WSML Deliverable

D28.4 v0.1

NON-FUNCTIONAL PROPERTIES IN WEB SERVICES

WSML Working Draft – November 16, 2005

Authors:
Ioan Toma

Editors:
Ioan Toma

Reviewers:

This version:
http://www.wsmo.org/TR/d28/d28.4/v0.1/20051116/

Latest version:
http://www.wsmo.org/TR/d28/d28.4/v0.1/

Previous version:
none

Copyright © 2004 DERI ®, All Rights Reserved.
DERI liability, trademark, document use, and software licensing rules apply.
Abstract

This deliverable addresses the description of non-functional properties for Web services. It provides a survey of solutions to this problem for Web services in particular and Software Systems in general. Furthermore, it takes the most relevant non-functional properties models adopted from O’Sullivan et al., 2005 and provides possible solutions for specifying them in a formal manner, using WSMO/WSML. These solutions can be seen as possible extensions of non-functional properties description support in WSMO/WSML.
Contents

1 Introduction 4

2 Survey of Non-functional properties approaches 6
   2.1 NFP and Web services 6
   2.2 NFP and Software Systems 7

3 A conceptual model for Non-functional Properties 10

4 Supporting Non-functional Properties description in WSMO/WSML 12
   4.1 Pure ontology-based approach 12
   4.2 WSML extensions or a new language for NFP support? 13

5 Conclusions and Future Work 15

Bibliography 16
1 Introduction

Built on current Web services technologies, like WSDL\(^1\), SOAP\(^2\) and UDDI\(^3\), Semantic Web services are providing a new level of automation for service related tasks like: discovery, composition, selection, negotiation, invocation, etc. Both technologies, Web services as well as Semantic Web services, consider services as fundamental, central entities. How these services are described is crucial for the successful realization of all previous mentioned service related tasks.

Three different aspects must be considered when talking about service descriptions: (1) functional, (2) behavior and (3) non-functional. The functional description contains the formal specification of what exactly the service can do. The behavior description contains the formal specification of how the functionality of the service can be achieved. Finally, the non-functional descriptions captures constraints over the previous two\(^4\). Among these three, the first two aspects of service description are the most investigated aspects. Although non-functional properties did not capture a very broad attention from the Web service research community, as functional and behavior descriptions did, one has to recognize the big importance of describing them. This is due to their high relevance for all service related tasks.

Non-functional properties might play an important in all service related tasks, especially in discovery, selection and substitution of services. It is simple to imagine a scenario in which services which can fulfil a user request and which provide basically the same functionality are selected based on some non-functional properties like price or performance.

The lack of real support (languages, methodologies, tools) for non-functional properties might be due to various factors:\(^5\):

- non-functional properties are usually too abstract and most of the time they are stated informally.
- in most of the cases there is no clear delimitation between the functional and non-functional aspects of a service.
- non-functional properties are often consider to be represented after the functional and behavior have been described.
- non-functional properties very often conflict and compete with each other (e.g. availability and performance).

In this document we investigate some existing solutions for describing non-functional properties of Web services in particular, and Software Systems in general, and we sketch some possible solutions for describing in a more useful manner non-functional properties. We do this by using semantic technologies, more precisely the Web Service Modeling Language (WSML)\(^6\) as a formal language to write down the non-functional properties specifications. A semantic approach for describing non-functional properties will basically enable reasoning on these properties, thus supporting the automation of tasks like: discovery, selection, composition, etc. that use these descriptions.

The document is structured as follows: Section 2 provides an overview of some of the existing solutions for describing non-functional properties of Web services and sketches some possible solutions for describing them. Section 3 gives an overview of some of the existing solutions for describing non-functional properties of Semantic Web services and sketches some possible solutions for describing them. Section 4 sketches some possible solutions for describing non-functional properties of Software Systems in general.
services in particular and Software Systems in general. Section 3 describes shortly a conceptual model, more precisely a set of models for non-functional properties provided in [O’Sullivan et al., 2005]. Having such a conceptual model in mind we try to sketch some possible solutions to write down the non-functional properties descriptions in Section 4. Finally, in Section 5 we conclude and present our future work.
2 Survey of Non-functional properties approaches

In this section we take a look at some of the most relevant approaches for describing non-functional properties of Web services and Software Components. Our intention is to give an overview of: (1) what is the set of relevant non-functional properties considered by each approach, (2) how they are modelled and (3) how they are formally represented. Besides the approaches presented here, other approaches might be analyzed and included in this survey in the future versions of this document.

2.1 NFP and Web services

This section gives an overview of the support provided by different Web services and Semantic Web services approaches in describing the non-functional properties of a service.

