Sub-Game Perfection in Semantic Web Services

Stephen Kinsella, Juan M. Gomez, Christoph Bussler, and Dieter Fensel

Digital Enterprise Research Institute,
National University of Ireland,
Galway
{stephen.kinsella, juan.gomez, chris.bussler, dieter.fensel} @deri.org

Abstract. We describe the dependence of web services on the pre and post conditions that trigger their use in on-line B2B and B2C transactions. We build a model that highlights these dependencies probabilistically and illustrate the use of the model with an example. We place this model in the context of the development of the nascent Semantic Web. Our result is that, once many web services are chained together, this dependence on different and increasingly complex triggers will produce a decay in the number of successful uses of the web service. This result lays a general theoretical foundation for a game-theoretic examination of web service interaction. We conclude the paper with a program for further work in this area.

1 Introduction

Web Services are usually modeled using process flow descriptions. In this paper we model a rudimentary web service as a series of games, a series of sub games and a series of equilibria. We recognize from the outset that the concept of a subgame-perfect equilibrium is an inherently incomputable one and we are at pains to make the distinction between a backwardly induced equilibrium such as we might find in a game of complete information, and the equilibrium be define inside a game of *incomplete* information. In this paper, we introduce the concept of sub game perfection, and then model a simple web service, buying a book, as depending on pre- and post- conditions, described below. We then generalize this example using the tools of game theory. We take the viewpoint that in order to understand the effectiveness of web services in general, it is necessary to understand the dependency of the web service on its various pre- and post- conditions, whatever they may be. Our lever is the introduction of the concept of subgame perfect Nash equilibria to the literature on web service modeling. To our knowledge, this has not been attempted before.

1.1 What are Web Services?

Complexity of integrating heterogeneous software components can be achieved by means of web services. They provide standard protocols for discovering, invoking, describing and composing services. Current web service technology, based on SOAP, WSDL and UDDI, has a very basic and non-automated interaction model. A web service provider publishes information in an UDDI registry and offers a programmatic access that potential customers can bind to. For this, interaction via XML based messages on a transport protocol with a web service provider provides the desired functionality. The dynamic binding to the consumed Web services is done using the information from the web service registry. In this process, no real automation is achieved, and the human actor is involved in the loop.

Composition of simple services in complex ones represents a natural evolution of the technology. Mainly in e-Business scenarios, the standardization efforts to integrate them into business processes have evolved into many proposals, from which BPEL4WS [1] seem to be ahead up to now.

However, the ultimate goal is to have an automation process of the aforementioned tasks required for a web service. For this, the new paradigm of Semantic Web Services comes into the arena. To bring web services to its full potential semantics must be used in order to describe web services full capabilities. The Semantic Web and its key enabling technology, ontologies [2] provide a means to do so. They interweave human understanding of symbols with their machine---processability, allowing declaration and description of services. The combination of Semantic Web and Web Services should enable users to locate, select, employ, compose, and monitor Web-based services automatically [3].

There are two main initiatives to describe Semantic Web Services. OWL-S is an upper-ontology for declaring and describing services by employing a basic set of classes and properties. It is composed of three parts, each one describing one aspect of the service: the Service Profile defines what the Service does, the Service Model how it works and the Service Grounding how to access it.

The Web Services Modeling Framework [4] describes a full-fledged framework for describing Semantic Web Services with relies in the principles of loosely-decoupling and strong mediation, plus four main elements, namely: ontologies, goal repositories, web service description and mediators. For the purpose of this paper, we will stress the fact that both approaches count on the notion of pre-conditions and post-conditions when describing actual web services. In a nutshell, preconditions are conditions over the input and post-conditions are conditions over the output. These concepts will be further developed in the following sections.

Why is this approach useful? There are many ways of modeling web services, and few of them incorporate both a process view of the service and a probability-based description of that service process. We believe this approach is both novel and useful in determining the optimal service description. We provide both a general specification and a specific example of the modeling methodology below.

1.2 What is Subgame Perfection?

First some definitions, then an example.

Definition 1 *The normal form representation of an n-player game specifies the players' strategies* $S_1,\ldots,S_n$ *and their payoff functions* $u_1,\ldots,u_n$. *We denote this game by* $G = (S_1,\ldots,S_n; u_1,\ldots,u_n)$.

