to generate business function to manipulate instances of the specified information models, to generate the actual service with required functions, and to generate a client access Application Programming Interface (API) for such service.

Just like other realisation models, activities in epitome can have their internal structures and interaction logic, which enable a better organisation of activities when they get more complex. For example, one of the activities in 6.7, namely "generateService", involves subordinate activities as depicted in figure 6.8. Generating a service involves a range of activities. The previously generated business code needs to be checked, compiled and deployed as a component. In the mean time, a service run time environment needs to be installed, initialised and kept running. The required services need to checked and generated. After these activities, the generated services and required business components can be deployed and published in the prepared service runtime. The client code to access the deployed services can be generated to facilitate service consumption.

6.4.2 Generate Flight Information Service using epitome

An epitome encapsulates the entire logic and necessary steps to implement the realisation design. The activities involved in epitome can also be automated because they are all represented as functional and autonomous services. Given
a working service environment and required information models, an epitome can automatically generate desired implementation artefacts that satisfy the desired end for identified problem situation. For example, the realisation task "generate flight information service" is implemented using the epitome described in Section 6.4.1. It takes a flight information model as depicted in Figure 6.3 of Section 5.3.1.2, and automatically generates the required flight information service to
Chapter 6: Case studies and evaluation

Figure 6.8: The subordinate realisation model of `generateService` activity in epitome "InformationServiceEpitome" - Epitome

manipulate flight information. The generated fully functional service is depicted in Figure 6.9.

This scenario demonstrates the mechanism of epitome to encapsulate the typical engineering capability during implementation phase. It shows that, by using realisation models, abstraction model, and services, epitomes provide an effective means to understand, specify, and conduct necessary engineering activities during implementation phase, which removes tedious manual process and improves productivity. It also facilitates agile response to business changes, which will be demonstrated in the next section.
Figure 6.9: The automatically generated web service interface based on normalised abstract models - implementation

6.5 Change and adaptation

Changes made to various models and processes during engineering activities are inevitable due to many factors, such as response to environmental or internal events, correction or improvements made to existing systems or outputs of previous engineering activities. For example, during SA of abstraction, we have derived a simple information model as described in Section 5.3.1.2. This section shows an example of how changes are made to this information model with systematic support for system regeneration.

Figure 6.11 depicts some changes made to the information models of abstraction. Specifically, the name of FlightInformation is changed to Flight, and some of its attributes are also changed. A new class called TaxiInfo is added with a number of attributes. The name of RoomInformation is changed to Room, which is now a child class of a new class called Hotel. This change represents typical evolution of business requirements. The original class diagram is modified using the provided
Figure 6.10: An example to make changes to high order abstract models - abstraction

UML tools.

The change in information models is picked up by the epitome, which changes the corresponding user interface model according to the underlying mapping rules between information models and User Interface (UI) model. It results in an automatically updated UI model as depicted in Figure 6.11, which shows that the name of Flight entity is updated, and new properties and their rendering interface are added. Other changes made to Hotel and Room are also reflected in the derived UI model.

Similarly, the generated web services are automatically updated by the web service engine, as depicted in Figure 6.12. The name of the services are changed accordingly, and new services to maintain the added entities are created. In addition, the WSDL description of these services also shows that the message types of these services are updated according to the new attributes.

This scenario exemplifies the way OASEF deals with change and adaptation by reusing both high level intellectual achievements, and low level implementation competence. It is achieved through the integration of a specific model-driven
6.5 Change and adaptation

Figure 6.11: The automatically generated user interface model based on the modified abstract models - realisation

approach and a capacity reuse mechanism, the epitomisation. On the one hand, the high level intellectual achievements are captured in separately derived, but inter-connected higher level models, such as the information models and CNM from abstraction, PSM from PSE, and DEM from DEE (see Section 5.3, Section 4.4, and Section 4.5). On the other hand, the low level implementation competence is captured in epitomes, which is a mechanism to model and reuse engineering capabilities (see Section 5.4).

As seen in this example, epitomes link the high level OASEF models in the world of mind with the low level artefacts in the world of reality. That is, if a specific implementation proved to be the best way to deal with certain problem situations, the capability to provide such a solution in implementation is captured and generalised in epitomes. They can be reused to generate concrete
Available services

**FlightService**
Service EPR: http://127.0.0.1:8080/axis2_1.5/services/FlightService

**Service Description:** FlightService

**Service Status:** Active

**Available Operations**
- createFlight
- findFlightByld
- findAllFlights
- deleteFlight
- updateFlight

**RoomService**
Service EPR: http://127.0.0.1:8080/axis2_1.5/services/RoomService

**Service Description:** RoomService

**Service Status:** Active

**Available Operations**
- deleteRoom
- findAllRooms
- updateRoom
- createRoom
- findRoomByld

**HotelService**
Service EPR: http://127.0.0.1:8080/axis2_1.5/services/HotelService

**Service Description:** HotelService

**Service Status:** Active

**Available Operations**
- findAllHotels
- findHotelByld
- updateHotel
- createHotel
- deleteHotel

**CustomerService**

Figure 6.12: The automatically generated web service interface based on modified abstract models - implementation

systems in reality to solve the same type of problem situations, or, in a different contextual setting where the problem situation varies slightly, to assist the process of understanding and improving the new problem situation. In the example that was presented here, the capability to derive implementation artefacts based on information model is captured in a well-formed epitome, which is reused to
6.5 Change and adaptation

automatically generate other implementation artefacts, in a way that has proven to work and be able to satisfy the desired end. Such a mechanism to realise reuse and automation helps to shorten the time required to solve engineering problems, and therefore improves productivity and decision-making efficiency. Moreover, depending on the quality of the capability encapsulated in the epitomes, which is typically verified by the previous empirical experience, this reuse mechanism could potentially improve the quality of improvement to problem situations.

