10-2015

Using a Work System Metamodel and USDL to Build a Bridge between Business Service Systems and Service Computing

Steven Alter
University of San Francisco, alter@usfca.edu

Alistair Barros
Queensland University of Technology, alistair.barros@qut.edu.au

Recommended Citation

Alter, S. and Barros, A. (2015) "Using a Work System Metamodel and USDL to Build a Bridge between Business Service Systems and Service Computing," working paper.

USING A WORK SYSTEM METAMODEL AND USDL TO BUILD A BRIDGE BETWEEN BUSINESS SERVICE SYSTEMS AND SERVICE COMPUTING

Abstract

This paper explores the support for more comprehensive modeling of service systems than that possible through modeling methods developed through partial perspectives, with uncertainties about their wider suitability and need for integration with other methods in this domain. It responds to a Dual Call for Papers from INFORMS Service Science and IEEE Transactions on Service Computing requesting contributions that address the barely explored challenge of establishing links between business views of service systems and more technical views from service computing. Competing definitions of service reveal that most business views of service emphasize acts or outcomes produced for others, whereas a service computing view emphasizes encapsulated functionalities that can be discovered and launched by service consumers. This paper uses work system theory (WST) and a related work system metamodel to represent a business view of service systems. It uses the Unified Service Description Language (USDL 2.0) to represent a service computing view of service systems. Application of the business view to the previously defined EU-Rent example illustrates how successively more detailed business-oriented descriptions of a service situation reveal needs for functionality that are well described by USDL. In other words, business service system views and service computing views, as represented by WST and USDL respectively, serve complementary purposes. WST supports modeling and analysis of business situations, while USDL is the basis of detailed descriptions of services as encapsulated functionality.

Keywords: *Service, service system, business service, service computing, work system metamodel, USDL*

1. The Challenge of Reconciling Contradictory Views of Service

There are fundamental contradictions between many characteristics of service as it is generally perceived in the business and social world versus characteristics of service that are required in the service computing world. In both worlds, service involves entities performing activities for other entities. The contradictions appear when one looks just a bit deeper.

In the business and social world, common understandings, theories and research related to service tend to view services as sociotechnical activities involving people who may or may not use technologies as they try to facilitate beneficial outcomes for others. Automated services such as telecommunications and automated search are evident in the business and social world, but at first blush seem more like technical infrastructure and are not the first examples that come to mind when most business people think about service. Views and theories of service that are articulated by business researchers often include concepts such as coproduction, value co-creation, customer experience, and awareness of customer needs, desires, and emotions. Those concepts imply that providers and customers engage in collaborative activities that typically involve mutual visibility, adaptation, and mutual empathy between the provider and customer.

The world of service computing requires a totally different approach by treating services as encapsulated functionalities that purposefully separate client entities from server entities. Those functionalities are launched by messages in a predefined format, produce responses in a predefined format, and are governed by explicit rules of engagement that determine which client entities have the right to request service from which server entities. The concept of encapsulated functionality minimizes the mutual visibility of the client and server. Server entities have no awareness of the status, needs, likes, and desires of the client entity beyond the specific information in a preformatted message that launches a service. Similarly, client entities have no visibility of the specific activities through which a server executes services except for pre-specified information in the server's response message. The great benefit of this approach is that it supports service representation in catalogues, programming architectures and methods based on modularity, loose coupling, and high cohesion that facilitate assembling computing systems from separate modules that can be defined and tested individually and ideally can be configured dynamically as needed.

These contradictory views of service are a source of confusion about the content and nature of service science and are a significant obstacle to meaningful conversation between researchers coming from different research traditions. In the business and social world, visibility and mutual empathy are viewed as commonplace and often expected as inherent in high-quality service. The service computing world expects and requires exactly the opposite.

Goal and approach. This paper addresses a Dual Call for Papers (Goul et al. 2014, p. 1) from INFORMS *Service Science* and IEEE *Transactions on Service Computing* that was summarized in the abstract. This paper's goal is to establish a bridge that overcomes the seemingly irreconcilable differences between the business/social versus computing views of service. It does this by demonstrating links between two representative sets of ideas. The business/social view of service is represented by work system theory (WST) and a related metamodel. The computing view of service is represented by the Unified Service Description Language (USDL).

The conceptual core of this paper's approach for establishing the bridge is treating "degree of encapsulation" as a service design variable whose extremes are "no encapsulation" (i.e., extensive visibility and direct collaboration by customers) and "total encapsulation" (no visibility or collaboration beyond information in predefined messages). Intermediate points between those two extremes involve combinations of collaborative activities and encapsulated activities. The actual operation of today's business world occurs primarily at those intermediate points, with a strong trend in the direction of greater encapsulation. In other words, the rhetoric of service in the business/social world seems to underemphasize the widespread presence and significance of automation, while the vocabulary and operational details of service in the computing world seem to underemphasize the importance of human sensibilities and human-to-human interaction

Approach. Many modelling methods, techniques and languages have been proposed for capturing information systems where value is derived largely from services, i.e. service systems, through resources, processes, systems, partners, customers and interactions,. Many of these methods, techniques and languages claim suitability for both business and IT aspects of service systems, but generally are based on conceptions anchored in one aspect, and then extend or adapt concepts from the other. Thus, the alignment and integration of complementary concepts and methods targeting service systems remains a topic for exploration.

This paper's approach is to provide insights by summarizing two complementary approaches to service, one from each tradition, and comparing their strengths, overlaps and gaps. Specifically, it uses work system theory (WST) and a related work system metamodel to represent business-oriented views of a sociotechnical service system. It uses the Unified Service Description Language (USDL[1]) to represent a detailed, technical approach to encapsulated functionality, which is one of the central concepts of service computing. Those views are illustrated using a reference example from the standards organization OMG. The example illustrates the complementarity between a business-oriented description of a sociotechnical service system based on WST and the encapsulated functionalities that can be described using a USDL specification. Different forms of possible integration between WST and USDL will be explored.

Value. To the authors' knowledge, this type of link between USDL and a business-oriented view of a sociotechnical system has not been demonstrated previously in the literature. Of particular value is the enhanced visibility of how to move from different levels of description and analysis of sociotechnical service systems to detailed specifications of service computing functionalities that can be encapsulated, discovered, and used in a very wide range of situations.

Organization. This paper proceeds as follows. First, it cites competing definitions of service from different disciplines and proposes that the definitions boil down to three basic approaches, two of which fit best with a business view of service, while the other fits best with a computing view of service. A summary of WST and the related work system metamodel provides concepts for describing sociotechnical service systems and demonstrates that most of those concepts also apply to totally automated systems. A summary of USDL identifies its goals and core modules. A conceptual comparison between the complementary views identifies areas of overlap and inconsistency. Application of work system concepts to summarize an example from the Object Management Group (OMG) demonstrates how increasingly detailed representations of a typical sociotechnical example reach a point where encapsulated functionalities in the style of service computing play an obvious role. The same sequence of representations demonstrates that totally encapsulated functionalities from service computing do not provide faithful representations of sociotechnical service systems. Reflections on the example lead to conclusions about whether and how it is possible to link business/social views of service systems and service computing views of such systems.

Comment about terminology. This paper discusses the relationship between a business-oriented view of sociotechnical service systems based on WST and a service computing-oriented view based on USDL. USDL was designed to cover totally automated service systems that operate in networked environments and also to capture the sociotechnical service systems context. This paper's initial sections use work system, service system, and business service system to refer to sociotechnical systems from the business perspective inherent in WST. The rest of the paper recognizes that those systems can be described using USDL, but

[1] The paper uses USDL 2.0 given the core concepts of service concepts established through this version of the language. Developments to USDL beyond this version have focused on the incorporation of an Open Linked Data, which is not relevant for the analysis of this paper.

that USDL and related languages and tools focus primarily on service computing rather than on describing work systems and service systems from a business viewpoint.