In addition to the normal form representation of the game, which we will use briefly at the end of the paper, there is the *extensive form representation* of the game, which we will use for the majority of the paper. Here the set of all legal paths through the space of positions is laid out in a tree. The informational constraints imposed on the players' choices are represented by grouping various 'nodes' of the tree into information sets; the interpretation being that whenever any node in an information set occurs a player must choose a move without knowing which particular node within the information set has occurred.

Definition 2 *The extensive form representation of a game specifies*

1. *the players in the game,*
2. *when each player has her move,*
3. *what each player can do each one of their opportunities to move, and*
4. *the payoff received by each player for each combination of moves that could be chosen by the players.*

It is important to define all our concepts precisely, because we will be manipulating them in slightly unorthodox ways later on. Thus, a *Nash Equilibrium* is defined in terms of the normal form game as the following:

Definition 3 *In the normal form game* $\gamma = (S_1, \ldots, S_n; \delta_1, \ldots, \delta_n)$ *the strategies* $(s_1^{*}, \ldots, s_n^{*})$ *are a Nash Equilibrium if, for each player i,* s_i^{*} *is player i's best response to the strategies specified for the n-1 other players.*

Definition 4 *A subgame is an extensive form game that*

1. *begins at a decision node* η *and is a singleton information set, but not the game's first decision node,*
2. *includes all the decision and terminal nodes following n in the game tree, but no nodes that do not follow n, and*
3. *does not cut any information sets.*

In many textbook examples, in games that assume fully rational players with unbounded computational skills, perfect knowledge of the opponent's skills, with well defined preferences over uncountable sets of lotteries, perfect knowledge of the preferences of the others' preferences, rationality and common knowledge, then the procedure for solving extensive form games is to compute the *Nash Equilibrium* by backwards induction. We will not do so here. Many textbook in game theory (See [5, pp.12-13] for examples) assume a continuum of unbounded integers, for example the reals, **R**, as the field onto which all mappings take place. There is no room inside functioning web service architectures for uncountable infinities. For an excellent summary of the seminal contributions to the emerging area of explicitly computable games, see [6], especially pages 89--97.

The procedure for constructing a subgame perfect Nash equilibrium is as follows. First, identify all the subgames that contain terminal nodes in the original game tree. Then replace each such subgame with the payoffs from one of its Nash equilibria.

Now think of the initial nodes in these subgames as the terminal nodes in a truncated version of the original game. Working backwards through the tree in this way yields a subgame perfect Nash equilibrium because the player's strategies constitute a Nash equilibrium in every subgame.

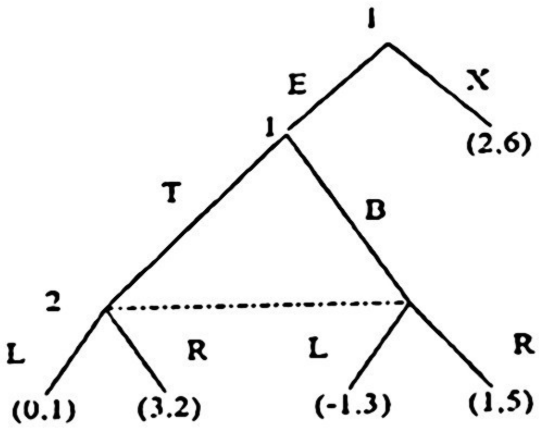

Fig. 1. Subgame Perfection with incomplete information

1.3 An Example

Consider the extensive form game shown in figure 1 above. This game has two subgames: one starts after player 1 plays *E*; the second one is the game itself, which can see from Figure 2 below.

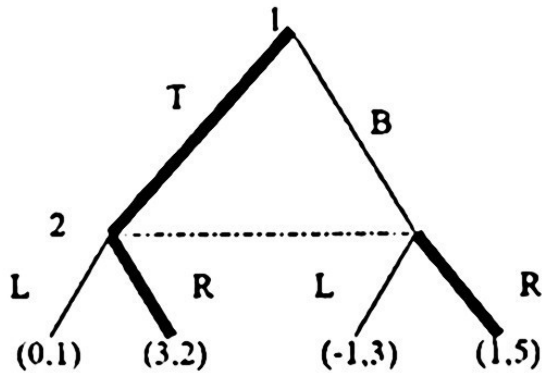

Fig. 2. The Subgames in Relief

We compute the subgame perfect equilibria as follows. We first compute a Nash equilibrium of the subgame, then fixing the equilibrium actions as they are (in this subgame), and taking the equilibrium payoffs in this subgame as the payoffs for entering in the subgame, we compute a Nash equilibrium in the remaining game.
As we can see from figure 3 below, the subgame has only one Nash equilibrium, as *T* dominates *B*: Player 1 plays *T* and player 2 plays *R*, yielding the payoff vector (3, 2).