*Organic aggregations* of epitomes therefore form important engineering technical assets, which can be exploited, captured, and reused to assist future engineering problems solving and decision-making. This captured technical competence makes an excellent complement to the *organic aggregations* of higher level intellectual capacities, such as output of World Images from the perception, Rational Images from the Rationalisation, and Capability Images from the Realisation.

As analysed in Section 1.1.1 and 2.2, in such a STE, SOS need to address a high degree of uncertainty and volatilty. This challenge is dealt with based on the notion of *organic aggregation* (see Section 4.1.2). That is, the overall OASEF engineering process is organised and conducted in an *organic* way, which means that OASEF is designed to fit the STE of SOS. Being *organic*, the OASEF process is both "sensible" and "agile" (see Section 4.1.2). In other words, in volatile and uncertain environments, engineering processes, including cognitive efforts towards decision-making, and the actions to respond to the changes, must firstly be justified, and then, secondly, be made at a pace that adequately and sufficiently matches the rate of the changes. Firstly, the "sensible" aspect of OASEF process is realised by a promotion of high level cognitive capacities in rationalisation such as PSE and DEE, and their dependent OASEF inner processes, the perception of the reality. Various modelling mechanism such as CNM, PSM, and DEM are provided to facilitate exploitation of these cognitive capacities. The *organic aggregations* of these high level intellectual achievements such as World Images and Rational Images provide aims and justification for the following actions. Without a clear understanding of these critical aspects, a process could not deal with any uncertainty and volatility with sufficient confidence.

Secondly, the "agile" aspect of OASEF process is realised by the integration of the capacity reuse mechanism with model transformation techniques. As discussed in earlier part of this section, the reuse and automation mechanism in the implementation process encapsulates the technical complexity and improves productivity and decision-making efficiency.

Moreover, the reuse of higher level intellectual achievements such as World Images and Rational Images also contributes to agile engineering decision-making.
As emphasised by Pollock, explicit cognitive operations such as perception and rationalisation are generally time-consuming and highly perishable [Pollock, 2008]. Humankind is often unable to cope with a high degree of dynamics and volatility during these cognitive processes [Pollock, 1995]. Therefore, it is critical to assist human beings to cope with such inability. Reuse is one way to achieve this, since humankind naturally reuse implicit impression and generalised knowledge from previous experience to make quick response to changing environments [Pollock, 2008]. Making such a mechanism of reuse explicit and widely available to people other than those who original possess it would greatly facilitate agile adaptation to changes. The complement of “sensible” perspective also makes the internal and external knowledge reuse more accurate, purposive, or “sensible”.

In summary, OASEF provides mechanisms to enable organic engineering decision-making to respond to uncertainty and volatility in the context of SOS, that is, being both “sensible” and “agile” by reusing engineering knowledge and capacities within their STE.

6.6 Further analysis and discussion

As illustrated by the case studies, OASEF activities and inner-processes can exist in reality, mind, or both. These activities are conducted via various models that are aggregated and transformed in a special way. This mechanism is based on several assumptions: Firstly, reality can be perceived, articulated, understood, and acted upon, and mind can be understood, rationalised, and reasoned about, and these two worlds are interrelated with each other. Secondly, although knowledge and capabilities required for engineering purposes can be aggregated indefinitely, and can be accessed in a controlled fashion, it is best to expose only a definite portion of them to different people within certain scope. It is so because of human cognitive limitations, and because of the different background people possess, such as culture, belief, education, personal interests. Thirdly, there is no universal modelling language that can be used to communicate anything to every recipient, and therefore there is no need to seek it.

Based on the last assumption, the content and representations of the knowledge and capabilities are explored and derived using various modelling techniques in OASEF, with different focus, and for different audiences. For example, human knowledge and understandings of reality are modelled at two levels: the general forms and particular forms, in other words, GA and SA. The case studies demonstrate that the organically aggregated GA using CNM provides a means to model the important correlation relationships among related factors, aspects,
6.6 Further analysis and discussion

objects, and means. These models help to derive other SA and rationalisation models since they provide justifiable knowledge based on well accepted empirical or theoretical experiences. Without these models, observation and analysis of reality would become less conscious and less systematic, or lack predictable direction and consequences. As a consequence, the derived models of the rationalisation and realisation, and the artefacts of the implementation, become less effective, incomplete, or incorrect.

6.6.1 Observed strength

The case studies show that OASEF promotes a focus on higher level human intellectual capacities using various models, including abstraction, rationalisation, and realisation. It provides support for deriving and managing these models in a unified fashion.

As demonstrated in Section 5.3.1, two types of abstraction, namely the GA and SA, provide an important foundation that shapes and guides our reasoning and action processes. The latter, being captured as CNM, provides crucial knowledge to guide successive activities. For example, research shows that high-level concerns such as quality of services, perceived value and trust, and adaptation to feedback are important factors to ensure successful online transactions [Parasuraman et al., 2005, Ku and Fan, 2009]. This general knowledge is captured in Section 5.3.1.1, which in turn guides the successive rationalisation in Section 6.2.3. On the other hand, SA models, in various forms such as general purpose UML and SoaML, provide understanding of systems and their environments. They provide contextual input and reference points for rationalisation and realisation, as illustrated in sections 5.3.1.2, 6.3.1, and 6.4.2.

