Between Service Science and Service-Oriented Software Systems

Yuhong Yan
Faculty of Computer Science
University of New Brunswick
Canada
Yuhong.yan@unb.ca

Juergen Bode
International School of Management
Dortmund, Germany
juergen.bode@ism-dortmund.de

William McIver, Jr.
Institute of Information Technology
National Research Council
Canada
Bill.McIver@nrc.gc.ca

Abstract

Some IT companies, such as IBM, are pushing research and education in Service Science. This is due to the transition of their business models from producing hardware and software to providing services, as well as service-oriented vision changes the way people design software systems. People believe that service is the common language between business developers and IT engineers and service bridges the gap between business requirements and IT implementation. But people still do not know clearly how to use the service related theories in software system analysis and design. This paper attempts to explore this unknown topic. We want to reveal how software can become a service and suggest guidelines for analyzing and implementing service-oriented systems.

1. Introduction

Service Science is promoted by many leading IT companies, such as IBM and SAP. These companies have been changing their business models from producing hardware and software to providing services. These services can be consulting services, application management, system maintenance, etc. At the same time the concept of Service Oriented Architecture (SOA) has become prominent. Services can be regarded at a higher level of abstraction than objects in object oriented programming. A service can be implemented by an object, a group of objects, or even a business process. Under this concept, the system should be analyzed at the service level, instead of the object level. People also expect that working at the level of services can bridge the gap between business development and IT realization. Though this promise is still to be realized, people have invented many open standards and their implementations around SOA to build real world systems. We can see that from a software engineering point of view, a service-oriented vision changes the way people design software systems, as well as how IT engineers provide services to their clients:

- How the nature of services influences software system design;
- What is the proper procedure for service engineering (to implement a software system using engineering methodology for fulfill the needs);
- How to bridge IT implementation to business requirements;
- What are the basic composite modules for a service-oriented system;
- The guidelines of service-oriented system design; and
- Reference architecture for service-oriented systems.

This paper discusses partially these problems. Starting from analyzing the emerging of Service Science, Management and Engineering (SSME) (Section 2), we get into the nature of services (Section 3). Then we discuss all the nature of services influences service-oriented system design (Section 4). Section 5 is a preliminary discussion on system engineering issues for service-oriented systems.

2. IT companies in service economies and the emerging of SSME

The subject of service economies is often raised in discussions about SSME. We want to show in the analysis below how IT companies transform their business models and show the service opportunities IT companies have. We also want to compare the different scopes of SSM and SC and their relationships to Software Engineering.

Post industrial countries have entered the so-called service economy era – where the service sector is the largest component of their economies. This number is over 80% for US and over 70% for Canada. What does this mean to IT companies? Let's check the business transformation in IBM in the recent years.

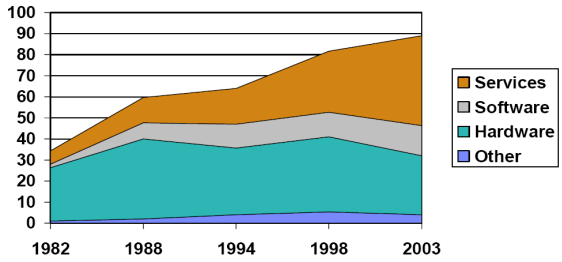

Figure 1. IBM moves towards providing services [1]

IBM has three major businesses: hardware, software and services (Figure 1). The services component has increased the most among the three (the top layer, or brown in color print). IBM is transforming its business from producing hardware and software to provide services. In 2005, services account for 53% of revenues and 35% of pre-tax income [2]. Figure 2 shows the breakdown of these numbers. The major services opportunities for IBM are: Strategic Outsourcing (SO), Consulting, Integrated Technical Services (ITS), Business Transforming Outsourcing (BTO), Application Management, and Maintenance. Through these services, IBM provides full IT solutions, as well as b-2-b solutions, to its partners. For example, in SO, the other companies are able to build strategic partnerships with IBM. IBM manages and operates the applications and IT systems for these companies under a mutually beneficial agreement. The outsourcing agreement may include the transfer of IT employees and IT assets to IBM. IBM provides service level assurances to ensure quality of service is attained and measured.