- **UDDI**
  The Universal Description, Discovery and Integration (UDDI) [Bellwood, 2002] defines a set of non-functional properties for a service provider identified by a businessEntity. The set of non-functional properties contains: the address, the phone numbers, and the email addresses of the service provider. Additionally some other information (metadata) about the service is available like for example the service category. Schemas like UNSPSC [UNS, 2000] can be used for this purpose. The set of non-functional properties and not only can be accessed using the set of UDDI APIs.

- **OWL-S**
  The OWL-S [The OWL Services Coalition, 2004] approach consider the following set of non-functional properties: service name, text description, quality rating, service category. Other non-functional properties can be included by using the ServiceParameter from ServiceProfile. These non-functional properties are described in the Service Profile part and explicitly formalized using OWL [McGuinness and van Harmelen, 2004].

- **SWSF**
  In Semantic Web Service Ontology (SWSO) [SWSL, 2005c], part of Semantic Web Services Framework (SWSF) [SWSL, 2005a] approach, Service Descriptors are used to provide information about non-functional aspects and/or provenance aspects of a service. The following non-functional properties are considered: service name, service author, service contact information, service contributor, service description, service URL, service identifier, service version, service release date, service language, service trust, service subject, service reliability and service cost. SWSO dose not provide a model for the non-functional properties of a service. The non-functional properties can be described by using relations like for example: name(service, service name). These relations can be expressed using SWSL-FOL [SWSL, 2005b] or SWSL-Rules [SWSL, 2005b].

- **WSMO**
  The Web Service Modeling Ontology (WSMO) [Roman et al., 2005] ap-
approach recommends a set of non-functional properties for each particular element of a Web service description. For example the service recommended non-functional properties are: accuracy, contributor, coverage, creator, date, description, financial, format, identifier, language, network-related QoS, owner, performance, publisher, relation, reliability, rights, robustness, scalability, security, source, subject, title, transactional, trust, type, version. The common set of non-functional properties for all elements in WSMO is provided by Dublin Core Metadata Initiative [Weibel et al., 1998]. WSMO does not provide a model for the non-functional properties of a service. Using WSML [de Bruijn et al., 2005], the Web Service Modeling Language, one can assign values to the non-functional properties of a WSMO element. Such a value can be any identifier and thus it can be an IRI, a data value, an anonymous identifier or a list of the former.

2.2 NFP and Software Systems

How to model and formalized non-functional properties of software components and software architectures has been an intense research area, from which Web services can benefit. In this section we are going to shortly describe some approaches for specifying non-functional properties for a software system. The focus is on the language used to specify the non-functional properties.

- **Process**

  In [Rosa et al., 2002] a language for non-functional properties called Process is proposed. The language has been designed to consider some specific aspects when modeling non-functional properties like: correlations and conflicts between non-functional properties. An example of non-functional properties that might be in conflict are: security and performance. A high security may probably determine a loss in performance and vice versa. Three abstractions are used to model a non-functional property. These are: NF-Attribute, NF-Property and NF-Action.

  NF-Attributes model non-functional characteristics that can be precisely measured (e.g. performance), non-functional properties that can not be quantified (e.g. security) and non-functional properties that simply can exist or not for a system (e.g. transactional properties like atomicity, consistency, etc.). NF-Attributes can be derived from other NF-Attributes and thus a class hierarchy of NF-Attributes can be created. The root of this hierarchy in the NFR class. Process takes rather a object oriented approach when defining NF-Attributes. A NF-Attribute definition can be seen as a class definition that might have other NF-Attributes as members. The user has the possibility to specify in a NF-Attribute definition the necessity to have the NF-attributes members defined or not (i.e. all, any, none, exactly one).

  NF-Actions model characteristics that affect the NF-Attributes. These characteristics can be software or hardware characteristics. An example of NF-Action is the encryption algorithm which has a major influence on the security NF-Attribute.

  NF-Properties model constraints over the NF-Attributes. These are expressed in terms of different levels for a NF-Attribute. For example the strong performance is a NF-Property which constrain the performance NF-Attribute. The following levels are considered: strong, medium, low.
Templates are used to model all the abstractions in Process$^{NFL}$. NF-Attributes have for example the following structure:

```java
attribute attributeId1 extends attributeID2{
    primitives primitiveAttributes;
    contribution kindOfContribution;
}
```

Using this template we can define the non-functional property performance as follows:

```java
attribute performance extends NFR{
    primitives space_performance, time_performance;
    contribution oneX;
}
```

One major drawback of Process$^{NFL}$ is that it has no formal semantics. Although, we think that some Process$^{NFL}$ features like: (1) the possibility to express correlations and conflicts between non-functional properties and (2) the possibility to express the compositional aspect of non-functional properties and strengths should be available in a good non-functional language.