2. Competing Definitions of Service

Table 1 shows typical definitions of service from different disciplines including marketing, production management, economics, IT management, and computer science. (Most were cited in Alter, 2012a). These definitions are classified into three general portrayals of services as indicated in the first column. Some definitions focus more on acts performed by service providers, some focus more on outcomes perceived by customers, and others focus more on encapsulated functionalities that can be discovered when needed and then used after being triggered by a request or precondition.

Portrayal	Definition
acts	"an act or performance that one party can offer to another that is essentially intangible and does not result in the ownership of anything." (Kotler and Keller , 2006, p. 402)
acts	"intangible activities customized to the individual request of known clients." (Pine and Gilmore , 1999, p.8)
acts	situations in which "the customer provides significant inputs into the production process." (Sampson and Froehle, 2006, p. 331)
acts	"value-creating support to another party's practices" Grönroos (2011, p. 285)
acts	the "application of skills and knowledge (operant resources) for the benefit of another party" (Vargo and Lusch, 2008, p. 6)
outcomes	"a time-perishable, intangible experience performed for a customer acting in the role of a co-producer." (Fitzsimmons and Fitzsimmons, 2006, p.4)
outcomes	"a change in the condition of a person, or a good belonging to some economic entity, brought about ... [by] some other economic entity, with the approval of the first person or economic entity." (Hill, 1977, p. 318)
outcomes	"an essentially intangible set of benefits provided by one party to another." (Clerc and Niessink, 2004, p. 104)
outcomes	"A means of delivering value to Customers by facilitating Outcomes Customers want to achieve without the ownership of specific Costs and Risks." (ITIL, 2011, p. 66)
encapsulated functionality	A service "is generally implemented as a course-grained, discoverable [business and/or] software entity that exists as a single instance and interacts with applications and other services through a loosely coupled (often asynchronous), message-based communication model." (Brown et al., 2005) …. "The component that consumes business services offered by another business component is oblivious to how the provider created the business service." (Cherbakov et al., 2005)
encapsulated functionality	"Services constitute encapsulated and exposed functionality drawing from core artifacts, e.g., those related to business processes, applications, objects, and resources ..." (Oberle et al, 2013, p. 158) ... A service can be manual, semi automated and fully automated, or abstract." (p. 164)

Table 1. Past definitions of service, clustered as three portrayals of service

The three portrayals of service in the first column of Table 1 suggest three related candidates for the definition of service:

1) A service is an act performed to produce outcomes for the benefit of others.

2) A service is an outcome produced for the benefit of others.

3) A service is an encapsulated functionality that produces outcomes for the benefit of others after being triggered by a request or precondition.

This paper assumes that the first definition of service is simplest and most natural in everyday business situations, such as providing food services, gardening services, or police services. It encompasses the other two definitions because production of outcomes for others requires activities. The second definition applies most directly to controlled, contract-driven situations, such as IT services performed under service level agreements. The third applies most directly to delegated production of precisely defined outcomes by human or automated agents that will produce those outcomes independently, with no oversight or visibility for the requesting entity. It describes service computing by explicitly treating a service as an encapsulated functionality that performs activities triggered by a request or precondition.

3. Service Systems as Work Systems

The desired integration between the business and service computing view needs to be achieved at the level of service systems, not just the definition of service. Service systems are organizational systems whose operational parts, notably services, relate directly or indirectly to organizational phenomenon. As such, services, business processes, organisational actors and resources, IT systems etc., should be traceable across systems, operations and strategy. This section explains how work system theory (WST) leads to a work system metamodel that is equally applicable to business service systems because business service systems are work systems, an idea suggested as a "fresh approach in the IS field" by Alter (2010).

Definition of work system. A work system is a system in which human participants and/or machines perform processes and activities using information, technology, and other resources to produce product/services for internal or external customers. (Product/service will be defined below). Enterprises that grow beyond an improvised start-up phase consist of multiple work systems. Typical business enterprises contain work systems that procure materials from suppliers, produce products, deliver products, find customers, create financial reports, hire employees, coordinate work across departments, and perform other functions. Almost all of those work systems include totally automated subsystems whose work is performed by software. Those subsystems are also work systems because the definition of work system covers both sociotechnical and totally automated work systems. Some work systems cross organizations, e.g., supply chains or other interorganizational systems.

The approach to work systems discussed here results from a long term attempt to develop a systems analysis method that typical business professionals could use to understand systems in organizations in whatever way would be most useful for them. That effort developed the work system method (WSM), which has been used by many hundreds of MBA and Executive MBA students in the United Stated, China, India, Vietnam, and possibly elsewhere (Alter, 2013, Truex et al., 2010). Various versions of WSM that have been used in different settings all focus on identifying a problem or opportunity, summarizing the "as is" work system, analyzing the situation, and recommending a proposed, "to be" work system

Difference between work systems in general and service systems in general. All service systems in organizations are work systems because they satisfy the above definition of work system. Almost all work systems in organizations are also service systems because they exist to produce outcomes for the benefit of others within the same enterprise or outside of the enterprise, such as external customers. The rare exceptions in organizational settings are work systems that produce outcomes for the sole benefit of their participants, e.g., a salesperson's creation and maintenance of a personal shadow system for keeping track of customer information that is not recorded in the organization's CRM system. This paper treats the terms work system and service systems as synonyms since work systems that are not service systems are unimportant for its purposes. The term work system will be used more often in reference to past research about work systems and work system theory (WST) and in reference to a work system metamodel. The term service system is used more often in relation to research specifically about service and service systems.

Work system theory. A work system metamodel will play a central role in this paper's explanation of the bridge between business service systems and service computing. That metamodel is one of a number of extensions of work system theory (WST), the theory underlying WSM. WST consists of three components that will not be discussed in detail here but have been discussed elsewhere (e.g., Alter, 2013; 2015).

