OASEF: A Synthetic Approach to Service Engineering

Yuanzhi Wang

Department of Computer Science
The Australian National University, Australia
derek.wang@anu.edu.au

Abstract. Engineering complex service-oriented systems presents grand challenges due to their great complexity, volatility, and uncertainty in their rapidly evolving technology and social contexts. It demands an effective engineering approach in order to satisfy the needs of service economics. This paper proposes an approach called Organic Aggregation Service Engineering Framework (OASEF). Experience from proof-of-concepts case studies shows that it provides a practical means to develop service-oriented systems. It enables and promotes a focus on higher-level intellectual engineering efforts, and provides a mechanism to capture and reuse engineering capacities in a model-driven environment.

Keywords: Service Engineering, Model-driven Engineering, Service-oriented Computing

1 Introduction

Deriving and managing complex Service-Oriented Systems (SOS) presents great challenges due to their nature and characteristics in a dynamically complex environments [1]. Firstly, SOS often involve great complexity in terms of variety, scope, and inter-relationship. On the one hand, technologies such as Service-Oriented Computing (SOC) and Cloud Computing (CC) greatly facilitate development and integration of applications and systems. 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. As a result, the size, variety, and scope of interrelated systems scales up rapidly.

Secondly, SOS often present a high degree of uncertainty. Autonomous services are often provided and consumed by different agents for different types of purposes, without necessarily knowing each other in advance [2]. Moreover, defining a service contract at any time cannot completely incorporate unknown usage with necessary variations. Therefore, it presents great challenges to satisfy not only the explicit requirements of current target applications, but also the needs of envisioned future applications or unknown potential users. Lastly, in highly volatile and heterogeneous environments, the velocity and variety of appropriate response have to match those of the environmental changes. An effective service engineering hence has to support and facilitate dynamic evolution by changing and reshaping their internal structures and behaviour, in a more frequent and predictable fashion, compared with traditional systems.

These special characteristics and challenges can hardly be resolved by merely adopting traditional software processes and engineering methodologies, or solely relying on more advanced implementation technologies [3]. In order to enable incorporation of wide range of business and social systems within rapidly growing global service economics, there is an important and urgent demand for a mature discipline of service engineering [4]. In our view, such a discipline, to be effective, must provide sufficient support for addressing the grand challenges of complexity, uncertainty, and volatility while engineering SOS in their open environments. We believe one way to approximate this ideal is to enable and promote derivation and reuse of important higher-order intellectual efforts and lower level technological capacities. This is because evidences have shown that great complexity involved in engineering problems is better dealt with by intellectual cognitive experience and capacities, which lead to sensible perception of reality, logical reasoning of problematic situations, and systematic derivation of conceptual plans [5]. Therefore, effective and efficient accumulation and use of these intellectual resources is the key crucial factor to manage the ever-presenting complexity, uncertainty, and volatility in a timely fashion while changes present themselves. In the meantime, in order to take full advantage of advanced technologies and implementation infrastructures, and at the same time, deal with their increasing heterogeneity and complexity, a discipline of service 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 service 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, both 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, our objective is to achieve an effective service engineering approach that: i) promotes a focus on high-level intellectual activities within the world of human mind, such as exploring and understanding problematic situations, and identifying and capturing higher order engineering purposes and intentions; ii) 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 iii) 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.

This paper presents a new approach to service engineering based on such a vision, which is called Organic Aggregation Service Engineering Framework

(OASEF). It is based on a philosophy that takes a synthetic approach to growing and managing processes and systems via sensible and responsive aggregations of resources and capacities. This work mainly has the following contribution: firstly, it emphasises exploration and exploitation of higher-order engineering efforts, and links them with lower-level technological resources within an overall process model; secondly, it promotes a concept of *organic* aggregation in engineering that captures engineering capacities as reusable resources, and, more importantly, uses them in a sensible, agile, and controlled fashion.

The remainder of the paper is organised as follows: section 2 presents the proposed OASEF framework in details, including a general engineering framework it conforms to, its conceptual model, and modelling methodology; section 3 briefly illustrates its important concepts and methods using results from two proof-of-concept case studies; section 4 analyses its features and limitations based on observation and experience during the case studies, and compares it with some related work; section 5 concludes this paper with a summary.

