The Business Service Representation Language: A Preliminary Report (Pre-Publication Draft)[⋆]

A. K. Ghose, L.S. Lê, K. Hoesch-Klohe, E. Morrison

Decision Systems Lab
School of Computer Science and Software Engineering
University of Wollongong
NSW 2522, Australia
{aditya, lle, khk789, edm92} @ uow.edu.au

1 Introduction

The recent interest in services science has led to research in *business service management*, using a multi-disciplinary synthesis of thinking from the disciplines of computing, management, marketing and potentially several others. Business services can be of many different kinds. The notion includes within in its ambit business process outsourcing services, clinical services, customer contact services as well as IT-enabled services, to name a few representative examples. A key component of any business service management framework is a service modeling language.

The development of such a language poses several challenges. Unlike web service modeling, business service modeling requires that we view human activity and human-mediated functionality through the lens of computing and systems engineering (and building a framework that is general enough to include both notions of services within its ambit). This clearly requires an enhanced set of modeling constructs that go beyond those that have been used for web service modeling. This also requires a modeling notation at higher level of abstraction - one that supports the description of complex business functionality using abstract modeling constructs that offer a natural fit with concepts used to describe these services in everyday discourse. In the course of our research, we have found a close correlation between the notions of *services* and *contracts* (although the two notions are by no means identical). Our study of real-life business service descriptions, in domains as diverse as government services, IT services and consulting services, suggests that some contractual concerns appear routinely in service descriptions, and are part of the discourse on service design and redesign. Our survey of existing service modeling frameworks also suggests that while these are interesting and worthwhile, none come with the complete set of required features. The development of the Business Service Representation Language (BSRL) described in this paper was motivated by these concerns.

[⋆] This work has been funded by the Australian Cooperative Research Centre for Smart Services http://www.smartservicescrc.com.au/

BSRL was developed as part of a project to develop a framework and supporting toolkit for strategic service alignment. The Strategy Modeling Language (SML) developed in this project provides the value modeling component to BSRL service models (discussed in detail later in the paper).

BSRL has since been used to:

- Model rural services in emerging economies, as part of a framework for service redesign to improve the quality of rural service delivery.
- Model government services in large state government agencies.
- Model internal (i.e., not customer-facing) services in large corporate settings.

2 The BSRL Meta-Model

A service model in BSRL consists of the following components:

Service ID
Preconditions
Post-conditions
Inputs
Outputs
Goals: These are the *intended effects/postconditions* of a service (note that not all postconditions are intended - e.g., it might not be my goal to debit my account with a certain amount of money, but that might be the effect of seeking to achieve the goal of purchasing an airline ticket using an online ticket booking service).
Assumptions: These are conditions on whose validity the execution of a service is contingent, but whose validity might not be verifiable when a service is invoked or during its execution. Assumptions need to be monitored during the execution of a service - if a violation is detected (once the truth status of the condition is revealed), the service may have to be aborted. Assumptions are common in informal service descriptions, but might not be identified as such. In our work modeling business services offered by government agencies, we have found references (in textual service descriptions) to lists of "'client responsibilities"'. These are statements of what service clients are responsible for doing, in order to enable the provider to fulfill the relevant service. These are clearly not pre-conditions, since they cannot be evaluated when a service is invoked. Indeed, checking to ensure that a client has fulfilled all client responsibilities is impractical in general. Instead, one can use the non-fulfillment of these responsibilities as a trigger for aborting the execution of a service, or for abrogating the contractual obligations of the service provider. "Force Majeure" clauses in contracts are also examples of assumptions (i.e., the provider commits to delivering a service provided no natural disaster intervenes etc.)
QoS specifications: Quality-of-Service (QoS) factors are described as a set of $\langle$QoS-factor, range$\rangle$ pairs, where the range provides the upper and lower bounds of QoS factors with quantitative evaluations (note that upper and lower bounds might be equal), or is a qualitative value.
Delivery schedules: These are specified as a set of $\langle$functionality, deadline$\rangle$

pairs. Arguably, a delivery schedule is part of a QoS specification, but these require special handling during service decomposition - hence the special status accorded to them.