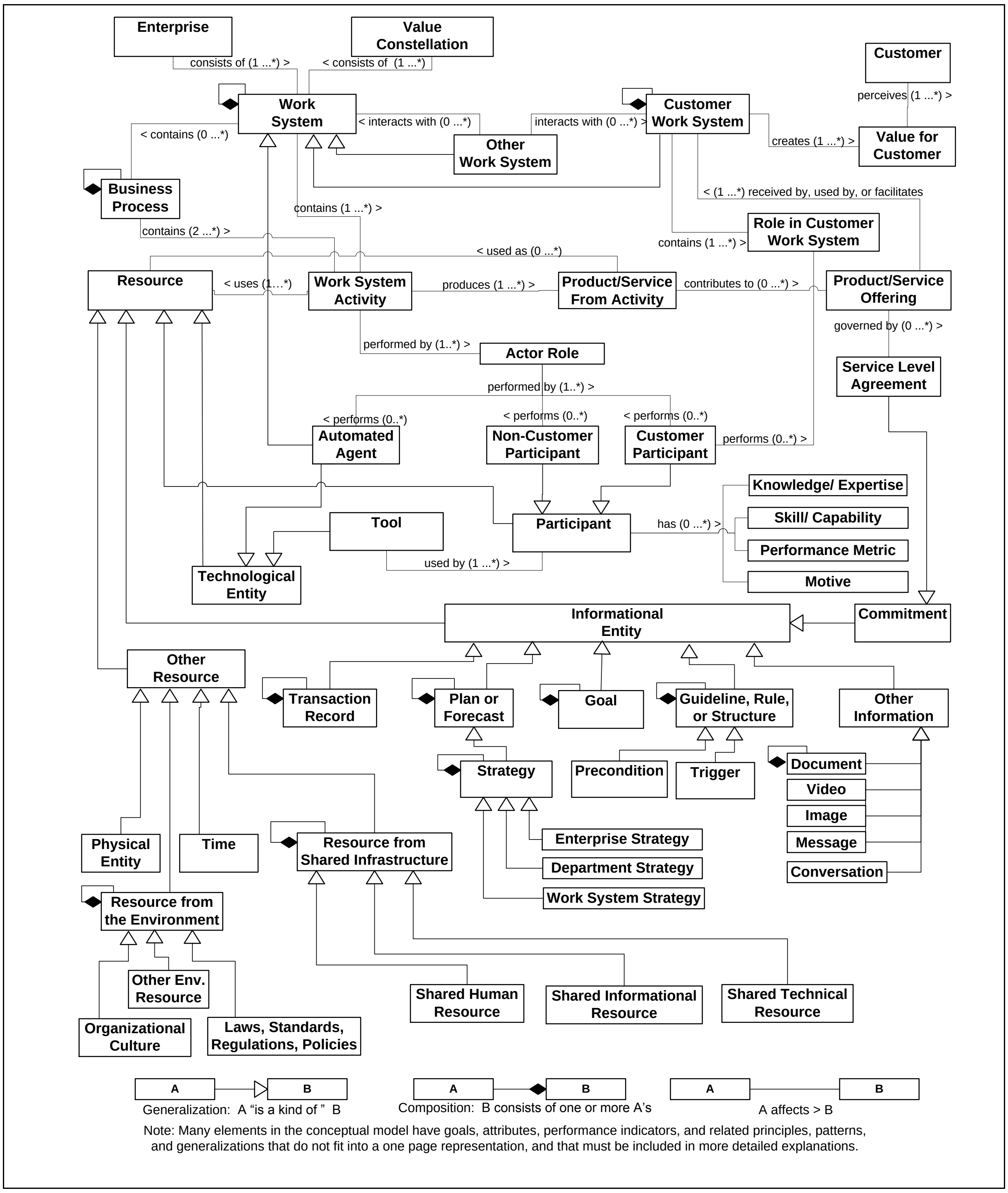
1) the definition of work system,
2) the work system framework, which identifies nine elements of a basic management understanding of a work system.
3) the work system life cycle model, which represents iterations through which work systems evolve over time via a combination of planned and unplanned change.

3.1 Work System Metamodel

Figure 1 is the fifth of a series of work system metamodels (e.g., Alter, 2012b) that outline more detailed views of a work system than are provided by the definition of work system (above) or by the work system framework (Alter, 2013, p. 78). The latter framework represents a basic, business-oriented understanding of a work system in terms of nine elements: customers, product/services produced, processes and activities, participants, information, technologies, environment, infrastructure, and strategies. The work system framework is useful for summarizing a work system and achieving mutual understanding of its scope and nature, but is less effective for detailed description and analysis. The more complete and rigorous metamodel supports more detailed description and deeper analysis without requiring specialized IT or computer science concepts and notations. The metamodel is equally applicable to service systems because service systems are work systems, as explained earlier. A note at the bottom of Figure 1 notes that the one-page representation hides many attributes of each entity type. The metamodel's users would consider and apply hidden attributes while defining the problem or opportunity, evaluating the "as is" work system, and justifying proposed changes that would appear in the "to be" work system.

The metamodel reinterprets elements of the work system framework in a more detailed way. For example, information becomes informational entity, technology is divided into tools and automated agents, activities are performed by three types of actors, and so on. This latest version of the metamodel was designed to trace links from provider resources to value for customers, thereby addressing common issues in marketing and service science that are beyond the current scope. Representation decisions in the metamodel try to maximize understandability while revealing likely omissions from evaluation, analysis, or design processes. Starting at the top, the metamodel says the following:

- **Enterprises** and **value constellations** consist of **work systems**.
- A **work system** is treated as a provider work system, in contrast with a **customer work system** in which **value for customer** is realized.



Note: Many elements in the conceptual model have goals, attributes, performance indicators, and related principles, patterns, and generalizations that do not fit into a one page representation, and that must be included in more detailed explanations.

Figure 1. Work system metamodel (Alter, 2015)

- **Work systems** always contain at least one **work system activity** and may contain one or more **business processes** if some of the **work system activities** are sufficiently interrelated and sequential enough to be considered a process
- **Work system activities** use **resources** to produce one or more **product/services from activity** that may be used as **resources** for subsequent **work system activities** and/or may contribute to a **product/service offering** for customers. Thus, a particular **product/service from activity** produced by a **work system activity** may be invisible to customers. In addition, a particular **product/service offering** may combine a number of **product/services from activity** in a way that is relevant to a customer but may not be relevant to internal work system activities that customers do not perceive. Note: the term product/service is used to bypass debates about differences between products and services that are reflected in some of the definitions of service in Table 1 but are not important for the current purposes.
- **Customer work systems** create **value for customers** by using **product/service offerings** produced by the (provider) **work system**.
- **Resource**s used by a **work system activity** may include human resources **(participants**), **informational resources**, **technological resources**, and **other resources**, each of which have a number of specific types that are included in the metamodel to minimize the likelihood that they will be overlooked in an analysis.
- **Work system activities** are performed by **actor roles** that can be performed by three types of entities, **noncustomer participants**, **customer participants,** and **automated agents**. Automated agents are machines or software entities that perform tasks autonomously once launched. They are encapsulated functionalities (the third definition of service noted earlier). Automated agents often move to the foreground as work systems are decomposed during analysis and design. This is the central transition point between focusing on sociotechnical business service systems and service computing systems.
- The outcome of **work system activities** that use human resources (**participants**) depends on the **knowledge/expertise**, **skills/capabilities**, **performance metrics**, **motives**, and other characteristics of those **participants**.
- The **technological resources** used in a **work system activity** may include **tools** that are used directly by **participants** (e.g., a person driving a truck) or **automated agents** that perform work autonomously after being launched (e.g., a search engine).
- **Informational resources** used in a **work system activity** may include types of **informational entities** such as **transaction records**, **plans**, **forecasts**, **commitments**, **goals**, **rules**, **structures**, **documents**, **video**, **images**, **messages**, and even **conversations**.
- **Other resources** that may be used in a **work system activity** include **physical entities**, **time**, **resources from the environment** such as **organizational culture**, **laws**, **standards**, **regulations**, and **policies,** and **resources from shared infrastructure** that include **shared human resources, shared informational resources,** and **shared technical resources.** Thus, **shared technical resources** are viewed as separate from **technological entities** that are dedicated to the work system itself.
- Both the (provider) **work system** and **customer work system** may interact with **other work systems** in ways that may have positive and/or negative impacts on either work system. Interactions with **other work systems** may involve direct or indirect dependencies and intentional, unintentional, or totally accidental effects.

The analysis and design of a business service system focuses initially on visible, sociotechnical business processes and activities. Some service computing activities become visible when sociotechnical service systems are decomposed into subsystems. For example, the analysis of a medical diagnosis and treatment system starting with activities of medical personnel and patients may also look at automated agents that come to the foreground, such as software that suggests times for patient visits or identifies potential drug interactions. Further decomposition reveals more basic service computing activities such as data transfer, data retrieval, and data display. Thus, the metamodel provides a path for creating and analyzing situation-specific models that combine activities of human participants and activities of automated agents. The trend to automate previously manual tasks increases the significance of combining human and automated activities in the same models.

4. USDL as a Way to Represent Services as Encapulated Functionalities

This paper's goal is to build a bridge between business service systems and service computing. The previous section covered a business service system viewpoint expressed through the work system metamodel in Figure 1.

This section explains the Unified Service Description Language (USDL), which this paper uses to represent a service computing viewpoint. USDL is a recent development from the service computing community that builds on the service computing view that services are encapsulated functionalities. USDL was developed to describe services along the full continuum from purely human/professional services to totally automated services performed by computers (Oberle et al., 2013). The following overview of USDL is quite brief, but provides sufficient background for visualizing the benefits of a bridge between a business service system viewpoint and a service computing viewpoint.