2 The OASEF approach to service engineering

It is important for a discipline of service engineering to have a well-founded engineering framework that coherently organises engineering activities and resources according to the nature and characteristics of services and systems under consideration. The aim of this work is to provide and assess such a framework.

2.1 OASEF conceptual model

The design of OASEF started from generalisation of other engineering disciplines. Figure 1 depicts a general conceptual framework to which OASEF conforms. It includes a theoretical foundation, or a coherent conceptual system, which consists of of inter-related theories, knowledge, and wisdom that are shaped or influenced by individual and social experience, beliefs, philosophies, and culture [6]. Such a foundation, either implicitly or explicitly, guides and shapes, positive experience and practices that are justified in the course of continuous engineering activities, which gradually form a range of concrete and specific guiding principles. These principles, articulated, well explained, and understood, provide valuable guidance for engineering activities in practice.

Moreover, based on the abstract foundation and principles, more concrete engineering activities are arranged and conducted within a specific engineering process, which, in turn, is realised or supported by a range of practical engineering methodologies. The latter provides specific and concrete means to realise the aiming higher level engineering purposes, and to solve practical human problems in a controllable, repeatable, and efficient fashion [6]. Specific languages, tools, and implementation techniques practically enable or facilitate the engineering processes and methodologies, and eventually produce or manage the target systems in reality. Altogether, these coherent engineering elements,

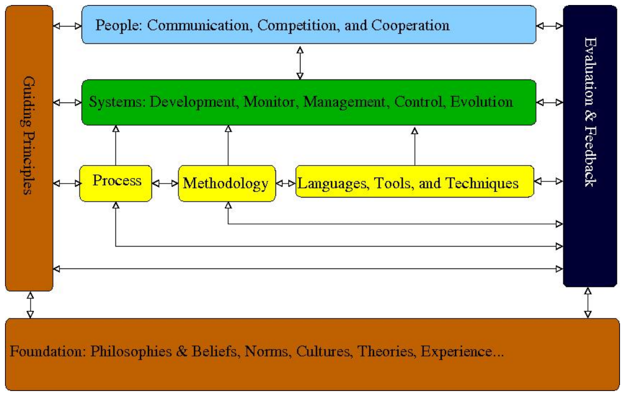
People: Communication, Competition, and Cooperation
Systems: Development, Monitor, Management, Control, Evolution
Process
Methodology
Languages, Tools, and Techniques
Guiding Principles
Evaluation & Feedback
Foundation: Philosophies & Beliefs, Norms, Cultures, Theories, Experience...

Fig. 1: A meta-model for service engineering

involving a variety of stakeholders, contribute to engineering of resources, infrastructure, events, processes, and systems that aim to satisfy identified engineering needs and human purposes.

The overall structure of OASEF is depicted in figure 2. It conforms to the general framework, and is based on a theoretical foundation rooted in some multi-disciplinary knowledge, such as general system theory and modern philosophy of mind. It also incorporates some practical principles by which its process and methodology are guided or reinforced. These guiding principles, such as “environment-driven view of service” [7], “pervasive change and evolution”, “support for systems adaptation and agility”, and “facilitating knowledge and capability reuse”, will not be presented here due to space limitation. Although their meanings are well known, widely accepted, and often taken for granted, they provide important empirically verified guidance for engineering activities.

As shown in the figure, OASEF incorporates a flexible process model called Organic Aggregation Process (OAP) in the context of service engineering. Figure 3 illustrates its inner structure and inter-relationships at a detailed level. The lines with text besides them represent specific process activities, whereas the joint point between two lines represents direct correspondence between one activity to another. Moreover, activities are arranged hierarchically in this structure. An activity represented by a single line, starting from point A to point B in the clockwise direction, may comprise a sequence of subordinate activities that are represented by a series of adjacent lines, which also starts from point A and ends with point B clockwise. Superordinate activities are represented by thicker lines and larger-size fonts on the hierarchy. For example, *perception* comprises subordinate activities of *sensation* and *abstraction*. The latter, in turn, comprises *system abstraction* and *general abstraction*.