Payment schedules: These are represented in a manner similar to delivery schedules. These are not ontologically part of a service QoS specification, but are arguably part of the set of assumptions. Like delivery schedules, these require special handling during service decomposition - hence the special status accorded to them.

Penalties: These are specified as a set of $\langle$ condition, amount $\rangle$ pairs, such that *amount* is the penalty paid if *condition* becomes true. Arguably, penalties are part of the QoS specification, but these require special handling.

Value model: For each stakeholder in the service, a distinct value model is included in the service description. A value model represents how a service delivers value to a given stakeholder. A value model can serve as the basis for service design, and for re-design in the face of change (where the impact on value models of alternative re-designs provides the basis for deliberation on how best to implement change).

Resource model: The ability to understand how a service needs to be provisioned is a critical component of service design. This understanding also underpins any attempt at service optmization. A resource model describes available resources in a manner as expressive as a UML class diagram, with the usual part-whole and generalization-specialization relationships. In addition, a special *uses* relationship is required to describe how a given resource might use another. In general, a set of BSRL service models might share a common *resource ontology* - the resource model for a service is then a reference to a set of resource classes/instances in this ontology. These might be at different levels of abstraction. For instance, a service might be described as using a "printer" resource, or more specifically an "inkjet printer" resource or even more specifically an "HP inkjet printer resource". Note that the notion of a resource is general, and might include in its ambit people, tools, energy, other operating inputs and so on.

We note that not all service models will populate every component of the template described above. We do not commit to a specific language for representing pre- and post-conditions, goals and assumptions. These could be described informally in natural language, or formally (such as in temporal logic, as used in goal-oriented requirements engineering [11] [9]). Our current tool support for BSRL (not described in this paper due to space restrictions) offers a controlled natural language interface [10] for specifying these, and then uses ontology-mediated techniques to obtain formal representations. Figure 1 provides the diagrammatic meta-model.

3 The Service Value Model

A value model, as mentioned earlier, is a critical component of a service model. It provides the basis for service design (in very much the same way that a requirements model provides the basis for system design - arguably, requirements