4.1 Background on USDL

USDL was "developed across several research institutes and publicly funded projects across Europe and Australia ... as part of a standardization push." It was "built and evaluated in a collaborative and interdisciplinary way where more than a dozen researchers" brought expertise in computer science, security, service level agreements, business economics, and law. USDL was designed for applicability to a wide range of services such as "purely human/professional (e.g., project management and consultancy), transactional (e.g., purchase order requisition), informational (e.g., spatial and demography look-ups)," and so on. "Use cases from the corporate world provided insights into topics such as cost center ownership and provisioning, dependencies in complex business and IT landscapes," structuring service bundles, and the "need to extend beyond service providers to intermediaries and outsourced players such as brokers aggregators, and channel partners." (Oberle et al., 2013, p. 156)

The view of services in USDL is quite different from views of service in marketing, strategy, and operations management. Those literatures view services as some combination of the first two definitions mentioned earlier (basically, acts for others and creation of outcomes for others). The definitions in Table 1 also illustrate that well articulated viewpoints within those two definitions also call for particular embellishments such as necessarily involving coproduction, customization or responses to requests, value co-creation, or service as a form of economic exchange. In contrast, USDL views service in relation to the third definition, services as encapsulated functionalities. The range of such services mentioned in the previous paragraph was quite broad, going from purely human/professional services through totally automated services that are largely invisible to customers. Thus, while service computing is fundamentally about computing, USDL also covers non-computing situations in which functionality is encapsulated, such as the process outsourcing example in extension #3 above.

Nature of services. Fundamentally, services as described by USDL "constitute encapsulated and exposed functionality, drawing from core artifacts, e.g., those related to business processes, applications, objects, and resources. ... Whereas business process activities are said to be orchestrated across collaborating resources, service capabilities are delivered to consumers by providers. ... They provide functionality aimed at delivering value to consumers in terms of expected outcomes, subject to delivery constraints, e.g., availability, pricing, copyright or disclaimers. In doing so, they alleviate consumers with ownership of resources, costs or risks. .... Services involve active parts, for example, operations or actions, exposed to consumers, often referred to as capabilities." (p. 158)

4.2 Nine Modules of USDL

As described in Oberle et al. (2013, pp. 164-173), USDL contains nine modules, each of which will be mentioned very briefly to summarize each module's purpose and identify some of the concepts within each module, thereby providing a further indication of content that might not be obvious from the name of the module.

1) The **Service Module** establishes the essential structure of a service and links to the other eight modules, thereby encapsulating "functionality from prior instrumental artifacts on a business or technical level." For example, **ServiceBundle**, allows services to be grouped without any execution relationship; **CompositeService** combines services with an execution relationship such as ordering of steps, unordered steps, or data dependency. Other components include **ServiceVariant**, **NetworkProvisionedEntity**, **Resource**, and **Dependency**.

2) The **Participants Module** "captures the organizational actors that are important for the provisioning, delivery and consumption of a service" (p. 168). The participant **Role** covers service owners, service providers, stakeholders, intermediaries, and end consumers.

3) The **Functional Module** "allows the capture of service functionality at an abstract level, anywhere along the human to automation continuum. USDL "supports the capture of service functionality in different layers, for different levels of concern (white-box, gray-box and black-box)." A **Function** (or service **Capability**) may feature one or more input and output **Parameters**, as well as one or more **Faults** (related to exceptions). A function has preconditions and produces post-conditions (effects). Two types of resources are defined for a function, namely those used in performing (**utilizedResources**), e.g., tools or organizational roles, and those manipulated (**affectedResources**), e.g., business objects." (p. 168)

4) The **Interaction Module** captures "behavioral aspects of services concern[ing] how involved participants interact with the service." An **Interaction** "models an act of communication between the consumer of the service and one or more other participants that have responsibility in delivery." A **Phase** holds the sequence of **Interactions** and requires as preconditions, and yields as post-conditions, a set of **Milestones**. (p. 169).

5) The **Technical Module** "serves the semantic association between technical interface description and elements of USDL." (p. 170). It supports both operation- based and resource-based interfaces. It supports a link to "interface description artifacts" such as WSDL files.

6) The **Pricing Module** covers the charging for services "as mutually understood by those who own or deliver services and those who consume them." The hierarchical structure for service pricing includes **PricePlans**, **PriceComponents**, **PriceLevels**, **PriceAdjustments**, and other practical aspects of charging for services (p. 171).

7) The **Service Level Module** "provides the glue between abstractly specified service level issues in other USDL." It includes concepts such as **ServiceLevel**, **GuaranteedState**, **GuaranteedAction**, **ObligatedParty**, and **ServiceLevelProfile**. (p. 172)

8) The **Legal Module** "addresses the need for legal certainty in compliance and service networks and in trading services on marketplaces," covering issues such as liability, privacy, and copyright by using concepts such as **UsageRight** and **UsageType** (p. 172).

9) The **Foundation Module** "factorizes common parts of the remaining modules as a consistent continuation of modularization." All other modules depend on it as a reference for one or more of its elements such as **AbstractDescription** and **NaturalPerson** (p. 173).

5. Comparing Business Service Systems and Encapsulated Functionalities

The previous sections summarized the work system metamodel and USDL as representative examples of the ideas in business service systems and service computing. This section uses those ideas to compare business service systems with service systems that are encapsulated functionalities. The comparison is between "business service systems" and "service systems as encapsulated functionalities," rather than service computing per se. The comparison is stated that way because the work system metamodel can be used to describe encapsulated functionalities, just as USDL can be used to describe business service systems that have no human participants. Thus, the insights come from comparing ideas underlying the two representative examples rather than from the details of the specific examples. Table 2 summarizes the comparison, aspects of which will be explained further. The next section will use an example to show how the comparison plays out in practice.

Topic	**Business Service Systems**	**Service systems as encapsulated functionalities**
Default assumption	Business service systems are usually viewed as sociotechnical systems with human participants.	Service systems are totally automated and have no human participants.
Range of possible application	Business service systems can be totally automated because they can take the form of automated agents that are work systems on their own right, according to the metamodel. Thus, business service systems can be used for work planning and coordination applications.	USDL was designed specifically to permit encapsulated functionalities that have human participants. The only limitation is that the client does not participate in service activities and has no direct visibility of how the work is done. Thus, USDL supports systems applications such as service interfacing, cataloguing and match-making.
Degree of encapsulation	This may range from very low encapsulation to total encapsulation. The degree of encapsulation is smaller to the extent to which customers participate in work system activities and/or have visibility of how the activities are performed for them.	This view requires total encapsulation. Customers are not participants and have no visibility beyond any information passed to them by the server.
Treatment of coproduction and value co-creation	Allowed, and often assumed, but not required because a business service system may be totally automated. This touches on debates that are beyond this paper's scope concerning whether value is always co-created (Vargo and Lusch, 2008) or whether value co-creation is optional (Grönroos, 2011).	Not allowed due to the requirement of total encapsulation.