The OAP concepts and process are integrated in OASEF at various level, as illustrated in the middle part of figure 2. 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, *Concep-*

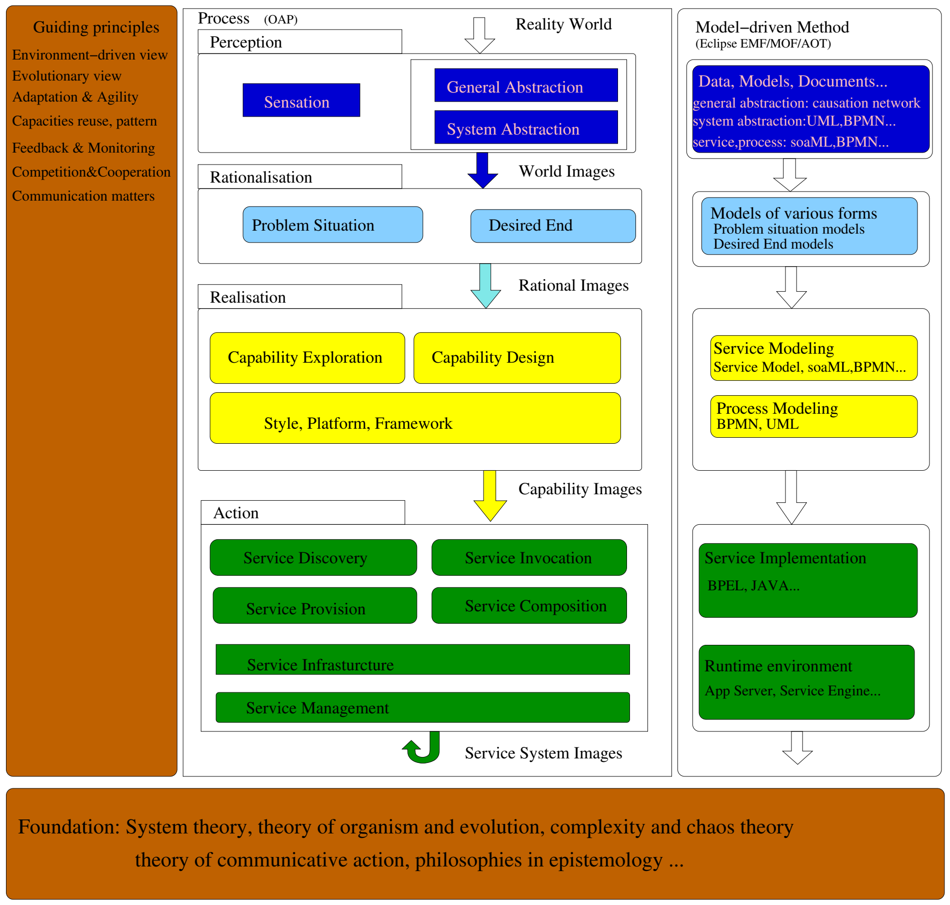

Fig. 2: OASEF: Organic Aggregation Service Engineering Framework

tion activities involve *Rationalisation*, the reasoning and sense-making processes that produce understanding of perceived situations, inferred problems, and desired ends, which are realised by subordinate activities such as *Problem Situation* and *Desired End*. *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 outcome of *Rationalisation*. Furthermore *Action*, conducted in the world of *Reality*, deal with concrete service systems by finding and invoking existing services, providing new services, or composing composite services. *Service Infrastructure* provides information of implementation environment such as Enterprise Service Bus (ESB), whereas *Service Management* provides control and monitoring of services in run-time environments.

Therefore, OASEF derives and manages services using interconnected OAP activities. Achievement of these activities collectively forms specific images representing unified repository of knowledge, which can be referenced, or used as