Section 6.2.3 shows that rationalisation provides a better understanding of the problem situations before any improvement or action is carried out. This is important because otherwise successive actions could be useless or wrong. It shows that even a simple online travel service-oriented system could have complex issues and problems that interconnect with others, which require improvements to be made from different perspectives, not necessarily by building new software or creating new systems. For example, issues related to a low level of user satisfaction, which was identified in Section 6.2.3, cannot be addressed by purely applying implementation technologies such as Web service and service composition techniques. Instead, sufficient methodological support is needed to enable exploration and justification of important problems and causes, and to incorporate high-level concerns with lower-level engineering activities. Section 6.3 and 6.4 shows that OASEF provides a mechanism to derive rational and sensible design
to address identified issues, and a workable implementation that improves the problem situations in the context of SOS.

Moreover, knowledge and capability reuse are facilitated in OASEF. This is achieved by two means: Firstly, as shown in the case studies, perception and conception generate understandable and reusable knowledge in the form of models, at every level including GA, SA, rationalisation, and realisation. For example, general knowledge of customer behaviour in Section 5.3.1.1 can be used in many situations such as B2C e-commerce and Customer Relationship Management (CRM) system, which provides useful guidance to analyse business behaviour, existing problems, desirable improvements and engineering goals. Secondly, by providing epitomisation mechanism as illustrated in Section 6.4, the capabilities to achieve desired ends are generalised and captured, which can be reused to achieve similar goals. For example, the capability to provide a service that allows users to search for flight information is captured in an epitome, which can be used to provide similar service for hotel information enquiries, or weather report information service. Similarly, the realisation process to achieve the higher level goal: Fast booking, can be captured and reused to solve a particular problem: Online travel booking takes too long. This capacity can be reused to solve improve slow online transactions of other kinds, such as slow process of online purchase in a B2C portal.

In OASEF, by using provided modelling tools, various changes can be made to higher-level models in a relatively easy way. This is because: First, these higher-level models abstract away irrelevant concerns and complex implementation and technical detail. Second, most concepts in OASEF model such as CNM and DEM are categorised or packaged at different levels to maintain a reasonably small size of model. As shown in Section 6.5, OASEF provides a model transformation and capabilities encapsulation mechanism to support propagate changes from higher-level models to implementations. For example, various attribute and structural changes made in higher level Flight information models, which is common in software engineering, are automatically adapted during realisation and implementation. OASEF therefore facilitates introduction of changes to the systems at higher level and generating new systems in response to the changes.

6.6.2 Identified issues

The case studies also reveal some issues and limitations. Currently not all parts of the OASEF conceptual model are systematically supported, for example, two of those in the reality world: sensation and control. The lack of the former means that observation of reality can only be made spontaneously by individuals based on their
own experience and knowledge, without quantitative, objective and systematic support. For example, there is no integrated means to sense the performance of a system. Therefore, it is hard to objectively and accurately determine if a problem is justifiable. The lack of control leads to a gap between implemented improvements or systems and their runtime value, that is, to serve particular purposes continuously during their life-cycle. As a consequence, the important feedback loop is broken here because the system effectiveness and value cannot be assessed and perceived. Some important characteristics could not be addressed, such as self-healing and self-adaptation, which is important to realise the objective of OASEF.

Although the higher level models in abstraction, rationalisation, and realisation are captured using general purpose modelling language and specially designed forms, they only capture parts of the reality and mind. Some forms are still prototype and do not capture some important aspects such as evidence to support means-end relationships. In CNM, the correlation relationship carries no differentiation factors in terms of significance that differentiate one relationship from another. For example, although all perceptual dimensions identified in Section 5.3.1.1 affect online service quality, evidence shows that some have more significant impact than others [Taylor and Strutton, 2009], such as perceptive ease of use, regarded as the most influential factor compared with information content, responsiveness and fulfilment.

Moreover, to achieve organic aggregation of knowledge in a harmonious way, and therefore form a justifiable foundation for other OASEF activities, there is one important issue to take into account. That is, how to address the randomness, confliction, contradiction, and inconsistency among various sources of knowledge. A similar issue exists with regards to ambiguity in model elements. That is, how to deal with semantic similarity of different name, or different meaning represented by the same name in different context.

Although some tool support are integrated in IPEOAP, it still lacks some important features, which makes it less efficient in some situations. For example, the process monitoring and debugging is not supported, which makes it hard to identify problems when things go wrong. Furthermore, there are no analysis tools applied to identify syntactic and semantic issues on the abstract models, which makes it inefficient to find and remove problems before models get transformed into other forms.

More importantly, although the proof-of-concept case studies are based on real world analysis and knowledge, the business settings are imaginary and designed to demonstrate and evaluate the objective of OASEF by the author. It is thus less
convincing compared with practice by third parties based on real business projects. Due to their demonstration purposes, the case studies did not cover all aspects of the system such as improve identified service quality issues, including security and trust. More work to evaluate its applicability in practice would be desirable, such as applying process metrics to quantitatively evaluate the effectiveness of this process, especially in comparison with other approaches.
Chapter 7

Conclusion

The future is wider than vision, and has no end.