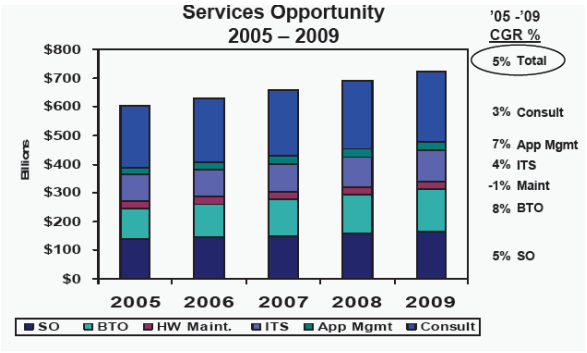

Figure 2. The service business in IBM [2]

In these services, customer value is created through reducing IT costs, improving IT service quality, providing the flexibility to transition and transform to the next generation of infrastructure and applications. Partner companies gain access to industry, business and technology experts and better focus on their core competencies.

Many variations of service science research and education exist today within the domains of management, operations research and economics to name a few. IBM introduced the new discipline Service Science, Management and Engineering (SSME). According to IBM, SSME is a multidisciplinary domain that involves knowledge from Business Administration and Management, Economics and Marketing, Social Sciences, as well as Science and Engineering (Figure 3). This new discipline studies the concept of service and its related phenomena as a whole. Compared to the existing studies, SSME seeks to understand the complexities of business-to-business services, while the existing studies concentrate on business-to-consumer services. Of particular interest is the nature of the impact of technology, people and process on how services are defined and delivered [3]. We should emphasize that SSME is also about technology innovation, especially IT innovation to help delivering services.

We claim that SSME is more of a management discipline than an engineering one. In the SSME open course offered by the IBM Almaden Research Center, the definition of service is purely an economics concept: "In economics and marketing, a service is the non-material equivalent of a good". Therefore, the nature of services is analyzed in the contrast of goods (cf. Section 3).

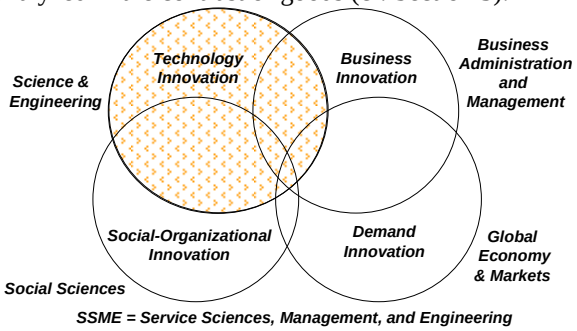

Figure 3. The scope of SSME [2]

The technology aspects of SSME (the shaded circle in Figure 3) can be called Service Computing (SC). In [6, p. 17], service computing is discussed as an emerging cross-discipline that covers the secience and technology of effectively creating and leveraging computing and information technology to model, create, operate, and manage business services. The core technology suite includes SOA and Web services, business process integration and business performance management, and service innovation methodology. We point out that [6] lists service-oriented business consulting, business process modeling, transformation and integration, and business performance management within the scope of SC.

For software engineering, we need to adapt to the new demands and equip ourselves with the new knowledge and

skills of SC. We need to study the new methodologies to design systems and provide services to our clients.

3. The nature of services

In economics and marketing, a service is the non-material equivalent of a good. It is further claimed that a service represents a process that creates benefits by facilitating either a change in customers, a change in their physical possessions, or a change in their intangible assets. We can give many examples of everyday services where this is the case [3]: transportation (trains, planes, delivery), hospitality (hotels, restaurants), infrastructure (communications, electricity, water), government (police, fire, mail), financial (banking, investments), entertainment (television, movies, concerts), and professional services (education, medical care). People have studied the nature of services in contrast to goods in management science. Table 1 lists some of the properties often attributed to services. We also know that these properties do not apply to all cases.