- **CQML**

In [Aagedal, 2001], Aagedal is proposing a modeling language for Quality of Services (QoS) called Component Quality Modelling Language. The following abstractions are used to model the QoS: QoS Characteristic, QoS Statement, QoS Profile and QoS categories.

QoS Characteristics are defined as use-defined types and they are the building blocks of QoS specifications. For example delay, which is an important QoS Characteristic for a real time service can be modelled as follows:

```java
quality_characteristic delay{
    domain: decreasing numeric milliseconds;
}
```

One can specify how characteristic are influencing the quality of the service. In our example the characteristic delay is decreasing the quality. QoS Characteristics can be derived from other characteristics (e.g. statisticalDelay:delay).

QoS Statements are used to specify restrictions of the values of QoS Characteristics. Taking our previous example we can specify that we have a low delay in case the value of the delay is lower then four. We write this using CQML as below:

```java
quality low_delay{
    delay < 4;
}
```

QoS Profiles are used to bind QoS statements with component specifications. In our example the QoS statement low_delay is bound to a component specification called component_specification:

```java
profile goodComponent for component_specification{
    provides low_delay;
}
```
All the previous mentioned abstractions: QoS Characteristics, QoS Statements and QoS Profiles can be grouped in QoS Categories. In the example below two QoS Characteristics: delay and output are grouped together:

```java
quality_category timelines{
    delay;
    output;
}
```
3 A conceptual model for Non-functional Properties

In this section we shortly describe the set of the most relevant non-functional properties of a service according to [O’Sullivan et al., 2005] technical report. We subscribe to the set of non-functional properties they proposed and we adopt the models for non-functional properties provided by them. For a full description of these models we refer the reader to the previous mentioned technical report.

1. **Service Provider**
   The *Service Provider* model captures information about: the service identifier which can be a UN/SPSC code, the service name and the provider of the service.

2. **Temporal Model**
   The *Temporal Model* provides the temporal concepts that are needed for time related descriptions of a service. These are: Temporal Date, Time, Temporal Interval and Temporal Duration. These concepts can further be refined in more specific concepts like Calendar Date for example.

3. **Locative Model**
   The *Locative Model* is used to model the location of a service. Concepts like: Address, Region, Route, Point, Street Directory Reference, PhoneNumber, URI, IPAddress and Spectrum are directly related to this model.

4. **Service availability**
   The *Service availability* combines temporal and locative aspects of the service to describe when and where one can interact with the service.

5. **Obligations**
   The *Obligations* model captures the responsibilities of both service requestor and service provider. Three kinds of obligations were defined: Pricing obligations, Payment obligations and Relationship obligations.

6. **Price**
   *Price* and *Payment* are seen as complementary non-functional properties. They represent two views of the same thing but from different perspectives. The payment(cost) is the user’s perspective and the price is the provider’s perspective.

7. **Payment**
   The *Payment* model captures the manner in which a service requestor can fulfill their payment obligations. As stated before payment and price are complementary.

8. **Discounts**
   Close related to the notions of price and payment is the notion of *Discount*. Discounts are viewed from the service requestor perspective and are categorized according to the payment method and requestor’s identity.

9. **Penalties**
   The *Penalties* are used by a service provider to specify what exactly will occur if a service requestor does not comply with a specific obligation.
10. **Rights**
   The *Rights* model captures the permissions granted to service providers and service requestors to perform operations.

11. **Language**
   Three kinds of *Language* support for describing a service are distinguished: *written language* (e.g. English), *spoken language* (e.g. English) and *standard language* (e.g. WSDL).

12. **Trust**
   *Trust* is a notion understood in various ways by different people. The Trust model is directly influenced by other models like *endorsement* and *service inception*.

13. **Quality**
   *Quality* is described relative to a standard, an industrial benchmark and/or a ranking schema.

14. **Security**
   The *Security* model is attached to the locative aspect of the service and it’s divided in two dimensions: identification and confidentiality.
4 Supporting Non-functional Properties description in WSMO/WSML

In the previous section we have shortly described a set of models for the non-functional properties of a service provided in [O’Sullivan et al., 2005]. Based on these models, we are sketching in this section two possible extensions for describing the non-functional properties of a service in WSMO/WSML. We take the temporal availability non-functional property as an example and we provide for both approaches the formalization of this non-functional property. Our final goal is to find the best solution for: (1) how to determine and specify the actual value for a non-functional property and (2) how to formalize it so that it could be machine readable and reasoned about.