Donald G. Mitchell

Contents

7.1 Summary .................................................. 183
7.2 Contributions ............................................ 189
7.3 Limitations and future work ...................... 191
As introduced in Section 1.1, in the present digital era, the rapid growth of technologies such as computer hardware and software has radically changed the way human beings interact with the outside world. The technology advances have contributed to a couple of significant social movements at every social level, that is, globalisation and accelerating change. As a result, it is critical to facilitate competition and collaboration among individuals and business communities, at a faster pace, and in a broader scope. In such a context, there is a critical demand for disciplined approaches to engineering flexible and dependable Software-Intensive Systems (SIS) and Service-Oriented Systems (SOS).

However, in today's socio-technical environments, it is challenging to engineer complex SIS and SOS using existing engineering knowledge and practice. In this thesis, we argue that such a challenge is caused by the emerging characteristics of today's large-scale, complex, uncertain, and volatile environments. We observe some fundamental problems of existing practice that cause an inability to cope with such a challenge. Specifically, there is a lack of: 1) suitable engineering processes that are designed for accommodating large scale and volatile socio-technical environments; 2) a focus on human perspectives in the context of engineering; and 3) the reuse of intellectual and engineering capacities and competence.

Our work is motivated by the demand for a disciplined approach to Software-Intensive Systems Engineering (SISE) and services engineering, which addresses the challenges of the socio-technical environments. As described in Section 1.4.2, a multi-phased research methodology was used to continuously and systematically explore practical means to approach our objectives. Research activities are conducted in an iterative fashion, with a focus on a range of inter-related research questions. These questions involve the nature of the problems and challenges, existing solutions to deal with them, a theoretical foundation in support of the new solution, an architectural conceptual framework, practical mechanisms and purposeful design in accordance with identified problems, and evaluation and feedback loop. These research activities are organised in an iterative fashion, which collectively form a novel approach to services engineering. We call it Organic Aggregation Service Engineering Framework (OASEF). It is designed to tackle the identified challenges of services engineering in particular, and SISE in general.

OASEF takes a human-centred and model-driven approach to promote both higher order intellectual capacities and lower level technical competence of human beings. On the one hand, it provides a mechanism to analyse and capture human cognitive or intellectual activities in the world of human mind, such as the derivation and utilisation of general and system-specific knowledge, the exploration and understanding of problem situations and desired ends, and the
cognitive goal-oriented planning and design. On the other hand, OASEF incorporates a model-driven method to realise and capture implementation processes to transform higher order conceptual models into concrete artefacts that eventually form the desired systems. Essentially, OASEF is based on a philosophy based on *organic aggregation*, which takes a synthetic approach to grow and manage organic aggregations of human engineering capacities, including both higher order human intellectual capacities and lower level technical implementation competence.

The remainder of this chapter is organised as follows: Section 7.1 summarises the main concepts, structures, and mechanisms of OASEF; Section 7.2 analyses the main contributions of this research; Section 7.3 presents the problems and limitations that are identified during case studies and evaluation. Some future work is also presented in the end of this chapter for the purpose of making future improvements.

### 7.1 Summary

OASEF is designed to tackle the special characteristics of SOS and their environments. To do so, we examined the nature and challenges of SOS, and identified some fundamental issues to deal with. We conclude that these challenges are fundamentally associated with the complex, uncertain, and volatile socio-technical environments.

In pursuit of a viable solution to address the emerging characteristics and challenges of SOS, we started with reusing existing knowledge and wisdom. We first derived a generic conceptual framework to organise engineering resources such as knowledge, systems, people, and processes, which is generalised from other engineering disciplines. Specifically, it consists of: 1) an underpinning theoretical foundation, 2) a synthesis of a range of guiding principles based on theoretical foundation and practical experience, 3) an engineering process to organise and coordinate engineering activities, 4) integrated methodologies to support and realise the adopted engineering processes in practice, and 5) languages, tools, and techniques to enable and facilitate the engineering processes and methodologies, 6) individuals, business organisations, and other human communities that interact with the systems under consideration, and 7) an evaluation and feedback process in relation to every other engineering elements.

Within such a framework, we particularly emphasise the importance of a theoretical foundation, which provides the basis of any other engineering resources and events and influences human responses to engineering problems. We argue that, for effective services engineering and SISE, it is essential to build the foundation
through a multi-disciplinary approach, taking into account the socio-technical environments of SOS. Therefore, a sensible theoretical foundation is laid for OASEF for the purpose of addressing SOS challenges. This foundation coherently synthesises important knowledge and insights from various pertinent disciplines, such as engineering, philosophy, and management. Specifically, OASEF foundation consists of organic aggregations of multi-disciplinary knowledge such as systems theory, complexity theory, modern philosophy of mind and action, and process philosophy. We do not intend to present all investigated areas and the obtained knowledge during the exploration. Instead, only some highly relevant portion are described in this thesis in support of the corresponding design, including the philosophy of mind and action, systems theory, theory of and evolution, the philosophy of pragmatism, and process philosophy. The knowledge within these areas is either explicitly referenced, or is regarded as important conceptual requirements for readers to understand the origin, scope, purpose, and meaning of OASEF activities. Moreover, the OASEF theoretical foundation also contains some well-defined concepts and notions, such as services, services engineering, and models. concepts and mechanisms.