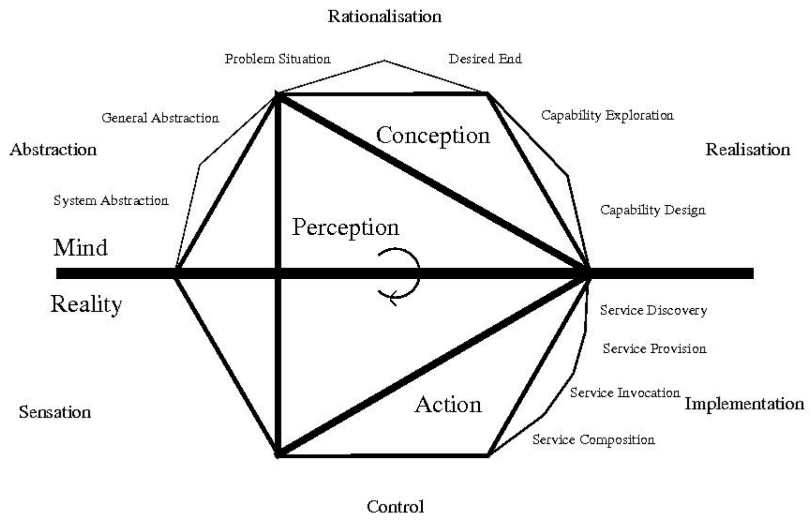

Fig. 3: The structure of OAP process model in service engineering

input, by successive activities. For example, the *Rationalisation* takes *World Images* from *Perception* as input. It produces *Rational Images* that represent rationalised *Desired Ends* models to deal with identified *Problem Situation. Realisation* takes the *Rational Images* and produces *Capability Images* that represent required realisable capabilities. The latter is taken by *Action* as input and eventually produces SOS that alter the state of the *Reality* with an aim of improving rationalised problematic situations.

2.2 OASEF modelling and tools support

The conceptual model of OASEF is brought into reality using specific modelling techniques and model-driven methodologies. Various abstract models, either general purpose modelling language such as UML, or specially designed modelling languages, are used as essential artefacts to facilitate OASEF activities, and to link these activities altogether throughout the engineering life cycle.

OASEF emphasises activities of *Abstraction* and *Rationalisation* in the early stage of the life cycle. The objective of *Rationalisation* modelling is to analysis and justify the rationale and needs of engineering processes toward sensible decisions of actions. Two types of modelling formalism, namely Problem Situation Models (PSM) and Desired End Models (DEM), are designed to capture the crucial higher order cognitive achievement. For example, output of *Problem Situation* activity is captured in terms of complex inter-relationships between various problems, facts and constraints, and high-level purposes. Together, PSM and DEM provide an important means to explore and represent problematic situations and desired ends that justify successive engineering activities and their produced engineering results.

As an example, the graphical notations of *DEM* are depicted in figure 4. Coloured rectangles with names inside represent various desired ends elements. They capture the engineering intentions in terms of their concreteness and scope. For example a *Goal* is more specific in scope, less abstract to define and communicate, and easier to measure, compared with *Objective* and *Ideal*. Specifically, light yellow rectangles represent *Ideal*, whereas light green and dark green ones represent *Objective* and *Goal* respectively. Moreover, white rectangles are used to represent higher order *Capabilities* that provide contextual meaning and desired value in support of service identification and realisation. Relationships among various DEM elements are represented by connecting arrowed lines. For example, an arrowed line can link a *Ideal* with its subordinate *Objective*, a *Objective* with composing *Goal*, or a *Goal* with its subordinate one.

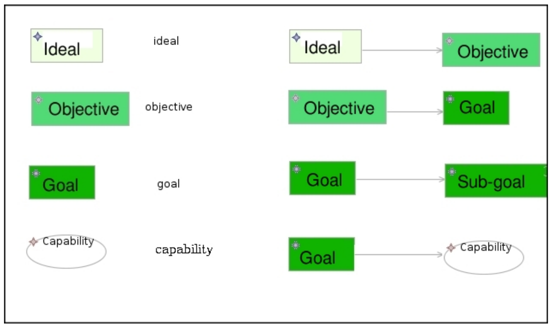

Fig. 4: Graphic notations of DEM

Every OASEF activities produce either domain-specific or general-purpose models that all conform to a unified meta-meta-models. Therefore, activities in OASEF can be conducted and inter-connected in a coherent fashion by manipulating and utilising every model in the same way. For example, UML models based on Eclipse Modelling Framework (EMF) are used to capture information and knowledge in *Sensation* and *Abstraction*, in the same way PSM and DEM are manipulated. UML activity diagram, soaML, State Machine, and BPMN are used in *Realisation* similarly. Some models transformation techniques on Eclipse modelling platform are also integrated in OASEF to transform various OASEF models into different forms and generate desired system in an automatic or semi-automatic fashion.