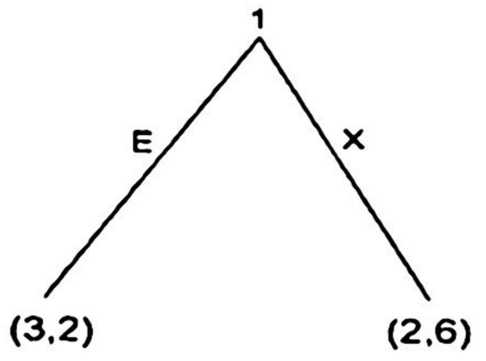

Fig. 3. Final Payoff to the Subgame Perfect Nash Equilibrium

Subgame perfection generalizes the notion of Nash equilibria to general dynamic games via the following definition:

Definition 5 *A Nash equilibrium is said to be subgame perfect iff it is a Nash equilibrium in every subgame of the game.* What is a subgame? In any given game, there may be some smaller games embedded; we call each such embedded game a subgame. Now we are ready to begin our model.

2 The Model

We begin with the description of the general pre—and post-condition schema for web services. Our model takes the 'players' of the game to be the pre and post conditions of the semantically enabled web services themselves. That is, they are predicates without which the service cannot function, and returns an annotated null value ϕ for any query sent to it. It will be clear from figure 3 that there is no way to compute the Nash equilibrium without resorting to the use of subgame perfect equilibria first. This is as a result of the inherent incomputability of the subgame schema, which we mentioned above. It should be noted that there are algorithms for computing sequential equilibria on recursively defined games, and that these equilibria will yield a subgame perfect equilibrium [7]. We do not use this formulation because we feel a more general treatment of the concept is less restrictive and, more importantly, more specific from the viewpoint of semantic web services. We begin with the description of the post condition.

2.1 Post Conditions

Post-conditions in WSMF describe what a web service returns in response to its input. OWL-S defines effects, as the results of the successful execution of the service as described by the OWL-s coalition at http://www.daml.org/services/owl-s/1.0/. These definitions are quite similar in nature: they define conditions that hold *after* the correct execution of the service. Hence we define post-conditions as conditions that hold after the execution of the Web Service. Similar to pre-conditions, our definition is not limited to state-of-the-world post-conditions (changes in the state of the world), but it also includes knowledge post-conditions, i.e. knowledge that is made available to the requester after the service execution. In the example of the service presented before, the post-conditions are the list of books in the given country from the given location. Imagine that this service is not a free service, and that the credit card of the requester is charged by a given amount. Then, the charge of the credit card would be a post-condition as well. Pre-conditions and post-conditions fully described the conditions that must be true *before* requesting the service and the ones that are true *after* its execution.

2.2 Post Condition Subgame

The idea. The model takes queries sent to the web service as the 'players' of the game. The web service can handle $n \in (1,\ldots,N]$ queries in a given time frame $t \in (1,\ldots,T)$. The 'strategies' of the game are simply complete plans of the system, completely deterministic and specified exactly. The payoffs to the queries are their passage through the web service, i.e. completion *C*, or non--completion, *NC*, after which the post condition returns a value of 0 at time *t* to the post conditions via a feedback process Υ which is completely specified and depends on $\Upsilon = (1-\alpha)\Upsilon_{x}^{t-1} + \alpha(\delta_{x}^{t-1})$, where $0<\alpha<1$.

The model assumes that there are discrete queries of the form *(x, y)* sent to the web service, and that these queries represent the search for a good—in our example, a book. These queries are then processed by the pre-conditions, checked individually and then collectively, and then sent to the service for further processing. Assuming the product is in stock, shipping charges are met, etc, the query is sent on, and fulfills the post condition, returning a payoff of either 1 or 0.

Assumption 1 *There is a closed metric space X and M(X) is the probability distribution on X, where X specifies some field* σ *, and at any stage of the process, there is a query x, y over* σ *which assigns some value* δ *to the post condition, where* $\delta \in (0,1)$.