Moreover, OASEF coherently incorporates a range of practical guiding principles, which are either deduced from philosophy and theories in the theoretical foundation, or generalised from engineering practice. The latter forms of principles include well-known software engineering principles such as higher level of abstraction, information hiding—, and separation of concerns, which are not explained in this thesis. Other guiding principles that are specifically developed for the purpose of this research are instead clarified explicitly, including environment-driven view, multi-disciplinary view, use “soft” methods instead of “hard” ones to deal with ill-understood phenomenon, maximise the reuse of engineering capacities, and everything is driven by models.

The foundation and guiding principles are chosen to help addressing the SIS challenges. For example, applying higher level of abstraction and separation of concerns helps to tackle the problems of complexity and volatility, because engineering practices prove that they help to reduce the conceptual complexity of problems, and lead to simple and flexible architecture that improves system scalability and flexibility. These principle are explicitly applied in OASEF. Firstly, OASEF was designed to raise the level of modelling abstraction from the human perspectives, that is, understanding and modelling cognitive capacities in an engineering context. Secondly, it supports the separation of concerns by coherently organising engineering resources in two distinctive, but inter-related worlds: reality and mind. Different concerns and aspects are encapsulated in various OASEF inner processes and activities according to their nature and characteristics.
7.1 Summary

As another example, *everything is driven by models* principle helps to organise and manage every OASEF activity and resource in a unified way, which facilitates the interconnectivity among various OASEF models since models can be easily referred to by each other. More importantly, it makes it possible to automatically transform models among different forms, for example, from high level models to low level implementation. As shown in the case studies, this design helps to improve system agility.

Based on the theoretical foundation and guiding principles, an important notion called *organic aggregation* is developed, for the purpose of attacking the grand challenges of modern SIS. It emphasises two aspects: 1) using a synthetic approach, as opposed to reductive ones, to observe and interact with the world outside ourselves; and 2) taking an *organic* approach to understand, derive, and evolve systems that fit into their socio-technical environments. That is, in addition to having a focus on agility, it is also critical to focus on both the internal coherence and external environmental fitness, such as human rationality, psychological coherence, social and cultural harmony. In this thesis, we argue that the notion of *organic aggregation* should be advocated and applied in engineering to attack socio-technical challenges.

Based on the OASEF theoretical foundation, guiding principles, and the notion of *organic aggregation*, we designed a novel engineering process conceptual model, which is called Organic Aggregation Process (OAP). It consists of *organic aggregations* of inter-connected activities that are organised in a coherent and flexible hierarchy. The overall structure of OAP forms a coherent linkage between higher order intellectual efforts in human *mind* and practical activities in the *reality*. The OAP process is considered *organic* because it involves coherent engineering resources and capacities that inter-connect with each other, which work as a whole during purposeful human activities. Moreover, its inner architecture and external nature enables SIS to be engineered in an *organic* and *aggregative* fashion. Therefore, both the inner structure of OAP and its inner mechanisms to build and manage systems applies the notion of *organic aggregations*. The ultimate objective of OAP is to form and grow the *organic aggregations* of engineering capacities and resources to derive and manage fit systems, including both high order human intellectual capacities, and lower level technical competence.

At the fundamental level, OAP consists of both world of *reality* and *mind*, with human experience connecting them altogether. Engineering resources such as higher order human intellectual resources and lower order technical resources exist within these two types of distinguishing worlds. Such a distinction helps to understand and treat engineering resources differently according to their nature. Generally, OAP involves: 1) human understanding of outside world, 2) cognitive
recognition of problem situations, 3) derivation of purposes and desired ends, 4) intellectual design of solutions and plans, and 5) practical actions that eventually change the states of the reality. Here, elements 2 to 4 mainly involve the world of human mind, whereas elements 1 and 5 involve both worlds and their interactions during human engineering experience.

Specifically, the organic aggregations of OAP activities, involving various engineering resources and capacities, are organised in a flexible hierarchy at various levels. At the first level, engineering consists of iterative processes of perception, conception, and action. These inner processes form a complete conceptual loop between the world of mind and reality. The important intellectual achievements from perception and conception are critical for organic engineering, since they justify that the engineering activities are necessary and sensible in the first place, and are of pragmatic value once putting into operation. In other words, OAP ensures a complete loop that justifies the necessity and meaning of the whole process, and realises the value of action and their results in the world of reality.

At the second level, inter-connecting OAP activities are at a finer level of granularity. Specifically, abstraction and sensation organically form perception. Conception consists of organic aggregations of rationalisation and realisation. Action involves implementation and control. The nature, meaning, and inter-relationships of various important concepts are thoroughly discussed and clearly defined in this thesis. Because of the focus of human perspectives in this research, we especially elaborated those concepts in association with the higher order human intellectual capacities, such as abstraction and rationalisation. We developed various inner processes to derive and capture these important aspects of human mind. Specifically, two types of abstraction are developed: System Abstraction (SA) and General Abstraction (GA). The former is used to obtain and capture the understanding of relevant parts of systems and the environment in which they reside. Whereas the latter is used to derive and represent the general experience, knowledge, understanding, insights and wisdom, either in a general sense, or discipline-dependent fashion. Another critical part of OAP, the rationalisation, instead, uses two types of inner-processes to analyse and capture the desired intentions. These two inner-processes are called Problem Situation Exploration (PSE) and Desired End Exploration (DEE). The former cognitively explores the problem situations, reflecting human beings perceive or feel unsatisfied situations in the reality. The latter explores specific human desired ends that capture what result is desired according to the level of certainty and difficulty involved in achieving it during the purposeful human activities.