Moreover, OASEF provides a mechanism called *Epitome* to reuse engineering capacities to conduct OAP activities. It is a typical and justified means, or capability to achieve some engineering purposes. It is generalised from proven examples that tightly link two OAP activities together, such as a specific *Realisation* that is able to achieve predictable and optimised results in reality to improve identified problematic situations. Applying an existing *Epitome* directly produce a previously proven outcome without having to go through the particular activities. For instance, it generate specific *Realisation* models that are able to improve the typical problems.

A number of supporting tools are integrated within an integrated supporting tools environment, called IPEOAP. It is developed using Java programming language on top of Eclipse IDE, EMF and modelling platform. A range of graphical model editors are developed to view, create and modify models for various OAP activities, such as PSM and DEM.

3 Case studies

Two controlled case studies were conducted as proof-of-concept, as oppose to proof-of-performance, which aim to assess whether, in general, its objectives are meet in real world settings. The first case study is in the context of online travel booking business, a typical scenario used in service community. While the second one is based on Australian First Home Saver Accounts (FHSA) scheme that was introduced by the federal government in 2008 to help residents to purchase their first homes. This section briefly presents some outcomes of these two case studies to illustrate the main feature, concepts and application of OASEF.

A PSM in the context of online travel booking exemplifies the high level analysis of *Problem Situation* in *Rationalisation*. It contains many higher level *Purposes* for online booking, such as “Ease of use”, “Quick Responsiveness”, “Reliable booking”, ”Economic price”, and “Secure and trust”. These PSM elements are generated systematically in accordance with general knowledge captured in *general abstraction* models within *world images*. It also contains a range of higher level *Problems* such as “The booking process takes too long”, which violates the “Quick Responsiveness” *Purpose*. This *Problem* is also caused by other *Problems* such as “Too many service providers involved”, “Some providers are less efficient than others”, “Insufficient network bandwidth”, ”Diversity in interface and interaction mechanisms”, and ”The ordering process is too complex” that is caused by “sequential correspondence”. Exploration of problems and purposes reveal their nature and relationships and help to identify, understand, and capture important problematic situations.

The PSM also help to systematically generate other models during successive engineering activities. As an example, since the problem “Online travel booking takes too long time” is identified as a major problem that affects a higher priority purpose “Quick Responsiveness”, a DEM, depicted by figure 5, is created to reveal the best desired improvements to address the problem. The construction process of DEM is guided by, and makes referenced to, elements in the above PSM. For example, The desired improvement of efficiency during travel booking process includes a higher level *ideal*, called “Fast booking”, which is achieved via a number of more specific *Objectives*, such as “Service provider filtering”, “Caching”, ”Simplify booking process”, and “Parallel processing”, which, respectively, filters out slow service providers, provides cache to provide repeated information locally, simplifies the booking processes, and interacts with service providers in a parallel and asynchronous fashion. The last *Objective* contains some lower-level specific *Goal*s, such as “Parallel booking” and “Parallel queries”. The identification of these improvement and desired ends at different level of details helps to discover the "right" and achievable goals and desired capabilities to address the important problems.