Assumption 2 *The eventual payoff to the post condition from the game* γ *is* δ_{x}^{t} *for some* $\delta \in (0,1)$*, defined above, which is the set of all feasible divisions of this post condition.*

It means that the post condition, once satisfied by the result returned by query *x* at time *t*, will return a value of 1 (A receipt of success) if the service has worked properly.

Assumption 3 *Each query (x, y) is risk-neutral as regards security, and the future is discounted exponentially.*

It can be seen that *D*, the set of all feasible divisions of the post-condition's successful or unsuccessful return is

$$D = (x, y) \in [0,1]^2 \mid x + y \leq 1 \tag{1.1}$$

and the subgame perfect equilibrium will be as follows: query 1accepts *(x, y)* with

$$x \geq \frac{\delta(1-\delta^{2k})}{1+\delta} \tag{1.2}$$

Thus, at any query time $t = 2n - 2k$ where *k* is a non-negative integer, the post condition returns a value of 1. And, of course, rejects with

$$x < \frac{\delta(1-\delta^{2k-1})}{1+\delta} \tag{1.3}$$

Query *y* will form a subgame perfect equilibrium around *x* by definition 4—that the decision node η was a singleton information set—η only considers the situation at the present and previous decision nodes, and not the future nodes. Thus we have query 2 offering

$$(x_{t-1}, y_{t-1}) = \left(1 - \frac{\delta(1+\delta^{2k+1})}{1+\delta}, \frac{\delta(1+\delta^{2k+1})}{1+\delta}\right) \equiv \left(\frac{1-\delta^{2k+2}}{1+\delta}, \frac{\delta(1+\delta^{2k+1})}{1+\delta}\right) \tag{1.4}$$

where the subgame perfect equilibrium can be found by backwards induction or solution of the state variable, δ .

Proof Equation 1.4 follows naturally from [8, pp. 111-112, §2] which also includes a proof of incomputable strategies in repeated games.

Consider figure 4 below. Here we see the service description for our example.