The general purpose OAP conceptual model is fully integrated within OASEF to organise engineering resources and to conduct engineering activities in the
context of services engineering, in an organic aggregation fashion. For example, perception, rationalisation, and realisation are used to derive and capture higher level human intellectual efforts. This is realised by applying the guiding principle: everything is driven by models (see Section 3.3.5). A model-driven method is integrated with OASEF at all levels. Achievements of OASEF activities collectively form specific resource images, in various forms of models. Resource images represent repositories of knowledge obtained during the OASEF activities or inner-processes. For example, the rationalisation takes a World Image from perception as input, which is the projection of relevant reality in the world of human mind. In turn, rationalisation produces a Rational Image that consists of both rationalised Problem Situation Model (PSM) through PSE activity, and desired human intentions captured in Desired Ends Model (DEM) as outputs from DEE activity. Moreover, realisation takes the Rational Image and produces a Capability Image that represents necessary capabilities to be realised.

Models and resource images are represented using appropriate modelling languages, either general purpose ones or Domain-Specific Languages (DSL). Due to our focus on higher order human intellectual capacities, we particularly emphasise abstraction and rationalisation in the early stage of the process life cycle. We design specific modelling languages and graphic notations for the Correlation Network Model (CNM), PSM, and DEM, in order to express and capture these important aspects of the higher order human cognitive achievements, during GA, PSE, and DEE activities, respectively. Every OASEF activity produces either domain-specific or general-purpose models that all conform to a unified meta-meta-model. Therefore, activities in OASEF can be conducted and inter-connected in a coherent fashion. That is all OASEF models are manipulated and utilised in the same way. Moreover, OASEF models are used not only to understand different aspects of the mind and reality or to communicate with peer stakeholders, but also to derive systems in a systematic, often automatic, way. Various Model-Driven engineering (MDE) mechanisms to transform models and generate code are utilised to systematically derive systems from various high-level models. The model transformation mechanism is especially useful to capture lower level engineering technical capacities. Altogether, these modelling mechanisms provide effective means to capture knowledge and outputs of various OASEF activities, and to utilise them to derive, or to transform them into, usable and executable SOS.

Moreover, OASEF provides a mechanism called epitome to reuse engineering capacities while conducting OASEF activities. An epitome is a typical and justified means, or capability to achieve specific engineering purposes. It is generalised from proven examples that tightly link two OASEF activities together. Applying an
existing epitome directly produces previously proven outcomes without having to go through the particular activities. Therefore, this mechanism provides a flexible way to conduct OASEF activities, and enables capturing and reuse of proven engineering capacities. Organic aggregations of these epitomes form a valuable repository of engineering capacities that can be identified, accessed, and reused by other stakeholders, for understanding or improving similar problem situations.

An integrated engineering environment called Integrated Process Environment for Organic Aggregation Process (IPEOAP) is also incorporated within OAP and OASEF. It consists of various supporting tools that facilitate various OASEF activities, such as manipulation of various OASEF models and generation of desired SOS. It is developed using the Java programming language on top of Eclipse IDE and other modules supported by Eclipse Modeling Project (EMP). A range of graphical model editors are provided to view, create and manipulate models of various OAP activities, such as CNM, PSM, and DEM. Moreover, IPEOAP also incorporates our epitomisation processes and supports automatic transformation of models and systematic generation of implementation artefacts. For example, PSMs are automatically transformed into documents that describe the problem situations using natural languages that applying logical rhetorical discourse arrangement. Similarly, UML activity diagrams in realisation can be translated into lower-level run-time artefacts, such as BPEL artefacts or Java code.

In order to evaluate the effectiveness of OASEF, two proof-of-concept, as opposed to proof-of-performance, case studies were conducted in real world settings, that is, case studies on Online Travel Booking (OTB) and First Home Saver Accounts (FHSA). The results show that the OASEF approach to services engineering help to alleviate the impediment of complexity, uncertainty, and volatility. More specifically, the following positive results are observed. Firstly, by using higher order models such as CNM, PSM, and DEM, OASEF enables and promotes coherent higher order intellectual activities in the world of mind, through which effective SOS are systematically derived. Models in abstraction, rationalisation, and realisation are captured as important engineering resources that can be located and aggregated together. They therefore form important human intellectual assets that provide creative and valuable essence to achieve the “right” and optimised systems. Secondly, activities involving implementation technologies in action, are also captured as reusable engineering capacities, and are used to automatically realise identified desired ends in a relatively easy way. Thirdly, using a model-driven method, OASEF creates an effective linkage between higher order intellectual efforts and lower level implementation processes, since both types of resources are organised in a unified, identifiable, reusable fashion. The organic aggregations of resource images are used collectively to systemically and
7.2 Contributions

automatically drive the engineering processes to produce SOS. An improvement in productivity is observed although not objectively measured.

To conclude, in OASEF, we promote a higher level of modelling abstraction and the separation of various engineering concerns in the world of mind and reality. This is achieved in conjunction with a range of other OASEF guiding principles, such as *everything is driven by models*, and *maximise the reuse of engineering capacities*. Taking a *multi-disciplinary view*, we particularly focus on higher order human intellectual efforts as important engineering resources to tackle complex and volatile systems. As discussed in chapter 3, application of these principles holds the key to attack the challenges of complexity, uncertainty, and volatility. Therefore, OASEF provides concrete various mechanisms to enable and facilitate formation and exploitation of both human higher-order intellectual capacities and lower-order technical implementation processes, which was implemented and experimented during case studies. The proof-of-concept implementation shows that OASEF creates a linkage between its two world of resources, the *mind* and *reality*, and therefore forms valuable *organic aggregations* of engineering knowledge and capacities.