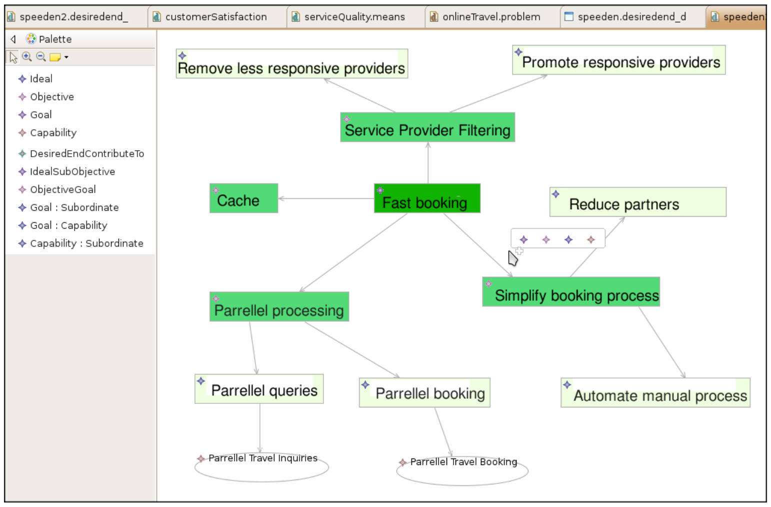

Fig. 5: An example of DEM in an online travel booking scenario

In OASEF, higher level models such as *System Abstraction*, PSM, and DEM are used to derive and capture intellectual achievement as crucial engineering resources. In the meantime, more concrete design is also captured by models. For example, UML Activity diagrams are used to describe the internal structure and behaviour of subordinate business processes in *Capability Design*, with the aim to achieve desired ends in DEM. A design of a process to "Close an existing FHSA account" is systematically derived in accordance with the identified problem of "Cannot manage FSHS account in current bank system" in its PSM. It contains interconnected required capabilities within an orchestrated structure, such as "Check Eligibility", "Acquire state information", "Validate Customer Closure Form", "Fund processing", and "Notify Customer Result". When conducting *Service Composition* during *Implementation*, the above *Capability Design* model is automatically transformed into more concrete formalism, such as Business Process Execution Language(BPEL) models, which can be directly executed in a run-time environment, such as Apache ODE BPEL engine used in the case studies. The transformation process from UML Activity Diagrams to BPEL can utilise various BPEL transformation tools such as the MDD4SOA tools set [8].

4 Evaluation and analysis

Some observation and assessment of OASEF are made during the design and implementation of proof-of-concept systems in real world settings. Due to a focus on, and methodological support for, higher-level intellectual efforts, OASEF helps to analyse and understand a complex situation, and to systematically derive sensible and rational business decisions, for example, in one of the case study, whether or not provide support for a bank business under specific legislation environment. The modelling mechanism and tool environment enables developers to concentrate on exploring and understanding higher order business issues including complex situations and desired business ends to be achieve, using models such as PSM and DEM. The graphical tools provided in IPEOAP facilitate the manipulation of these higher-level intellectual efforts, such as design and manipulation of *Abstraction*, *Problem Situation*, and *Desired End*. The use of unified EMF modelling meta-model enables interactions and cross-references among various OASEF activities, using either general purpose UML models or specially designed models such as DEM.

In the meantime, "accidental complexity" of underlying implementation technologies is largely hidden during the case study due to capturing and exploitation of engineering capacities by using *Epitomes* and automatic model transformation techniques. For example, the processes to create database persistence logic, web service provision and invocation, and web user interface are completely captured in various *epitomes*. Consequently, given a *System Abstraction* model such as information model for a flight or bank account, OASEF enables 100 percent automatic code generation, which produces systems capable of collecting information from users, persisting data, and providing relevant information service. Experience from the case studies shows that, based on well captured higher order models and technological capacities, generation of concrete systems at the implementation level is relatively fast, and requires little human manipulation.

Some issues and limitations are also revealed by the case studies. Although the proof-of-concept case studies were based on real world settings, the implemented systems are not assessed in real business environment. In fact, since the controlled case studies were designed to demonstrate and evaluate the objectives of OASEF, by the same person who designed the framework, the results are hence less convincing compared with empirical practice by third parties. Moreover, various OASEF activities are neither monitored nor validated in the engineering process, which makes it hard to identify problems when things go wrong. Furthermore, due to its proof-of-concept purpose, these case studies did not cover all aspects of the framework such as *sensation* and *control*. It hence requires further work to improve the prototype and conduct thorough empirical evaluation in practice, such as applying evaluation metrics to quantitatively assess the effectiveness in real world projects, ideally on a benchmark system, and in comparison with other approaches.

Based on the empirical experience from the case studies, OASEF is also compared with a range of other approaches, which have varying focus, objectives, and realisation methods. Kohlborn et. al. proposed a consolidated approach that aims to provide a good business and IT alignment by layering them separately with certain linkage in between [9]. For each layer, four stages, namely *Preparation*, *Identification*, *Detailing*, and *Prioritisation*, are used to progressively identify, elaborate, and provide desired services that, collectively, form the systems. However, the nature and content of its higher order activities such as "Conducting interviews", "Conducting capability analysis", and "Defining domains" are vaguely defined and lack practical guidance. Moreover, this work provides insufficient supports for capturing and managing both higher level business and lower level technologies in a flexible fashion within its predefined layers. Its strength is weakened in practice due to the lack of concrete formalism and modelling techniques. In comparison, OASEF, founded on sounds theoretical base, defines the scope, purpose, and relationship of its activities, and more importantly, provides specific modelling mechanism to manage them.