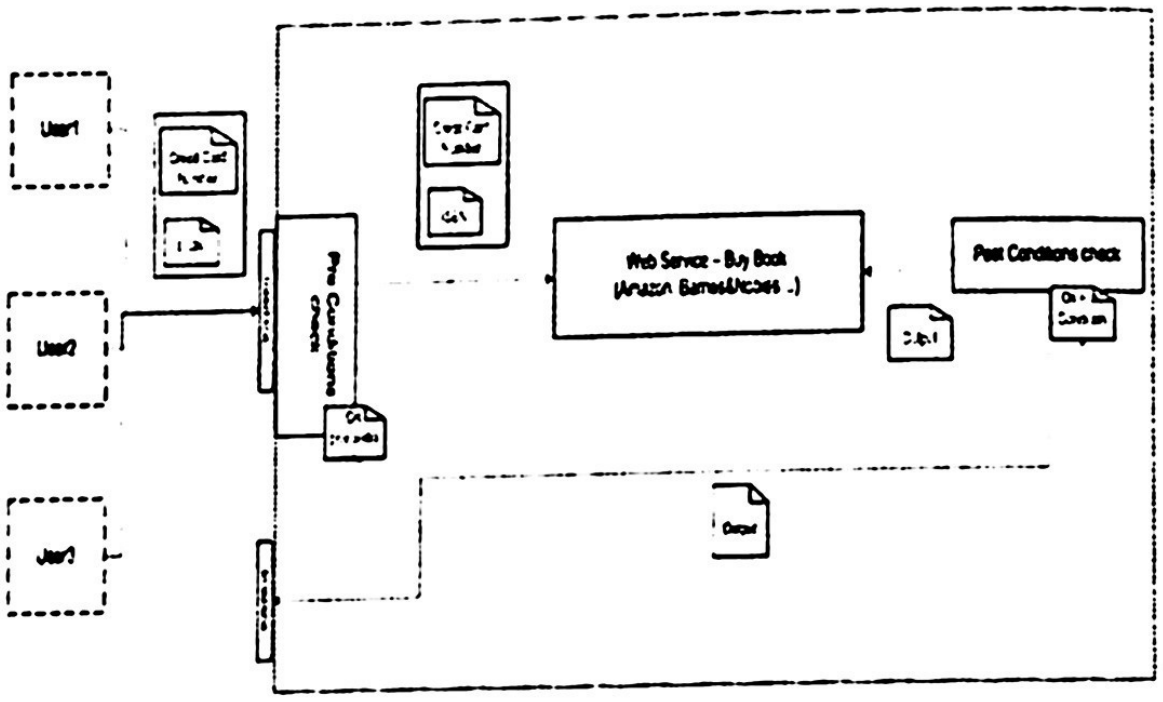

Fig. 4. Service Description 1-Process View

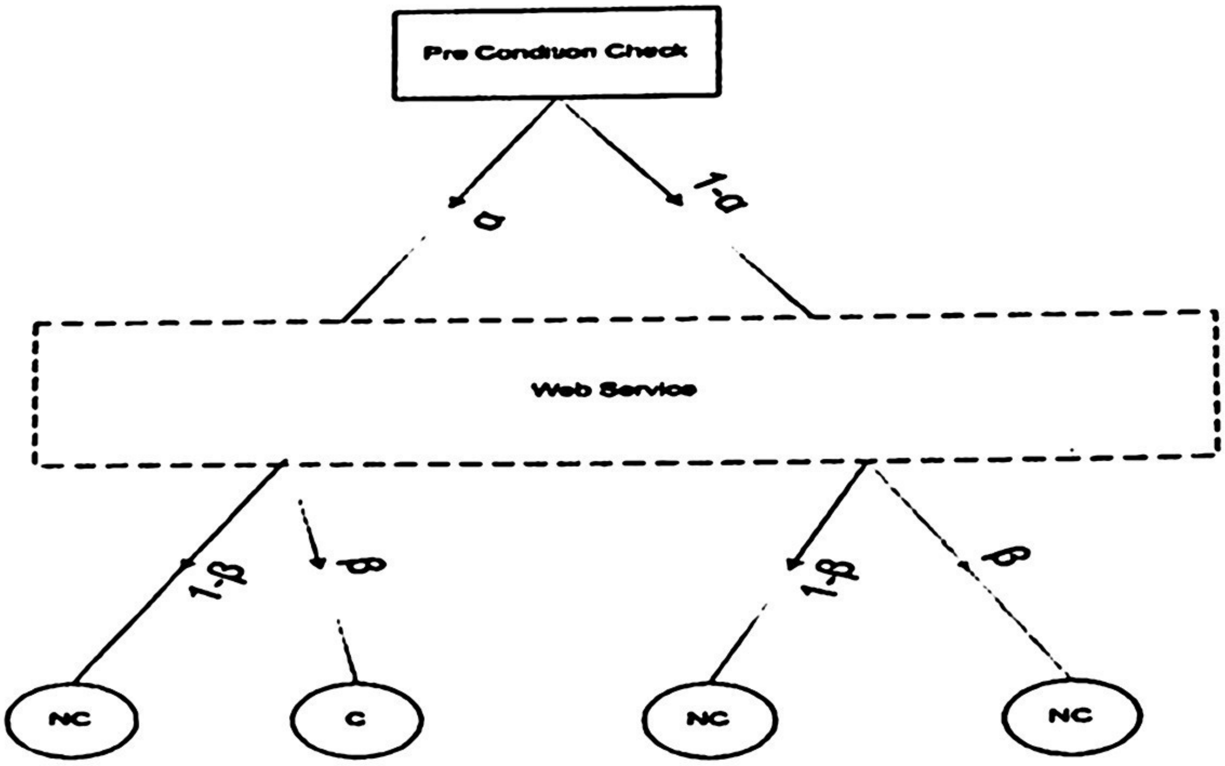

Fig. 5. Service Flow Description 2, extensive form game theoretic view

2.3 Pre Conditions

Pre-conditions in WSMF describe what a web service expects for enabling it to provide its service. OWL-S (http://www.daml.org/services/owl-s/1.0/) bases its definition of *Web Service functionality* in concepts coming from the planning community, adapting the definition of a pre-condition to the Web Services domain. In OWL-S, pre-conditions are defined as logical conditions that must be satisfied before requesting the service. These conditions are referred to the state of the world, in contrast with inputs, that are regarded as knowledge pre-conditions. It can be seen that all these definitions reflect a similar concept: conditions that must hold in order to apply an action (similarly, to request a service). Pre-conditions act as pre-requisites to use a service or action, that is, they must hold prior to the use of the service or action. Following this line, we define pre-conditions as conditions that *must* hold previous to the request of the Web Service for enabling it to provide its declared service. As an example, consider a Web Service that provides a list of the books in a given country, from a given location. The pre-conditions of this service are the location and the country. Now imagine this Web Service provides its service only to registered users that paid their monthly subscription rate. In this case, an additional pre-condition would be that the requester has paid the rate.