7.2 Contributions

Due to the critical demand of SIS and SOS in today's socio-technical environments, there is a great opportunity for this research to be of value. Because of the rigid and systematic research methodology we took, our work started from the exploration and formation of a sound theoretical foundation. This approach results in a novel and sensible way to deal with SISE in general, and services engineering in particular. The observation, exploration, and reflection of various research efforts contribute to the emergence of a new engineering discipline of services engineering, and contribute to the evolution of SISE in the modern socio-technical settings of the 21st century [Wang and Taylor, 2008, Wang, 2010a,b].

Specifically, this work has the following contributions.

- It develops a multi-disciplinary theoretical foundation from a broad socio-technical perspective. Fundamental notions, concepts, and knowledge that are pertinent to SIS, SOS, and services engineering are identified, synthesised, developed, articulated, and defined. This theoretical foundation therefore provides a coherent base for development of other approaches to tackle the challenges of complexity, uncertainty, and volatility of modern SIS.

- It establishes understanding and insights of the problems, challenges, and
special characteristics of SOS and modern SIS, which help to clarify critical research questions and objectives of the new discipline of services engineering. Knowledge about the nature and characteristics of SOS helps to set the stage for general software engineering research communities, in order to study and understand the modern socio-technical environments of software.

- It proposes a sound and coherent engineering framework for coordinating and organising engineering knowledge, capacities, processes, and other engineering resources. Such a general engineering framework can be used as a starting point or reference model to systematically develop or evolve other approaches to services engineering.

- It develops a profound notion called organic aggregation, which has wide applicability in the context of engineering. Such a notion helps to derive fit systems in a synthetic fashion. Being organic means being coherent, ordered, logical, harmonious, or sensible, taking into account all characteristics that a healthy organism possesses in order to fit into its environments. Therefore, an organic system approach takes more environmental aspects into consideration, such as rationality and fitness, compared with traditional structured and agile approaches that have limited focus on agility and end user involvement.

- It proposes a novel general purpose software engineering process model called OAP, which is developed based on the solid theoretical foundation for the purpose of addressing identified challenges of modern SIS. It consists of a coherent and flexible conceptual structure of carefully-categorised, well-organised, and inter-connected activities at various levels.

- It promotes a focus on, and provides supporting mechanisms for, high-level human intellectual activities at a higher level of abstraction in the world of human mind. Specifically, it facilitates: 1) deriving and capturing general human knowledge about specific and general systems using GA modelling, 2) exploring and understanding relevant problem situations during PSE, and 3) identifying and understanding higher order human engineering intentions, desired ends during DEE. Specific modelling mechanisms and modelling languages are designed for these important activities, such as CNM, PSM, and DEM domain-specific modelling languages. The OASEF approach therefore provides concrete means to raise the level of modelling abstraction and separates human intellectual concerns in the world of mind and reality.

- It provides a novel mechanism called epitome and epitomisation to capture and reuse engineering capacities based on the general concept of organic aggregation. During the prototype implementation, this mechanism is
incorporated with a MDE modelling techniques, which, collectively, link the aggregated higher-order intellectual capacities with other lower-order implementation processes. The latter can be systematically and sensibly derived from the former.

- It provides a proof-of-concept integrated environment called IPEOAP which provides unified modelling features for our specially designed DSL and the general purpose modelling language such as UML. Various models, especially those at higher levels of abstraction in mind, such as CNM, PSM, and DEM, are graphically supported in IPEOAP to capture the perception and rationalisation of abstraction models, problem situations and desired ends. Model manipulation and transformation tools are integrated within the integrated environment to support the OASEF activities to systematically generate workable SOS.

7.3 Limitations and future work

Like other human cognitive activities, developing software can be extremely difficult. The great difficulty fundamentally comes from the ill-understood nature of human minds. Our effort to incorporate this aspect of human perspective into SISE is therefore exploratory by nature. The multi-disciplinary approach we take involves good understanding and creative synthesis of various knowledge pertinent to engineering and human beings. Due to a lack of prior knowledge in social sciences and steep learning curves, the journey of this exploration has not been smooth. For example, studying and understanding the meaning of various schools of philosophical thoughts from enormous materials was painstaking. Language sometimes impedes precise understanding of the meaning of various arguments. As a result, the current understanding and synthesis of various knowledge is limited and insufficient. What has been investigated and incorporated within OASEF is just a tiny portion of the enormous body of pertinent knowledge, and may not hold true under all circumstances. Some understanding and analysis might be very superficial or incoherent.

The formation and synthesis of a multi-disciplinary foundation to form a viable solution has been, and will always be, a continuous process. The only way we can battle such a cognitive limitation is through continuous learning, unlearning, and relearning. We will look into the state of the art research that involves human perspectives in software engineering and systems engineering, such as
cognitive engineering, human-machine systems, Socio-Technical Systems (STS), and System of Systems (SoS). To expand our interest in understanding the human perspectives in an engineering context, knowledge and wisdom in various social science disciplines such as psychology and organisational management will be investigated, and consciously applied in future software engineering research.