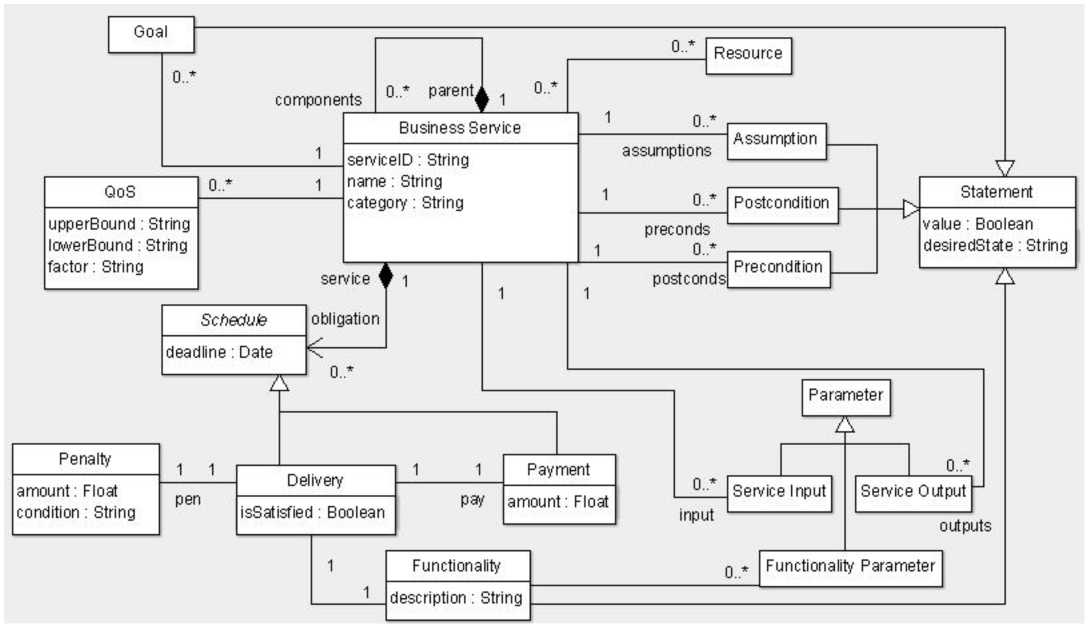

Fig. 1. BSRL meta-model

modeling is a special kind of value modeling). Value models support service re-design, with alternative re-deisgn assessed in terms of their impact on the value model. Traditionally, value modeling has been considered in economics and decision theory, with *utilities* being used as value measures. Utilities are inadequate for our purposes for many well-understood reasons (which we shall not elaborate here for brevity) including the difficulties associated with obtaining numeric measures of the utility to a given stakeholder from a service. More recently, Gordijn et al have proposed the e^3 Value framework [3] which provides conceptual modeling constructs to describe how actors exchange *value objects*. We believe that the notion of value objects can be generalized, and that ultimately a service or system delivers value to a stakeholder by helping achieve the goals or objectives of the stakeholder. In an enterprise setting, *enterprise strategies* provide the goals/objectives of the enterprise stakeholder. We have developed the *Strategy Modeling Language (SML)* to provide a formal basis for representing enterprise strategy, and have evaluated its expressive adequacy over a range of real-life organizational strategy documents (SML turns out be adequate). SML provides a useful value modeling framework and is outlined below.

An SML *strategy model* is a set of *strategy statements* of the following three kinds:

– A *goal*: Goals are descriptions of conditions that an organization seeks to achieve. Goals admit boolean evaluation, i.e., an organization is able to clearly determine whether it has achieved a goal or not (consider the following example: *"Our corporate strategy is to be the market leader in mobile handsets"'*). This notion of a goal closely corresponds to that used in goal-oriented requirements engineering.

– An *objective function*: An objective function is a construct used in operations research techniques to define what the *preferred* or *optimal* solution to an optimization problem might be. These are typically articulated as *maximize f* or *minimize f*, where f is a function defined on the *decision variables* (using which the constraints that *feasible solutions* are required to satisfy are also written). Our analysis of a large number of actual corporate strategy documents, as well as the management literature, suggests that startegies involving *corporate performance measures* or *key performance indicators (KPIs)* are articulated in the form of maximization or minimzation objectives. Consider the following statements of strategy: *Our strategy is to minimize order lead times, or, "Our strategy is to maximize customer satisfaction"'.* In the first of these, the intent is to minimize a function encoding order lead time while in the second, a funcion encoding some definition of customer satisfaction (for instance, using average customer wait times at the customer contact centre, the number of escalations, the number of product returns etc.) is maximized.
– A *plan*: A plan is a set of goals together with a set of sequencing and related coordination constraints. In the most general sense, a plan can be as complex as a process model. In this paper, we will view plans only as linear sequences of goals. This is because SML is designed to be used by senior management, i.e., individuals within an organization who might be involved in strategy formulation. Also, our analysis of a large number of actual corporate strategy documents suggests that strategies are typically articulated at a very high level of abstraction, where control structures more complex than linear sequencing is never required. A typical example is the following anonymized but actual strategy statement: *Our strategy is to first gain market acceptance in NZ, then position ourselves in the UK market, then use the UK market credibility to enter the Australian market.* There are three steps (goals) in this strategy, connected via a linear sequencing relation.

In the following, we will use the terms *strategy model* and *value model* interchangeably.