2.4 Pre Condition Subgame

Let α be the collection of first stage strategies $(\alpha_1,\ldots,\alpha_t)$ and β be the collection of second stage strategies $(\beta_1,\ldots,\beta_t)$. Then $\beta(\alpha)=\left(\beta_1(\alpha),\beta_2(\alpha),\ldots,\beta_y(\alpha)\right)$ indicates the choices made by the second stage of the web service after returning a positive check on pre-condition α. Again, the expected payoff is a sum over the finite space, X, and recording the 'moves' of (α,β) is $E[\delta_x \mid \alpha,\beta \geq \delta_x[\bar{\alpha}_x,\alpha_{-x},\beta],\forall\bar{\alpha}_x\in\alpha]$, and similarly for β_y. This being given, two of the conditions of the subgame must be generated to satisfy definitions 3 and 4.

$$\forall x \in X, E[\delta_x \mid \alpha,\beta \geq \delta_x[\bar{\alpha}_x,\alpha_{-x},\beta]]\forall\bar{\alpha}_x\in\alpha \tag{1.5}$$

and

$$\forall y \in X, E\left[\delta_y \mid \alpha,\bar{\beta}_y(\alpha)\right],\forall\bar{\beta}_y\in\beta \tag{1.6}$$

Condition (1.5) states that every first-stage query will select a best response if that is possible. Condition (1.6) requires that for any realization α in the first stage,

$\beta(\alpha)$ be a Nash equilibrium of the ensuing second-stage game. For any $\alpha_{*} \in \alpha$, let α_{x} be the set of Nash equilibria of the second-stage game defined after α is realized. Thus, $\beta(\alpha) \in NC(\alpha)$ iff (1.6) holds for $\beta(\alpha)$.

As an example, consider Figure 5. Here we re-enter our example, looking at our 'buy book' example from a different view, reformulating the problem of pre- and post- condition selection in terms of probabilistically defined recursive strategies as discussed in [8]. We have two pre-conditions that must be satisfied before the web service can be initiated. These are: that the prospective customer must have a valid ISBN number and they must have a valid credit card number. Call the probability that the 'ISBN' is valid α and the corresponding probability that the credit card is valid β. Now see the process reformulated as a sequential decision making problem: even if both pre-conditions are being checked in parallel, it is useful to see one's dependency on the other for the initiation of the web service. Now, because of the open—world assumption of most web services, and the incomplete nature of most information sources, a useful way to solve this problem is to search for the subgame equilibrium within the system, in this example the only channel through which the web service will be initiated--- the payoff *C*, highlighted in red. Only when both pre-conditions are satisfied (i.e. when the extensive form tree structure is at 1 at the decision node, and all other positive probabilities are summed over *M(X)*) will the web service be invoked. It only remains to compute the overall equilibrium. From (1.5) and (1.6) above, we can see that the overall equilibrium for the pre-conditions will be found at $\bar{\beta}_{*}(\bar{\alpha}_{*})$ although the disequilibrium dynamics of a real time functioning system may keep the strict equilibrium set of queries far from this equilibrium. Nevertheless, given our game as constructed, there is an overall ideal solution point, where pre-and post-conditions are at their subgame-perfect equilibria and the feedback mechanism defined above returns a value of 1.

2.5 Implications

The implications of our model for construction of Semantic Web Services could be widely grouped in three main areas.

Execution performance In business processes where the flow of execution reverts into a big workload and fault tolerance problem for the hardware and software systems involved. Our model could be a best-of-breed solution to assign dynamically resources to the web services with a lower probability of being executed. In our further work section, we look to a use case study for just such an eventuality.

Business intelligence performance In order to avoid the trap of producing huge inconsistent designs or not balanced architectures in modern e-Commerce systems, our model provides a formal and consistent metrics to evaluate the whole performance of the processes and the execution flow in terms of business intelligence.

Graphs and heuristics depending on our model could enhance the re-engineering and come out with a better solution in terms of cost-effectiveness and efficiency.

Non-functional information A detailed description of the service based on metadata or on semantic annotation could be inferred from our model. The Quality of Service (QoS) or policy, as described in [9] could add some information about the performance of the system. Also pay-per-use techniques, such as currently applied in real-world e-Business applications can learn from the estimated use provided by our model.