Lamparter and Sure also proposed an interdisciplinary methodology that combines a Web Service engineering method with market engineering and ontology that aims to coordinate services and customer in a collaborative environment [10]. Although it covers a full range of system analysis and design activities, no specific means is provided to manage uncertainty and volatility in a dynamic environment, which makes it less effective when changes are required in a timely fashion. Whereas OASEF attacks this issue by allowing flexible aggregations of previously-proven engineering capacities and process automation in a unified environment, in accordance with captured higher order rational justifications.

There are also a number of other model-driven approaches to service engineering [11, 8] that provides various modelling facilities and tools to map business processes into executable SOS. For example, a model driven technique is applied to build SOS by converting higher level business processes, captured in UML activity diagrams, into executable BPEL files [8]. Some tools are developed to facilitate reasonably smooth transition from business design captured in BPMN and activity diagrams, into system implementation [11, 8]. However, unlike OASEF, these approaches do not focus on, and provides no support for, some higher level intellectual efforts such as identification of problems and improvement. Although our previous work [7] also applied a model-driven approach to facilitate higher level modelling through conceptual *orientation* and *decision*, it does not realise our vision of service engineering due to some great difficulties, such as a lack of clear definitions and concrete formalisms for problem-level abstraction, and inadequate separation of various engineering concerns during each phase.

5 Conclusion

In this paper, we propose a synthetic approach to service engineering. It incorporates some inter-related elements including a conceptual foundation and guiding principles, a novel process model, a model-driven method, and an integrated supporting environment.

Two proof-of-concept case studies were conducted in real world settings, and are briefly presented in this paper to illustrate some important concepts and techniques. The results show that this approach can be applied in real-world settings to facilitate service engineering and achieve its design goals in general. More specifically, the following positive results are observed. Firstly, by using higher order models such as PSM and DEM, OASEF enables and promotes coherent higher order intellectual activities, by which effective services and systems are presupposed. Models in *Abstraction*, *Rationalisation*, and *Realisation*, are captured as important engineering resources that can be located and aggregated together. They hence 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 type of resources are organised in a unified, identifiable, reusable fashion. The coherently aggregation of resources and capacities are used altogether to systemically and automatically drive the engineering process to produce desired SOS.

Some important issues and limitations are also revealed during the prototyping and evaluation processing. Corresponding improvements and more thorough evaluation of its practical effectiveness in real world projects are essential to consider in future research.

References

1. Papazoglou, M., Traverso, P., Dustdar, S., Leymann, F., Kramer, B.: Service-oriented computing: A research roadmap. Service Oriented Computing (SOC) (2006)
2. Chang, S.H., Kim, S.D.: A service-oriented analysis and design approach to developing adaptable services. Services Computing, 2007. SCC 2007. IEEE International Conference on (2007) 204–211
3. Dijkstra, E.: The humble programmer, acm turing lecture 1972. Communications of the ACM **15** (1972) 859–66
4. Cardoso, J., Voigt, K., Winkler, M.: Service engineering for the internet of services. Enterprise Information Systems (2008) 15–27
5. Ackoff, R.L.: Ackoff's best, his classic writings on management. John Wiley & Sons, New York (1999)
6. Checkland, P.: Systems thinking, systems practice. J. Wiley, New York (1981)
7. Wang, Y., Taylor, K.: A model-driven approach to service composition. In: Service-Oriented System Engineering, 2008. SOSE '08. IEEE International Symposium on. (2008) 8–13
8. Mayer, P., Schroeder, A., Koch, N.: A model-driven approach to service orchestration. In: IEEE International Conference on Services Computing, 2008. SCC'08. Volume 2. (2008)
9. Kohlborn, T., Korthaus, A., Chan, T., Rosemann, M.: Identification and Analysis of Business and Software ServicesA Consolidated Approach. IEEE Transactions on Services Computing (2009) 50–64
10. Lamparter, S., Sure, Y.: An interdisciplinary methodology for building service-oriented systems on the web. In: Services Computing, 2008. SCC '08. IEEE International Conference on. Volume 2. (2008) 475–478
11. Brambilla, M., Dosmi, M., Fraternali, P.: Model-Driven Engineering of Service Orchestrations. In: Proceedings of the 2009 Congress on Services-I-Volume 00, IEEE Computer Society (2009) 562–569