Value models underpin the analysis required for *service evolution*. We shall use the notion of service evolution to denote situations in which service models need to be modified. Drivers for such modifications might be:

– Service re-purposing, necessitated by altered requirements/goals/strategies that the service was designed to realize.
– Service improvement, i.e., improving the performance of the service relative to one or more QoS factors
– Operational drivers, such as changes to service delivery platforms.
– Compliance, i.e., service re-design triggered by a finding of non-compliance.

We are interested in two different kinds of analysis to support service evolution: impact analysis and trade-off analysis. In impact analysis, we aim to understand the impact of a proposed change on the value model of a service. For instance, which strategies in a given stakeholder's value model will become unrealized as a consequence of the change? Which stakeholders' value models will be impacted by the proposed change? In trade-off analysis, we seek to identify the best amongst alternative service designs (in terms of their impact on the value model) implementing the required change.

The impact of a candidate change to service model can be represented by $(\bigoplus_{i=1\ldots n} V_i - V_i')$, where V_i is the value model for stakeholder i associated with the original service design, V_i' is the value model for stakeholder i associated with the changed service design, $-$ is an abstract difference operator (thus $V_i - V_i'$ represents elements of V_i that are not in V_i'), and $\bigoplus$ is an abstract combination operator (thus, if the value models were represented as sets, then this would be the set union operator). These abstract notions can be further elaborated in a formal fashion to obtain an algebraic generalization of a set of useful instances, but this is not presented here due to space constraints. Note that this captures one of possibly many intuitions on how to assess the value impact of a change - others (such as ones that accord importance to the new value proposition in V_i') could also be of interest.

A given change constraint (e.g., re-purpose the service in a given manner, make a service design compliant, or improve a QoS factor to meet a given threshold) can be implemented in multiple different ways. Trade-off analysis will be required to identify which of these alternative realizations of the change request we would choose to adopt. One way this could be done is to seek the alternative which minimizes impact (either with respect to set inclusion or with respect to set cardinality).

4 Related Work

Web Service Description Language (WSDL) [2] provides an implementable technical language that describes how a service can be accessed and invoked over the web. Semantic web services (WSMO, OWL-S, WSDL-S etc.) [6] allow for semantic markup of service description, and support pre- and post-conditions. SAWSDL [4] is another such language. In [7], O'Sullivan has examined both the functional non-functional properties of business services. Most of these features have been incorporated in the Universal Service Description Language (USDL) [1] which aims to provide a general framework for generic services (i.e. either technical or business). This language features three service perspectives namely business, operational, and technical.

Table 1 compares existing service modeling frameworks with BSRL in terms of the modeling constructs of interest.

The following observations supplement the comparison in Table 1:

- WSDL provides support for specifying low-level technical details that are not of interest in business service modeling.

	WSDL	WSDL-S	SAWSDL	OWL-S	USDL	BSRL	WSMO
Input/Ouput	✓	✓	✓	✓	✓	✓	✓
Pre-Condition	×	✓	×	✓	✓	✓	✓
Post-Condition	×	✓	×	✓	✓	✓	✓
Goals	×	×	×	×	×	✓	✓×
Value Model	×	×	×	×	×	✓	×
Assumptions	×	×	×	×	×	✓	✓×
QoS Specifications	×	×	×	×	×	✓	?
Delivery Schedule	×	×	×	×	×	✓	×
Payment and Penalties	×	×	×	×	×	✓	×

Table 1. Comparison of Service Description Languages

– Web Service Modeling Ontology (WSMO)[1] provides for assumptions but our notion of assumption differs in several ways. WSMO assumptions are only meant to be checked at service invocation time, while BSRL assumptions are constantly monitored. BRSL assumptions are intended to also model the contractual obligations of parties outside of the service provider. As well, BSRL assumptions play a role in service decomposition.
– BSRL does not make explicit provision for ontologies but these could clearly be used in conjunction with BSRL to provide semantic markup of *all* constructs.
– BSRL does not provide support for process modeling as OWL-S does. This is a deliberate choice to provide a sufficiently abstract notation for business service modeling.