Amongst others future directions of the model would allow a further analysis of the scalability in high-computing grids or overlay networks.

3 Conclusion and Related Work

This paper has given a new way to characterize web service description as a series of probabilistic dependencies upon several pre- and post- conditions. In our admittedly rudimentary example the web service itself is purposely left a mystery, a 'black box' around which we highlight the ancillary but crucial role of pre- and post conditions in invoking and feeding back to the web service the queries that trigger it. We recognize the fact that, as conceived, this model is uncomputable and therefore of little practical use the web service developer. However, a simple restriction of this more general schema by defining a recursive game would be enough to allow the computation of the subgame perfect equilibrium in a sequential game. As stated above, this model fails only in its generality, not in the execution of the pre- and post- condition schema outlined above. We feel that this amounts to a suggestive failure, and one on which further work can and will be built.

3.1 Related Work

Several approaches investigated the issue of using pre- and post-conditions in the software engineering area. Preconditions, post-conditions and invariants are usually identified during the software development process, especially during the analysis and design phases. The UML specification [10] defines the Object Constraint Language (OCL) [11] that allows a designer to specify constraints of this kind on a particular object or method. Recent trends in the Semantic Web Services area such as the Web Service Modeling Ontology (WSMO) [12] use pre-conditions and post-conditions at the capability level of the web service. Finally, reference implementations such as the Web Services Execution Environment (WSMX) [13] pretend to integrate them in their future conceptual model and used them in runtime, as part of their execution model.

3.2 Future Work

When one introduces one web service and connects it to another and then another in a supply chain, for example, one increases the probability that one failed pre-condition will cause the entire chain of services to fail. Future work will concentrate on three aspects of this problem.

1. What is the specific rate of decay of this chaining of web services?
2. A use-case study based on the Adaptive Services Grid (ASG) project. ASG provides the integration of its sub-projects in the context of an open platform, including tool development by small and medium sized enterprises. Based on semantic specifications of requested services by service customers, ASG discovers appropriate services, composes complex processes and – if required – generates software to create new application services on demand. A case study examining just how long these integrated chains of software services can be will compare the predicted rate of decay to the actual rate observed throughout the project.
3. A tool to simulate the development of complex integration processes based upon the similarity (or lack thereof) of pre- and post- condition architectures.

Other future work will concentrate on refining several sub-classes of game-theoretic models like the rudimentary one above for use in the description of semantic web services in specific cases. The web service itself, modeled as a black box in this paper, can be recast as a sequential decision process, albeit a very complex one. This paper is a small step in that direction.

References

1. Curbera, F., Goland, Y., Klein, J., Leyman, F., Soller, D., Thatte, S., Weerawarana, S.: Business Execution Language for Web Services. BEA Systems and IBM Cor- poration and Microsoft Corporation (2002)
2. Fensel, D.: Ontologies, A Silver Bullet for Knowledge Management and Electronic Commerce. 2nd edn. Springer (2003)
3. McIlraith, S., Son, T., Zeng, H.: Semantic web services. IEEE Intelligent Systems. (Special Issue on the Semantic Web) 16 (2002) 46–53
4. Fensel, D., Bussler, C.: The web service modelling framework wsmf. Electronic Commerce Research and Applications 1 (2002)
5. Gibbons, R.: A Primer on Game Theory. Prentice Hall, New York (1992)
6. Velupillai, K.V.: The computable approach to economics. Taiwan Journal of Political Economy 1 (1995) 53–109
7. Azhar, A., McLennan, A., Reif, J.: Computation of equilibria in noncooperative games. Technical report, Duke University (1992)
8. Velupillai, K.V.: Computable Economics. Oxford University Press, Oxford (2000)
9. M.Dean, G.Schreiber: OWL Web Ontology Reference. http://www.w3.org/TR/2003/PR-owl-ref-20031215/ (2004)
10. Booch, G., Rumbaug, J., Jacobson, I.: The Unified Modelling Language (UML). (1999)

11. Warmer, A., Kleppe, J.: The Object Constraint Language (OCL): Precise modeling with UML. (1985)
12. Roman, D., Lausen, H., Keller, U.: Web Services Modelling Ontology (WSMO). (2004)
13. Cimpian, E., Mocan, A., Moran, M., Oren, E., Zaremba, M.: Web Services Execution Environment (WSMX). (2004)