Customer	By default, a work system's customer is generally assumed to be a person, group of people, or organization. The customer for and automated agent (which the metamodel treats as work systems on its own right) could be a person or another automated entity.	As with business service systems, the customer for totally automated or only partially automated functionalities could be people or other encapsulated functionalities.
Pivotal artifact	"Product/service offering" represents the artifacts that are produced for customers (of the work system, who may be internal customers within a firm or external customers).	There are three types of pivotal artifacts: * Service interface exposing service functions * Message sent to the service system by the client to launch the service. * Responses returned from the service system to the client and/or outcomes produced by the service system.
Customer responsibilities	1) Customers participate directly in many business service systems. 2) Customers are responsible for cooperating with and not interfering with service providers 3) Customers are responsible for creating value for themselves	1) Customers must define and express service requests consistent with established formats and contracts. 2) Customers must maintain a means of receiving responses from the service system. 3) Customers are responsible for creating value for themselves.
Service interactions	Service interactions occur wherever customer participants and noncustomer participants play actor roles in the same work system activity. Some service systems rely heavily on service interactions and others have few service interactions	Service interactions occur only through messages passed between a customer (client) or a customer's work system and the service system that executes the desired work.
Customer experience	Customer experiences start during any co-production that occurs and extends to customer work systems that receive a provider work system's product/service offerings and use them to facilitate value for customers.	There is no customer experience of specific totally encapsulated services that are launched by other automated services. The customer experience for totally encapsulated services that are launched by human customers involves the initial contracting for the service, the specification of the request, and the use of the response. The customer experience cannot include involvement in the service system activities or visibility of how the activities are performed (other than any related reporting that is part of service system's pre-defined response).
Service level agreement	Having a formal service level agreement is optional. Many business service systems have informal commitments to exert best efforts.	Having a formal service level agreement is optional. Outsourcing arrangements usually have some type of service level agreement.
Subsystem traceability	Work systems can contain other work systems. The metamodel handles that using the entity type automated agent, which is a type of actor role for performing an activity. Automated agents are work systems on their own right.	Service systems provide interfaces of system components, which in principle can be contained in, or linked to, larger systems.

Table 2. Comparison of business service systems and service systems viewed as encapsulated functionalities

It is useful to add several points to the comparison in Table 2.

Human and non-human customers. Both business and computing views of service involve doing something for another entity. In business service systems the customer or client is usually a person, group of people, or organization, but especially in the decomposition of a larger service system there may be subsystems in which a person needs to respond to an automated agent. In those cases, the customer might be viewed as the automated agent that requested the response from the person. Conversely, in totally encapsulated service systems that might be defined by USDL, the client may be a person who requested something or may be another totally encapsulated service system that requested something. Thus, both views of services in Table 2 may have human and/or non-human customers.

Product/services. Use of the term product/service bypasses debates about distinctions between products and services that are tangential when analyzing operational systems. in business service systems, product/services are produced through work system activities that contribute directly or indirectly to the service system's product/service offerings for its customers. The same can be said about an encapsulated functionality since the response that it produces can be viewed as a product/service offering for its human or computerized customer.

Types of processes. A business service system may contain one or more business processes but must contain at least one activity. That distinction allows the metamodel to cover a full range of business process possibilities in service systems, including the following:

- largely unstructured creative processes (such as many design or artistic processes) that might use tools but have no pre-specified sequence and may involve extensive iteration.
- semistructured knowledge processes (such as medical diagnosis or legal analysis) that use tools and procedural knowledge but may involve situationally determined iterations.
- workflow processes (such as reimbursement processing) with a prescribed sequence but whose individual steps are treated as black box subroutines whose details are unknown.
- highly structured processes (such as pharmaceutical and semiconductor manufacturing) where conformity with both workflow sequence and the details of each step are essential.

The general assumption for service computing is that each service is defined rigorously in terms of its inputs, processing, and outputs, although the processing may be subcontracted to other services that presumably also are defined with similar rigor As noted in Table 2, that general assumption seems most natural in relation to totally computerized service systems, but also can apply to sociotechnical service systems that are totally encapsulated, such as when specific tasks are outsourced from one organization to another without any visibility for the customer about how the outsourced activities are performed.

Co-production and value co-creation. Some definitions of service in Table 1 imply that service necessarily involves co-production by providers and customers. The metamodel says that co-production occurs in any work system activity whose actor roles include customer participants and noncustomer participants. In relation to value co-creation, Vargo and Lusch (2008) says that value is always created in service. Grönroos (2011) says that value co-creation is optional. The metamodel says that customer work systems create value for customers, thereby clarifying that value co-creation occurs where activities in the customer's value creating work system coincide with work system activities within the provider's work system.

6. Example Illustrating the Metamodel as a Path to Service Computing

This section uses the "EU-Rent" example to illustrate how the metamodel can outline a model of a specific situation that includes both typical business service activities and totally automated service computing activities. This example summarizes the operation of a car rental company, including renting the car, picking up the car, dropping off the car, ending the rental, and accepting payment. OMG (the Object Management Group, an industry consortium that deals with enterprise integration, portability, and interoperability issues) used it to illustrate aspects of its products such as Semantics of Business Vocabulary and Business Rules (SBVR) (OMG, 2013). The nature of the EU-Rent scenario is apparent from the following excerpt: *"EU-Rent is a company that rents cars to persons, operating from geographically dispersed branches. The cars of EU-Rent are divided in car types (brands and models); for every car type there is a particular rental tariff per day. A car may be rented by a reservation in advance or by a 'walk-in' customer on the day of renting. A rental contract specifies the start and end dates of the rental, the cartype one wishes, the branch where the rental starts, ..."* (Op't Land & Dietz, 2012).

We approach this example by considering six levels of service description. The first two levels provide little detail but are useful beginnings of a basic understanding of a service system. Tables 3 and 4 illustrate how work system ideas support a richer understanding that focuses on business issues and largely shies away from technology and technical description. The fifth and sixth levels go into detail about how encapsulated services operate. Visualizing the six levels is useful in recognizing the transition point where business-oriented work system ideas begin to lose traction and service computing concepts necessarily take over.

Level 1: a phrase or sentence. The simplest way to describe a service is with a phrase that states what is being done for whom. Examples include teaching a class for MBA students, manufacturing a house for a family, and renting a car to a customer. In each case, the phrase is consistent with the first definition of service (acts for others), has implications related to the second definition (outcomes for others), but says nothing about encapsulated functionality. While this level might seem trivial, it proved useful to MBA and Executive MBA students by clarifying that the primary topic is a work system rather than the software it uses.

Level 2: a set of activities. Listing a set of activities provides a view of a service that says more than a level 1 phrase, but still provides too little information to support an analysis. Simple examples are the sections of Tables 3 and 4 that list activities. Graphical representations provide a richer way to represent activities, as is apparent from widespread use of flow charts, swim lane diagrams, and service blueprinting. WSM treats graphical techniques as optional when identifying a problem or opportunity, summarizing the "as is" work system, analyzing the situation, and recommending a proposed, "to be" work system. In some cases graphical representations are unnecessary. In others, they are extremely helpful.

Level 3: a work system snapshot. Table 3 summarizes the example using a tool from WSM called a work system snapshot (Alter, 2013, p. 86). Covering no more than one page, this type of summary is useful for clarifying the scope the work system or service system being analyzed or designed. Its goal is to help in clarifying a work system's scope by identifying the main participants, activities, product/services produced, customers, and important information and technology. While useful for summarizing the "as is" and "to be" work systems, this type of summary is still quite limited because it does not attempt to reveal important details such as which activities use specific information and what triggers the occurrence of each activity.

Customers	Products/ Services
• Renter • Driver	For customers: • Rental of car consistent with rental contract For providers: • Payment for rental

Major Processes and Activities
• **Renting agent** starts rental through interaction with **renter.** • **Driver** picks up the car. • **Driver** drops off the car. • **Drop-off agent** ends the rental. • **Renter** pays for rental.