Table 1. Attributes to Services

Properties	**Examples**	**Problematic in the case of**
Simultaneous production and consumption	Surgery	Parcel transportation
Customized	Consulting, auto repair	Television, parcel transportation, hotel chain
Requires personal presence	Hair dressing	Mail delivery
Impossible to store	Theater, airline	Parcel transportation, auto repair
Labor intensive	Consulting, retail	Airlines, resorts
Intangible	Education Entertainment	transportation, dry cleaning

Normally services are provided at the same time they are consumed. It is a coproduction process that the customer's inputs are necessary. The positive example to support this is surgery that the patient receives the service when the doctor provides the service. But the negative example is parcel transportation. Though the customer gives the input of the service, the service is taken on after the customer's involvement.

The coproduct process brings up the other attributes. Services are customized to the customer's needs. From medical care to consulting, the experts tailor their solutions to individual customers. Yet there are standardized services, such as television and services in hotel chain that they do not distinguish their customers' individual needs. On the contrary, they want to standardize their services to meet all kinds of people.

As it is a coproduct process, the presence of the consumer is normally necessary, e.g. hair dressing. But some of the services take place without the involvement of consumers, e.g. mail delivery.

Also from simultaneous production and consumption, service is normally impossible to be stored. But some exceptions are parcel transportation and auto repair where the services can be queued and delivered with a delay.

Services are normally labor intensive, e.g. consulting and retail business. But by offering large scale and standardized services labor usage can be reduced, e.g. resort services.

The last attributed is intangible. It is unique to services in contrast to goods, because services can be applied to people's mind, e.g. education and entertainment, or intangible asset, e.g. banking and legal services. But services can also be tangible that they are applied directly to goods or other possessions, e.g. transportation and drying cleaning.

Since there are always positive and negative examples for the above attributes of services, what characterizes services is not these properties, but the process. The process of services in Figure 4 has two sets of inputs: internal factors from the service provider and the external factors from the customer. The output of the service is to satisfy the needs of the customer. Please notice that the existence of an external factor is a constitutive element of services. Therefore, according this definition, standard software or music on CD is not a service. Services are transformation processes which satisfy economic needs and are performed upon customers or customers'objects.

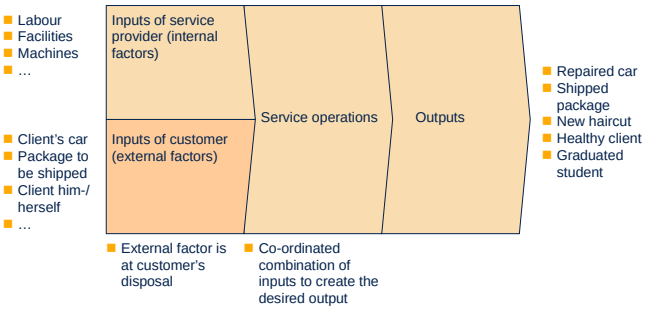

Figure 4. The process of services

This paper, we do not discuss how to provide the consulting services or the maintenance services to clients. We want to explore how to design a software system using service-oriented vision.

4. The nature of service-oriented software systems

Service Oriented Architecture (SOA) is a new paradigm to design distributed systems. In SOA, services are autonomous, platform-independent entities that can be described, published, discovered and loosely coupled. Services represent a higher level of function abstraction compared to objects in object-oriented programming. Our real world projects are in the domains of health care and emergency response. SOA are used to build a common platform to integrate different data sources and connect standalone applications. Web services can be regarded as an implementation of SOA. Web services are based on a set of W3C and OASIS open specifications: SOAP for communication, WSDL for service interface description, UDDI for service publishing and index, BEPL for Web service process description. The advantages of Web services are interoperability, function encapsulation and abstraction, loose coupling, reusability and composability. Especially, its interoperability is based on adopting XML and HTTP, the two techniques that are widely accepted by many operating systems and programming languages. People are working on new specifications that define all aspects of a service-oriented architecture (e.g. security, semantics, etc).