Specifically, current research has the following major limitations. Firstly, the profound notion of organic aggregation has not been fully developed and justified. For example, there is no clear answers to a number of important questions, such as: how do we measure the degree and value of being organic? What are the guidelines to determine or prioritise different organic factors in different environments? When is rationality preferred, as opposed to agility, and to what extent they need to be satisfied? More importantly, how to be organic in any environment, especially during the runtime? Answers to these questions would help to develop a radical paradigm shift towards organic methodology or organic engineering. Which could be taken up by the currently active agile community. Moreover, although the aggregative approach aligns well with the concept of service-orientation, it remains unclear if there is any differentiation between them. Moreover, are aggregations of capacities differentiated with aggregations of knowledge or information? The aggregations of experience, knowledge, capacities, services, processes, or any other types of systems needs methodology, tool, and infrastructure support.

Furthermore, the general notion of organic aggregation needs to have a more concrete grounding in specific contexts, because being organic is context-sensitive. Guidelines and instructions are needed to form concrete formalisms and comprehensive infrastructure to realise the pragmatic value of this notion. Although we implemented a proof-of-concept tool environment to support aggregation of certain type of engineering capacities, it is experimental and provides no infrastructure to store and search for various aggregations. Therefore, our vision of a paradigm shift from agile and service-orientation towards organic aggregation needs both theoretical and practical development, justification, and support, without which, it is just another "great" idea that produces no or limited real value.

Secondly, although the proof-of-concept implementations and two case studies have been conducted to evaluate the conceptual design, this new approach has not been comprehensively evaluated in real world projects. The scope and methods of the evaluation remain unclear at this stage. This is considered a critical shortcoming since this research aims to produce pragmatic value in practice. Without sound and convincing evaluation against certain benchmarks, logical deduction and subjective observation and experience would remain suspicious and uninteresting for serious practitioners. Moreover, the proof-of-concept implementation lacks comprehensive tool support and incomplete features and functionalities.
7.3 Limitations and future work

The evolution of some third party tools also breaks the integrated environments from time to time, which requires significant time to accommodate such changes. Furthermore, many important areas and ideas have not been fully investigated and realised, such as the system monitoring, the concept of control, automated design and operation of sensation, feedback control, maintaining communicational organic aggregation among stakeholders, and social aspects of human perspective and collaborated engineering environment, to name a few.

Thirdly, OASEF only touches a small part of the human perspectives, that is, the understanding and exploitation of cognitive activities within human mind during engineering processes. We have not considered various influential factors that cause differentiation between various individuals. Furthermore, we have not taken into consideration the communicate and collaboration patterns within or across groups. The means to enable and facilitate communication of different thoughts, and the way they influence each other remains unclear. Moreover, we have not obtained understanding of how psychological, cultural, and organisational human perspectives could affect engineering. For example, do psychological and social factors affect the development and improvements of intellectual capacities and technical competence? If so, how? Understanding in these areas is important to achieve organic aggregations of fit systems.

We intended not to be over-ambitious and to tackle every aspect of services engineering in particular, and software engineering in general. We find it difficult to keep the right balance between the width and depth of the research along the journey. However, research is an intellectual human activity. It involves cognitive decision-making that also has various human perspectives. The final path we choose is influenced by many factors such as personal interests and research environments. Nevertheless, this research turns out to be a very interesting journey. It is also the beginning of many other ones. Developing a strong vision is one thing, working towards it and seeing it growing are others. We have not seen the end yet as our vision is far from being realised. In fact, the more we work on it, the more is needed to be done. In order to make more significant contributions to a nascent discipline of services engineering, and a “still-young” discipline of software engineering, much more effort is necessary in the future.
Bibliography


Ceponiene, L and Nemuraite, L. Transformation from requirements to design for service oriented information systems. *ADBIS Research Communications*, 2005.


Humphrey, W. Managing the software process. Addison-Wesley, 1990.


BIBLIOGRAPHY


Acronyms

API Application Programming Interface
BPEL Business Process Execution Language
CC Cloud Computing
CNM Correlation Network Model
CRM Customer Relationship Management
DD Data Dictionary
DEEM Desired Ends Model
DFD Data Flow Diagram
ERD Entity Relationship Diagram
ESB Enterprise Service Bus
GA General Abstraction
IPEDAP Integrated Process Environment for Organic Aggregation Process
MDR Model-Driven Engineering
OAP Organic Aggregation Process
OASEP Organic Aggregation Service Engineering Framework
OOAD Object-Oriented Analysis and Design
PSM Problem Situation Model
SA Symbolic Abstraction
SOA Service-Oriented Architecture
SOAOSM Service-oriented architecture Modelling Language
SOEC Service-Oriented Computing
Acronyms

API  Application Programming Interface.
BPEL  Business Process Execution Language.
CC  Cloud Computing.
CNM  Correlation Network Model.
CRM  Customer Relationship Management.
DD  Data Dictionary.
DEM  Desired Ends Model.
DFD  Data Flow Diagram.
ERD  Entity Relationship Diagram.
ESB  Enterprise Service Bus.
GA  General Abstraction.
IPEOAP  Integrated Process Environment for Organic Aggregation Process.
MDE  Model-Driven engineering.
OAP  Organic Aggregation Process.
OASEF  Organic Aggregation Service Engineering Framework.
OOAD  Object-Oriented Analysis and Design.
PSM  Problem Situation Model.
SA  System Abstraction.
SOA  Service-Oriented Architecture.
soaML  Service oriented architecture Modelling Language.
SOC  Service-Oriented Computing.
**Acronyms**

**SOS** Service-Oriented System.

**UI** User Interface.

**UML** Unified Modelling Language.

**WSDL** Web Service Description Language.