Participants	Information	Technologies
• Renting agent • Renter • Driver • Drop-off agent	• Availability of cars at pick-up location • Rental contract (arrangement for payment, pick-up branch, drop-off branch, start date, end date, type of car, tariff, driver's driver license, arrangement for fuel in gas tank upon drop-off • Condition of car upon drop-off	• (not specified)

Table 3. Work system snapshot of EU-Rent scenario (Alter, 2014)

Activity	**Actor Roles**	**Information used**	**Information captured, created, updated, or deleted**	**Trigger**	**Pre-conditions**	**Business rules**	**Post-conditions**
Renting agent starts rental by interacting with **renter.**	• Renting agent • Renter	• Availability of cars • Credit card or other payment capability • Driver license of driver	• Rental contract	• Renter's request for rental	• Driver has valid driver license	• Rent only if the driver has a valid driver thus, it would be possible license.	• Car rented and available for driver's use
Driver picks up the car.	• Driver	• Rental contract	• Car picked up	• Car rented, available for driver's use	• Car rented available for driver's use	• Can leave location only if rental agreement exists.	Departure of driver from EU Rent pick-up location
Driver drops off the car.	• Driver	• Location of drop-off site		• Driver is ready to drop-off the car.	• Driver is ready to drop-off the car.	• Drop off the car at a branch of EU Rent, not elsewhere.	• Car returned to EU Rent.
Drop-off agent ends the rental.	• Drop-off agent	• Rental contract • Condition of car	• Drop-off date, time, place • Mileage driven • Car's condition	• Car dropped off	• Car dropped off • Valid rental contract	• Adjust charges based on rental contract.	• Rental terminated.
Renter pays for rental.	• Renter	• Rental contract • Return time, date, location • Car's condition	• Drop-off date and time • Car's condition	• End of the rental	• Valid rental contract • End of rental	• Renter pays based on tariff from rental contract.	• Fulfillment of renter's part of rental contract.

Table 4. Summary of the EU-Rent scenario using entity types from the metamodel

Level 4: a tabular summary based on the work system metamodel. Table 4 uses selected entity types in the metamodel to summarize the EU-Rent situation in more detail. It identifies familiar activities involved in renting a car. Actor roles appear in the second column. Information appears in two columns: information used, and information captured, created, updated, or deleted. Table 4 includes informational entities that are essential for integrating business service and service computing views of a specific service system, e.g., triggers, preconditions, business rules, and post-conditions

The metamodel can be used as the basis of many other tabular representations of different aspects of a work system, such as different types of information used by specific activities, or activities that use a particular informational entity or type of informational entity. The general form of Table 4 can also be used in hierarchical representations by decomposing a work system into subsystems. For example, each activity in Table 4 can be treated as a separate work system containing many smaller activities, each of which creates and uses certain information, has certain preconditions and triggers, and so on.

Level 5: encapsulated functionalities used by the work system. The usefulness of a level 4 work system description hits a limit when many of the activities are performed by automated agents that operate in network environments and may be selected dynamically based on conditions far removed from the work system's primary business logic. While automated agents in the metamodel are work systems on their own right, using the metamodel to represent such situations would be unnecessarily inconvenient because the metamodel is at the wrong level of generality. Many generic issues must be dealt with in a world of encapsulated functionalities that are discovered, selected, and executed through networks. This is where the metamodel should link to a service description language or other approach designed specifically to deal with the breadth and complexity of such situations, as will become apparent in the next several sections.

Level 6: services described as executable code. This last level is about programming methods and is beyond the current scope.

6.1 Extending the Example to Illustrate Links between Business Service Systems and Service Computing

The transition from the fourth level of service description to the fifth level is the point where a business service system perspective becomes difficult to use and necessarily links with a service computing perspective. This can be visualized through three fundamentally different extensions of the situation summarized in Table 4.

Extension #1: License checking software. Assume that the renting agent's interaction with the renter includes using "license checking" software that searches databases to check the driver license's validity. The renting agent launches a search process that may invoke many automated subprocesses and may cross many enterprise boundaries. The software is an automated agent (an encapsulated functionality) that operates autonomously once launched. It is triggered by a specified, formatted input from an actor role that has the right to use the software; it invokes a cascade of other software entities through pre-defined formats and contracts, ultimately producing a response for the renting agent.

Adding the step "check validity of driver license" to Table 4 will augment the original business service system description with an activity that relies totally and visibly on service computing. Since the automated agent is a work system, the format of Table 4 can be used to specify its operation as a set of activities performed by other automated agents. Such specifications are far from the interests or competence of typical business professionals. IT professionals should complete the specification, ideally using tools designed for that purpose.

Extension #2: Use of workflow software. Assume that EU-Rent decides to use BPM workflow software with an "enactment service" that "takes care of control and execution" (van der Aalst 2013, pp. 15, 17). The enactment service would initiate and track activities performed by human participants and by automated agents. Inclusion of the workflow software in the business service system could be represented by revising descriptions of activities performed by EU-Rent agents. Each activity would be initiated by the enactment service and then would be performed by the agent, after which the workflow software would control storage of data and status changes. The enactment service would be treated as a separate automated work system that operates continually, looking for conditions that require it to initiate action or record results. Thus, the enactment service would operate continually as a ubiquitous work system (within EU-Rent's rental operations), whereas the license checking software would operate only when initiated by a human agent.

Extension #3: Process outsourcing. Recognizing customer complaints about long lines at its office, EU-Rent hires an outsourcing firm called Rental-Services, Inc. (RSI) to perform skilled work previously done by rental agents at each site. When a customer arrives at a rental site, a low-skilled EU-Rent employee performs a one minute customer qualification step (What is your name? Do you have a reservation? Do you have a credit card?) and leads the customer to a video kiosk that enables rental interactions with an RSI agent at an RSI call center that can handle several hundred customers from different EU-Rent offices at the same time. The contract between EU-Rent and RSI specifies rental procedures in great detail.

In extension #3 RSI provides an encapsulated functionality (the third portrayal of service in Table 1) that receives a request from a customer at an EU-Rent site, performs required interactions with the customer, and returns a message back to an on-site EU-Rent employee about the resolution of the rental request, either identifying the car that has been rented or providing the reason why the rental request must be declined. From EU-Rent's viewpoint, the first step in Table 4 would expand into three steps: 1) EU-Rent agent performs initial customer qualification activity. 2) RSI creates the rental contract through interactions with the customer. 3) EU-Rent agent completes rental interaction by providing keys or providing a printed reason for declining the rental. The second of those three steps can be viewed as either a) a separate work system in which an RSI agent interacts with a customer at an EU-Rent facility or b) a service in which an RSI agent interacts with a customer at an EU-Rent facility. These views are almost identical on the surface but require extensive technical knowledge for completing the specification in either case.

6.2 Encapsulation of Functionality as the Point of Transition

The three extensions all highlight encapsulation of functionality as a point of transition between a business service system description and a description of a type of service that delivers results upon request while hiding its operational details from the business service system view.

- **License checking extension**. The agent enters a request that the license checking service answers. The service provides encapsulated functionality that is beyond the scope of a typical business professional's concern. A business professional wants to know that a correct answer is produced, but has little skill, knowledge or interest related to encapsulated functionalities that produce that answer.

- **Workflow extension**. The enactment service represents encapsulated functionality that operates in the background to initiate and track activities.
- **Process outsourcing**. The outsourcing vendor's employees perform rental services for customers at local offices. This service is an encapsulated functionality because it operates on request and returns one or more pre-specified types of responses.

In all three cases, the functionality is accessed through a network and might be executed anywhere. Those execution details are beyond the scope of typical business concerns, assuming that the functionality has been specified correctly, tested thoroughly, selected as preferable to other functionalities for the activity at hand, and provided by the enterprise itself or by a trusted supplier. In relation to describing or documenting the larger business service system, it makes sense to treat the encapsulated functionality as a black box, whereby activities within the business service system only need to access the encapsulated functionality, to provide information it needs, and to receive results it produces.

In three cases the encapsulated functionality might be described using the work system metamodel since all of the encapsulated services can be viewed as services systems (and hence work systems). The first two extensions involved totally automated service systems while the third extension was a totally encapsulated sociotechnical service system. In all of the cases, the functionality was encapsulated in a way that separated it from other functions in the EU-Rent work system and allowed it to be initiated on demand and executed elsewhere. An expanded version of all three examples could have included additional interactions between people at the rental site and the encapsulated functionality. That would only require that the encapsulated functionality would control subordinate functionalities that took care of specific tasks using information obtained through interaction with people at the rental site.