We define a **Service-Oriented System (SOS)** as a software system that provides services according to the our formal definition of service. We need to know what makes a software system become a service defined in Section 3. What kind of attributes does a SOS have? How can we build a SOS from the top (i.e. business requirements) down (i.e. decomposing the business requirements into service modules)? These are the questions we want to discuss in this paper.

4.1. Service-oriented systems vs. object-oriented systems

Currently most software systems are implemented in object-oriented programming languages (OO languages). OO languages have the features of encapsulation, inheritance, modularity and polymorphism. In an OO runtime environment, the state of an object can be defined as an assignment of values to its properties, but only by invoking external methods (or functions) of an object can the state of an object can be changed.

A service in the form of software should be a process that takes in internal and external factors, has service operations and fulfills customer needs. The external factors are the information about the customer requests. The internal factors are the resources and their settings from the service provider. Service operations coordinate the inputs and create the desired outputs. We want to show that not all software systems are service-oriented systems. Only when software is in a form in which processes comprising a service can be specified can software be regarded as a SOS. A comparison between SOS and OO systems is listed in Table 2.

We can use current SOA techniques to implement a SOS; however, current SOA is not a service-oriented programming methodology. SOA is, rather, a higher level abstraction than OO programming with respect to its uses in analyzing and implementing SOS. At design time, we need to translate business request into XML formatted messages, e.g. SOAP or REST messages. The service operations can be implemented as business processes or objects. The service operations are made available to the outside world in the form of an interface description file (e.g. WSDL). At the execution time, the customer's needs are satisfied by processing the received business requests.

Using current SOA techniques, service innovation is pretty much like a remote procedure calls. There is no methodology to map business needs to messages that bring external factors to the SOS. This is done at the design time manually through designing the services. The process of business requests (within the messages) is pretty much fixed. Customization promised by a service is not achieved. This may provoke the study of service engineering – how to design a SOS to process business needs directly and how to make the processes more flexible and adaptable.

For us, OO system is lower in function abstraction. An object itself is does not fulfill any customer needs. If we want to make a OO system to fulfill a business need, we need to translate the business request to parameter level – lower than SOA abstraction.

Table 2. Compare object oriented systems with service-oriented systems

	Object Oriented	**Service Oriented**
Design Stage	Define object statics (classes, attributes, constraints); Define object relations; Define object dynamics; Implement business logic inside objects;	Translate business requests into XML formatted messages; Translate service operations into service interfaces; Implement service operations as business processes or objects;
Execution time	Objects call each other and change their states; System status are changed;	Services send messages to invoke each other; Business requests are processed; Customer needs are satisfied

4.2. What makes a SOS?

The elements in the process of services in Figure 4 are the necessary conditions for a service from the economics point of view. We consider that similar conditions make a software system becomes a SOS. We present the following hypotheses. We believe these hypotheses should be true for any SOS.

Hypothesis 1: the software uses both external and internal factors for service operations (coproduction).

Hypothesis 2: the software system satisfies the needs of the customer.

Hypothesis 3: the software system has a process to fulfill the business requests.

From hypothesis 1, external factors are necessary for a SOS. The customer inputs can be simply a customer order. The importance of the condition is that the service process can only start when there is a triggering condition from the customer. This also suggests a SOS to analyze customers and their objectives in order to provide better services. The internal factors of a SOS are the resources and facilities that the service provider puts into the SOS.

As a service system, a SOS satisfies some needs of the customer (hypothesis 2). As a service system, the satisfaction of the customer depends on the perception of the services and the perception can be individual. Different people may give different evaluation of the services. The criteria of QoS are mostly objective measure of the quality of service. We can see that the current QoS mostly is determined by the internal factors of a SOS. Some people have proposed customer satisfaction to be included into QoS [10].

We can see that the service is a process that accepts the inputs, execute the operations and produces output (hypothesis 3). The service operation itself can be a process composed by other services. This probably provides the flexibility of a SOS.

Besides the above hypotheses, the following properties also characterize a SOS.