Pijpers et. al. have presented a framework and methodology for modeling business services called the e^3 Family [8] that examines three perspectives of an organization one of which is focused on business service modeling. The use of separation of concerns in [8] aids in clarifying discussions by relevant stakeholders. The i* agent-oriented conceptual modeling language [11] has also been used for service modeling. Neither of these approaches provide support for the contractual aspects of BSRL. i* supports QoS modeling only in very general terms, as diagrammatic *softgoals*, related to other i* constructs via positive and negative *contribution links*. Neither i* nor the e3value framework support assumptions, penalties or schedules. i* is arguably better suited as value modeling framework and has been proposed as a diagrammatic front-end to the Strategy Modeling Language [5].

5 Conclusion

BSRL has been evaluated by modeling a range of government, IT and consulting services, and has been found to be adequate. The most detailed evaluation was carried out with government services, where a substantial service catalogue of

[1] WSMO homepage http://www.wsmo.org

a state government agency was modeled in BSRL. Our experience also suggests that BSRL can provide guidance to analysts in terms of what needs to be included in service descriptions - without such guidance, some service descriptions end up incomplete and inadequately structured.

BSRL deliberately ommits any support for process modeling. This is based both on observation of actual business service documentation practice and on the obligation to offer a sufficiently abstract set of modeling constructs. However, it is easy to see how BSRL service descriptions can be refined into process descriptions with the goals, post-conditions and QoS specifications providing guidance to the process designer.

This preliminary report ommits considerable detail. It also omits a detailed framework for service decomposition and service-contract interplay. These, plus detailed experience reports, will be made available in the full version of the paper.

References

1. Cardoso, J., Winkler, M., Voigt, K.: A service description language for the internet of services. In: Proceedings of First International Symposium on Services Science. pp. 1–10. No. 1, Berlin, Germany (March 2009)
2. Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S.: Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. Internet Computing, IEEE 6(2), 86–93 (2002)
3. Gordijn, J., Yu, E., Raadt, B.: E-service design using i^* and e^3 value modeling. IEEE Software 23(3), 26–33 (2006)
4. Kopecky, J., Vitvar, T., Bournez, C., Farrell, J.: Sawsdl: Semantic annotations for wsdl and xml schema. IEEE Internet Computing 11, 60–67 (2007)
5. L.-S. Le, B.Z., Ghose, A.K.: Representation of strategy using an i*-like notation. In: Proc. of the 4th International i* Workshop held in conjunction with CAISE-2010. pp. 113–117 (June 2010)
6. Lara, R., Roman, D., Polleres, A., Fensel, D.: A conceptual comparison of wsmo and owl-s. In: (LJ) Zhang, L.J., Jeckle, M. (eds.) Web Services, Lecture Notes in Computer Science, vol. 3250, pp. 254–269. Springer Berlin / Heidelberg (2004)
7. O'Sullivan, J.J.: Towards a precise understanding of service properties. Ph.D. thesis, Queensland University of Technology (2006)
8. Pijpers, V., Gordijn, J., Akkermans, H.: Business strategy-IT alignment in a multi-actor setting: a mobile e-service case. In: Proceedings of the 10^{th} International Conference on Electronic Commerce. ACM (2008)
9. Regev, G., Wegmann, A.: Defining Early IT System Requirements with Regulation Principles: The Lightswitch Approach. In: 12^{th} IEEE International Conference on Requirements Engineering. pp. 144–153. Los Alamitos, CA, USA (2004)
10. Schwitter, R.: Controlled natural languages for knowledge representation. In: Proceedings of 23^{rd} International Conference on Computational Linguistics. pp. 1113–1121. Beijing, China (2010)
11. Yu, E.: Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of RE-97 - 3^{rd} Int. Symp. on Requirements Engineering. pp. 226–235 (1997)