6.3 Could USDL Model the EU-Rent Example?

It would be possible to use USDL to model the EU-Rent example if the EU-Rent service system could be viewed as an encapsulated functionality. All three previously mentioned extensions of the EU Rent example were presented as services in this sense, i.e., as encapsulated functionalities that provide responses after being triggered by requests. The assumption that the entire EU-Rent example can be viewed this way is a bit less convincing because it was presented as a work system whose core, its business process, was revealed and elaborated instead of being treated as a black box functionality that executes upon request. On the other hand, the explanation of USDL in Oberle et al. (2013) included Road Transport and Ocean Export examples. Those examples might be represented as business processes, thereby implying that at least in principle, it would be possible to use USDL to model the EU-Rent example. If those examples could be modeled using the nine USDL modules mentioned above, it should be possible to model the EU-Rent example in a similar way.

Even if this application of USDL were possible, the desirability of modeling the EU-Rent example using USDL is questionable. Tables 3 and 4 demonstrated that is easy to apply WST and the work system metamodel for modeling the EU-Rent example, at least through the first four levels. On the other hand, just the brief description of the nine modules in USDL illustrates that modeling even those first four levels using USDL would require detailed knowledge of various concepts and modules in USDL. This could be attempted only by professional IT architects or software developers who had been trained on USDL or who were able to read technical manuals to learn it themselves. Using USDL would be far beyond the interest or capability of typical business professionals, many of whom would have little difficulty producing something like Tables 3 or 4 after a small amount of explanation. In addition, USDL was designed to support detailed descriptions of interactions, technical decisions, pricing, and legal issues that are need to be documented at some point, but that are beyond the scope of descriptions that are used for obtaining a basic understanding of a service system.

7. Implications for Establishing a Bridge between a Business Service System Viewpoint and a Service Computing Viewpoint

This paper's previous sections used WST (and a work system metamodel) as a proxy for a business service system viewpoint and USDL as a proxy for the service computing viewpoint. This section uses those proxies to explore implications for addressing the question in the Dual Call for Papers and establishing a bridge between the two viewpoints.

7.1 Partial Overlap of WST and USDL

Concepts and terminology in WST and USDL overlap to some extent, but their purposes diverge. WST's primary purpose is to support understanding and modeling of sociotechnical work systems and service systems, while USDL was designed to support a business-to-computational view of an encapsulated functionality that it calls a service. WST is more comprehensive since it covers both services in the USDL sense and other business functionality. USDL focuses only on detailed description of services. Also, USDL is designed to articulate technical implementation considerations, whereas WST reflects a business, management or user perspective and treats technical implementation as beyond its scope

Value generating activity. While an exhaustive comparison of the work system metamodel and USDL is beyond this paper's scope, each embraces a pivotal concept related to value-generating activity that can be used to accentuate commonalities and differences between the approaches. For the work system metamodel, activities within the provider work system produce product/services that contribute directly or indirectly to product/service offerings for customers. When performed by human participants rather than automated agents, those activities generate outcomes that depend on knowledge/expertise, skills/capabilities, performance metrics and motives. Activities use various types of informational, technological, human, and other resources that are identified in the metamodel. These different concepts, directly or indirectly related to a work system activity, demonstrate the richness of business service systems phenomena that the work system metamodel supports.

In USDL, services are containers for value-generating activities. They capture relations across services (prescriptive relations or compositional structures or descriptive relations or dependency constraints) among other broader associations (e.g. pricing policy). Services fundamentally provide capabilities, which are abstractions of computational operations, having inputs and outputs with data elements of arbitrary nesting, faults, preconditions and postconditions. Capabilities manipulate computational resources such as business objects in application systems or utilize resources such as organizational roles or tools. Capabilities can be exposed through technical interfaces and can be used to support interactions with consumers (customers). Collectively, these are concepts relevant to the service computing.

Thus, WST and USDL overlap on their respective concepts of value-generating activities, i.e., work system activities for WST and service capabilities for USDL. An additional overlap across the two metamodels is the resources that are used when human participants or automated agents perform activities.

7.2 Practicalities

As demonstrated through earlier examples, use of the work system metamodel hits a limit when many of the activities are performed by encapsulated functionalities that can be described in great depth using USDL. It is possible to describe encapsulated functionalities using the work system metamodel, but USDL is a much better approach because it is designed specifically to handle that type of situation.

The metamodel was designed to cover typical business systems and to organize ideas that are easily understood by business practitioners. It was not designed to handle topics and issues that are essential when dealing with encapsulated functionalities, such as: operational sufficiency of functions available through a service (input/output document messages, pre/post condition rules expressed against data elements and references for exception handling); composition and artifactual (resource) dependencies of multiple functionalities; interaction protocols for consumer access to functions; pricing of service capability use subject to automated pricing models; legal aspects of accessing and operating functionalities, and so on. A great deal of research has gone into approaches for dealing with these issues.

At least in principle USDL could be used for modeling work systems like the EU-Rent example even though the complexity and rigor of USDL is relevant mainly to software developers. Based on cognitive load theory (Sweller, 1994), a high level of formality may be counterproductive for business professionals trying to understand work systems at the first four levels of description. CLT says that intrinsic cognitive load is related to the inherent nature of the material, whereas extraneous cognitive load is related to how the material is presented. The type of WST-based representation in the first four levels has very low extraneous cognitive load because it is based on familiar ideas and does not require use of overly precise concepts that are difficult for most people to understand.

8. Approaches for Moving Forward

Based on the foregoing observations, we see four possible approaches for bridging the two viewpoints whose characteristics are summarized in Table 5. The first of the four approaches (complementarity) is based directly on the example presented earlier. The second approach (WST front end to USDL) probably has the greatest potential. The other two approaches are mentioned for completeness but do not seem as likely to lead to significant progress.

	WST and work system metamodel	**USDL**
Usability by business professionals	High	Low
Precision and rigor	Low - moderate	High
Focus on general business structure and performance issues	High	Low
Focus on service-specific topics such as pricing, legal, and service level agreements	Low	High
Applicability for internally directed and externally directed systems	High	High, even though designed for externally directed services
Cognitive load	Relatively low	Much higher

Table 5. Comparison of WST approach and USDL approach

Complementarity. With this approach, the business service system viewpoint expressed by WST or a similar set of concepts is used through the first four layers, thereby providing clarity about the nature, scope, and general operation of the business service system. At that point, technical experts use business process modeling tools such as BPMN for defining business logic in detail and USDL or something similar for specifying details of encapsulated functionalities that are invoked by specific process steps. As implied by Table 5, the general logic of this approach is to avoid pretending that one approach solves all problems, and instead to mix tools and methods in ways that address different issues effectively and do not try to force one approach on all topics and issues. This approach requires conscious separation between using WST and the work system metamodel versus using USDL. Nothing prevents iteration, however because it is always possible to improve the work system model and then update the USDL models.

WST front end to USDL. Business services can be described in a way that allows describing their functional aspects through WST and their non-functional aspects such as pricing, legal and technical infrastructure in USDL. With this approach, the functional description of business services would not be forced into an encapsulated approach that is more suitable for technical services. Business service activities and interactions would be described through WST activities, while strict interaction protocols (document exchange sequences which are important for certain applications e.g. B2B domains like transportation management) could be described using an encapsulated view of the service. Thus, USDL would play a purely cataloguing purpose for non-functional and basic functional aspects. Technical services would be described through USDL and traceable to a work systems context captured in a metamodel based on WST. The description of services would become more harmonious across WST and USDL, with the USDL part providing strictly encapsulated services that are aligned to a work systems context.