1. **SOS is a Self service**. A SOS is an automatic system that runs without human intervention. Therefore, it is a self-service for its user.
2. **SOS can handle demand fluctuation easily**. As service cannot be stored, how to handle needs fluctuation is not easy. In traditional service system, the service capacity may be exhausted so that the demands at the peak time may not be processed in time. SOS can handle this easier than traditional service system. The software service can easily be duplicated and deployed in more servers at peak time.
3. SOS can handle tangible goods as well as intangible assets and information.
4. SOS does not necessarily mean customization. Though accepting external factors gives the possibilities that SOS can deliver highly customized services, this is not necessary for a SOS to do. Currently, many business processes are standardized for all the customer requests.

5. System engineering for SOS

Since SOS is a new schema to develop software systems, we need a set of new methodology to analyze and develop such kind of systems. We propose some of the issues related to system engineering for SOS.

Service Analysis and Identification

Services are designed on the business level not on the IT-level. Services are supposed to be a common language between the business developers and the IT engineers. Services should satisfy customer's needs. Service can be a composite business process. Then start from customer's needs, how do we identify services? Or in another word, how do we group/partition the functions/requirements into services? Here are some preliminary ideas inspired by [11]:

1. Balance between small and big granularity. If a service too small, the orchestration logic can be too complex. And communication overhead can be heavy. If a service is too big, that means big parameter list and multiple purposes. It can increase the complexity to implementation and reduce the reusability of services.
2. Group the repeated invocation patterns into a service. If functions A and B are detected to be used together in various cases, grouping them into a service means this service can be shared by multiple business processes. Therefore, the reusability of this service is increased.
3. Satisfy customer's needs. It is different from objects (cf. Table 2). Services ices. One entity that can be called service should satisfy customer's needs.
4. Services suggested by industry standards (e.g. CIDX, RosettaNet).
5. Analyze service experiences for SLA and QoS. Keep it in mind that Quality of Service (QoS) depends on the customer's perception of services. It can be different to different people. Service experience in an important indicator when designing services. Further, it is important to determine Service Level Agreement (SLA).
6. Analyze business context, such as organization structure, roles, and business processes.

There are existing methodologies in software engineering to acquire user requirements. We also need to investigate if these methodologies can be applied to SOS.

Service Definition and Modeling

This is a step to define the technical aspects of services, such as interfaces of services and interactions among services. Services are the fundamental building blocks for SOS. We need to identify, specify and realize the right services. We need to model the following issues. Please see Table 3 for more details.

- Service functions
- Service provider
- Service non-functional aspects

Table 3. Service Definition and Modeling

	Attributes	Details
Service functions	Service operations	Interface name Interface parameters
	Service operation description	e.g. WSDL
	Location	e.g. SOAP service end point
	Interactions	With other services
	Processes	For process implemented service
Service provider	Business information	e.g. UDDI entry
Non-functional Aspects	QoS Policy Security/Privacy SLA	

Table 3 is just a very rough consideration on how to model services. People already see that services should be in different types and implemented in different patterns. Though not knowing what kind of patterns or templates the services should follow, people have proposed different solutions in their realization of SOS.

We can imagine that we need to have multiple service types and each follows a special kind of design patterns. In [12], the services are in the following types:

- Application Service
- Business Service
- Control Service
- Coordinator Service
- Entity-centric Business Service
- Hybrid Service
- Integration Service
- Process Service
- Task-Centric Business Service

This classification probably will not be followed by industrial companies. But it reveals the fact that different types of services can be designed for fulfilling a special kind of tasks.

In other industrial products, services are mixed with objects. SAP has the following types for their enterprise services system [11]:

- Business objects
- Service interfaces
- Process models
- Business scenario and business process objects
- Mapping objects

Though it is still an open question about designing the proper types of services suitable for all kinds of problems, people agree that there are some common principles to follow:

1. Well defined based on open standards. Following industrial standards can improve the interoperability and reusability of services.
2. Capable for loosely coupling.
3. With the appropriate granularity and correct abstraction level.
4. Correspondent to the demands of the business processes.