The process outsourcing example mentioned earlier as extension #3 is a relevant example. Without something like a WST-based model, it is likely that a process outsourcing model based totally on USDL would omit important issues. For example, using USDL would lead technical experts to focus on the encapsulation of functionality, whereas business professionals probably would be concerned about having proper visibility about how the outsourced work was being done, especially if they view outsourcing as a way to improve business performance rather than a way to "export a mess." Thinking of the outsourced work as part of a larger business service system (i.e., not an encapsulated functionality) would shine more attention on the customer's responsibility in making sure that the work actually was done well by the outsourcing provider.

WST-based model of USDL. Since the work system metamodel treats automated agents as encapsulated functionalities, at least in principle it is possible to create work system models of all of the modules within USDL. The resulting work system descriptions would view the nine components of USDL as separate work systems that could be incorporated into or parameterized for a particular WST-based model of a business situation. Proceeding in this direction would basically be an exploratory research project to see how far the metamodel could be extended. Notice that this would be a process-oriented model, not an UML model.

USDL refinement of a work system model. USDL was designed to incorporate both totally automated and sociotechnical service systems that can be encapsulated. The possibility of modeling sociotechnical systems implies that USDL might be used to model some of the types of sociotechnical work systems that WST and the work system metamodel were designed to model. The qualification “some of the types” reflects the limitation that encapsulation is not possible in many sociotechnical service systems in which customers have significant responsibilities, or co-produce the outcome or where there are aspirations of "value co-creation." The latter situations are important focal point in the service discourse in general management.

9. Conclusion

This paper’s goal was to use a work system metamodel and USDL to build a bridge between business service systems and service computing systems. That would be a step toward the type of transdisciplinary research suggested by the Dual Call for Papers from *INFORMS Service Science* and *IEEE Transactions on Service Computing*. This paper started by identifying three portrayals of service. It treated business service systems as work systems, implying the relevance of a work system metamodel that it used as the basis of a path toward combining business service activities and service computing within a single model of a service system. It identified six levels for describing a service system and explained why a work system approach was more appropriate for business-oriented description and analysis up to the fourth level. USDL provides a much more appropriate basis for the fifth and six levels in situations where it is important to describe and analyze encapsulated functionalities that operate through networks.

A fundamental distinction related to views of service. The work system metamodel and USDL cover some of the same conceptual territory and overlap in various ways, but there is a key distinction based on different fundamental views of what service is about. The work system metamodel is based implicitly on the first definition of service that was mentioned at the outset, *an act performed to produce outcomes for the benefit of others.* With that implicit definition, the work system metamodel can accommodate the other views of service, i.e., services as outcomes and services as functional entities such as web services. The metamodel expresses the outcome of activities as a “product/service offering” because that is the outcome that a customer expects, receives, and experiences. Any clearly bounded work system also can be viewed as an encapsulated functionality that produces particular product/services for customers. However, the fact that customers may be work system participants makes it more difficult to assure any particular outcome due to customer-related factors and various exogeneous factors, both of which are beyond a provider’s control.

It is possible that practicalities related to nature and spirit will impose fundamental limits on reconciling sociotechnical service systems and service computing systems. Service computing systems are totally automated. The components were created by people but do not exhibit human agency, human variability, and human frailties when executing pre-defined activities. Sociotechnical systems are quite different. The four types of business processes mentioned in the comparison of the business service view and encapsulated functionalities view are a reminder that many activities with human participants are inherently creative or knowledge-intensive and do not call for a high degree of pre-defined, tightly controlled structure. In addition, research related to adaptations, workarounds, and emergent change all start from real world observations of obstacle- or insight-related non-conformance or deviations from existing patterns of activity.

Need for interfaces between business and technical views. Difficulties in communication between business and technical professionals have been a long-standing problem that has been discussed for decades under a variety of headings ranging from user participation and project risk factors to digital divides and business/IT alignment. This paper's discussion of the six levels for describing a service system and of the transition between specifications that need more of a business orientation versus those that need more of a technical orientation could lead to better tools and methods.

Despite those practical issues, the effort to articulate areas of greater integration between business service systems and service computing systems could yield substantial benefits. Many existing business service systems probably would perform more efficiently and effectively if they could incorporate more of the spirit of service computing. The attempt to reconcile business service systems and service computing systems could yield important benefits for sociotechnical service systems by providing better integration of human creativity and judgment with machine stability and repeatability.

References

Alter S (2010). Viewing Systems as Services: A Fresh Approach in the IS Field, Communications of the Association for Information Systems. 26(11): 195-224.

Alter S (2012a). Challenges for Service Science, Journal of Information Technology Theory and Application. 13(2): 22 -37.

Alter S (2012b) Metamodel for Service Analysis and Design Based on an Operational View of Service and Service Systems, Service Science. 4(3): 218-235.

Alter S (2013). Work System Theory: Overview of Core Concepts, Extensions, and Challenges for the Future, Journal of the Association for Information Systems. 14 (2): 72-121.

Alter S (2014) Potentially Valuable Overlaps between Work System Theory, DEMO, and Enterprise Engineering, IEEE Conference on Business Informatics, Geneva.

Alter S (2015) Work System Theory as a Platform: Response to a Research Perspective by Niederman and March, Journal of the Association for Information Systems, in press.

Brown AW, Delbaere M, Eeles P, Johnston S, Weaver R (2005). Realizing service-oriented solutions with the IBM rational software development platform, IBM Systems Journal. 44(4): 727–752.

Cherbakov L, Galambos G, Harishankar R, Kalyana S, Rackham G (2005). Impact of service orientation at the business level, IBM Systems Journal. 44(4): 653–668.

Clerc V, Niessink F (2004). IT Service CMM: A Pocket Guide, van Haren Publishing.

Fitzsimmons, JA, Fitzsimmons MJ (2006). Service Management, 5th ed. New York, NY: McGraw-Hill

Grönroos C (2011). Value creation in service logic: A critical analysis, Marketing Theory, 11(3).: 279-301.

Hill TP (1977). On goods and services, The Review of Income and Wealth. 23: 315-338.

ITIL (2011). ITIL Glossary and abbreviations: English, available from http://www.itil-officialsite.com/InternationalActivities/ITILGlossaries_2.aspx , viewed on Oct. 25, 2014.

Kotler P, Keller K (2006). Marketing Management, 12th ed., Upper Saddle River, NJ: Prentice Hall.

Oberle D, Barros A, Kylau U, Heinzl S (2013). A unified description language for human to automated services. Information systems. 38(1): 155-181

.OMG (2013). Semantics of Business Vocabulary and Business Rules (SBVR), v1.2. http://www.omg.org/spec/SBVR/1.2/ viewed on Oct. 25, 2014.

Op't Land Ml Dietz JL (2012). Benefits of enterprise ontology in governing complex enterprise transformations,. In Advances in Enterprise Engineering VI (pp. 77-92). Springer Berlin Heidelberg.

Pine BJl Gilmore JH (1999). The Experience Economy: Work Is Theater and Every Business a Stage, Cambridge: Harvard Business School Press,

Sampson SE, Froehle CM (2006). Foundations and Implications of a Proposed Unified Services Theory, Production and Operations Management. 15(2): 329-343.

Sweller J (1994). Cognitive load theory, learning difficulty, and instructional design, Learning and instruction. 4(4): 295-312

Truex D. Alter S. Long C (2010) Systems Analysis for Everyone Else: Empowering Business Professionals through a Systems Analysis Method that Fits Their Needs, Proceedings of ECIS.

van der Aalst WMP (2013) Business Process Management: A Comprehensive Survey. ISRN Software Engineering 2013:1-37

Vargo SL. Lusch RF (2008). Service-dominant logic: continuing the evolution, Journal of the Academy of Marketing Science. 36: 1-10.