A Very Simple Reference Architecture for SOS

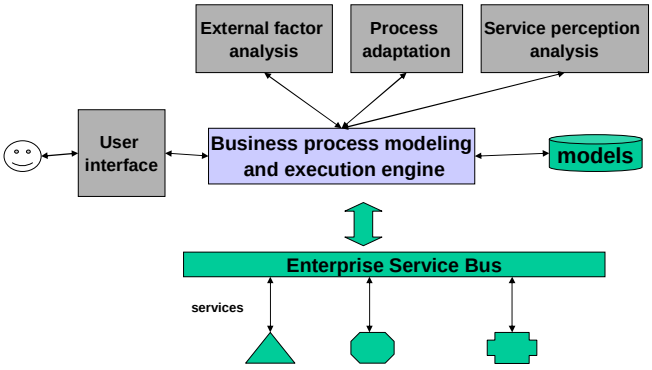

Figure 5. A very simple reference architecture for SOS

According to the issues we discussed in this paper, we propose a very simple reference architecture for SOS (Figure 5). It is centered by a business process modeling and execution engine. The services can be connected to the execution engine via a service bus. Though many systems use other architectures other than service bus, such as hubs or one-to-one interfaces, service bus has its advantages to build complex and large scaled systems. As we have discussed through the paper about the nature of services and SOS, we need to have functions to make the SOS more qualified to provide services to the customers. Therefore, the modules like external factor analysis, process adaptation and service perception are proposed. We consider these are the functions needed for a SOS.

How to realize these modules are out of the scope of this paper.

Compared with other existing reference architecture, our simple architecture illustrates the demands from service-oriented vision. It does not include other features that are possibly necessary to build a system, e.g. security features. The main purpose of this reference architecture is to inspire people to study service-oriented features.

6. Conclusions

This paper discusses several interesting topics related to how service science can influence service-oriented software system design and development. Starting from the nature of services from a management science point of view, we want to reveal how software can become a service and suggest guidelines for analyzing and implementing service-oriented systems. We want to inspire people to work on this direction in order to build efficient, reliable and adaptive service providing software systems.

References

[1] Robot Marris, a IBM presentation. Unknown source.

[2] Patricia Murphy, "IBM Business Perspective 2006", *www.**ibm**.com/investor/events/events_**2006**.phtml*

[3] IBM Almaden Services Research, "SSME course modules", *http://www-304.ibm.com/jct09002c/university /scholars/skills/ssme/resources.html.*

[4] IBM Almaden Services Research, "SSME course modules", *http://www-304.ibm.com/jct09002c/university /scholars/skills/ssme/resources.html.*

[5] Klause-Peter Faehnrich and Kyrill Meyer, "The perspective of Computer Science", in Bernd Stauss, Kai Engelmann, Anja Kremer and Achim Luhn (eds.), *Services Scince: Fundementals, Challenges and Future Developments,* Springer, 2008. p103-109.

[6] L-J Zhang, Jia Zhang and Hong Cai, "*Service Computing*", Springer, 2008.

[7] Yuhong Yan, "Service Computing: Foundations, Design and Implementation", *http://www. flydragontech. com/ebcourseWinter2008/6905outline.htm.*

[8] Wikipedia, "Open source", *http://en.wikipedia.org/ wiki/Open_source*.

[9] Maksym Petrenko, Denys Poshyvanyk, Vaclav Rajlich, and Joseph Buchta, "Teaching Software Evolution in Open Source", *IEEE Computer*, Nov. 2007 p25-31.

Barcia, E. and Striuli, A., "Quality as a measurement of customer perception and satisfaction inmobile TLC", *the proceedings of 1996 International Conference on Communication Technology Proceedings, (ICCT'96),* 5-7 May 1996, p400-403 vol.1.

[11] SAP, "Enterprise Service Design Guide", *www.sap.com/platform/netweaver/pdf/BWP_ES_Design_ Guide.pdf.*

[12] Thomas Erl, "*SOA: Principles of Service Design*", Prentice Hall, July 2007.