4.1 Pure ontology-based approach

This approach, which we call "pure ontology-based approach" dose not imply any extensions of the WSML language. It is based only on the usage of ontology(ontologies). Ontologies are used to describe the non-functional properties domain. Further the ontology can be imported and concepts refereing to the non-functional property can be instantiated and used in the service description. For a better understanding we exemplify below the modelling in WSML of the temporal availability of a service. We take a ticket booking service as an example but we don’t focus on the capability and interface but rather on the non-functional properties of the service. We want to express that the service is available every Monday to Saturday from 07:00 to 24:00 CET and is not available on the rest of the time (Monday to Saturday from 24:00 to 07:00 and on Sunday) due to various reasons (e.g. backup operations, etc.).

```xml
wsmlVariant: "http://www.wsmo.org/wsml/wsml−syntax/wsml−flight"
namespace: {"http://example.org/temporalNFPOntology#", 
               dc: "http://purl.org/dc/elements/1.1",
               wsml: "http://www.wsmo.org/wsml/wsml−syntax#" }
ontology: "http://example.org/temporalNFPOntology"

concept dayOfWeek subConceptOf integer
    dc#description hasValue "day of a week is represented by an integer from 1 to 7"
    dc#relation hasValue validDayOfWeek
endnfp

concept tempAvailability
    startTime ofType time
    endTime ofType time
    day ofType (1 ∗ dayOfWeek)

axiom validDayOfWeek
    definedBy
    !− ?x memberOf dayOfWeek and
    lessEqual(?x,1) and greaterEqual(?x,7).

instance instTempAvailability memberOf tempAvailability
    startTime hasValue_ time(07,00,00,01,00)
    endTime hasValue_ time(24,00,00,01,00)
    day hasValue [1,2,3,4,5,6]
```

Listing 4.1: Ontology for Temporal non-functional properties
The ontology for temporal non-functional properties should contain a concept like `temporalAvailability` which is modelled as a concept in the previous ontology. The other non-functional properties mentioned in Section 3 should be as well modelled.

```xml

webService "http://example.org/trainTicketBookingService#"
  nfp dc#description hasValue "train ticket booking service"
  nfpO#temporalAvailability hasValue nfpO#instTempAvailability
endnfp

importsOntology { "http://example.org/temporalNFPOntology" }

capability ttbs#trainTicketBookingServiceCapability
  sharedVariables ?x
  precondition definedBy (?x memberOf ItinerarySearch and ?y memberOf ItinararySelection) or ?x memberOf Itinerary.
  postcondition definedBy exists ?y (?y memberOf Reservation[forItinerary hasValue ?x]).
```

Listing 4.2: Temporal non-functional properties usage for the TrainTicket service

### 4.2 WSML extensions or a new language for NFP support?

In this approach the specification of non-functional properties for a service is supported by extensions of the WSML language itself or by defending a new language for non-functional properties that can be easily integrated with WSML. A syntax and semantic have to be develop for this extension of WSML or for the new language. The language/extension might have constructs like `temporalAvailability` to specify the temporal availability of a service. Other constructs will be needed as well. We sketch below a possible solution.

```xml

nonFunctionalProperty temporalAvailability(?startTime, ?endTime, ?day)
  definedby
  ?startTime memberOf time and ?endTime memberOf time and lessEqual(?startTime, ?endTime) and ?day memberOf dayOfWeek
```

Listing 4.3: Definitions of Non-functional properties

```xml

webService "http://example.org/trainTicketBookingService#"
  nfp dc#description hasValue "train ticket booking service"
  wsmlnfp#tempAvailability (?startTime, ?endTime, ?day) definedby
  ?startTime memberOf time and ?endTime memberOf time and lessEqual(?startTime, ?endTime) and ?day memberOf dayOfWeek
endnfp

importsOntology { "http://example.org/temporalNFPOntology" }
```

Listing 4.3: Definitions of Non-functional properties
Listing 4.4: Extension support for non-functional properties

capability ttb#trainTicketBookingServiceCapability
sharedVariables ?x
precondition
  definedBy
  (?x memberOf ItinerarySearch and ?y memberOf ItinararySelection) or ?x memberOf Itinerary.
postcondition
  definedBy
  exists ?y (?y memberOf Reservation[forItinerary hasValue ?x]).
5 Conclusions and Future Work

This version of the deliverable aims to be a first step towards defining a better support for Web services non-functional properties descriptions in WSMO/WSML. We have surveyed some of the most interesting solutions for defining and modelling non-functional properties and we have proposed some initial solutions for providing richer and useful support for non-functional description of Web services in WSMO/WSML.

As future work we plan to further develop the set of ontologies mentioned in Section 4 using WSML [de Bruijn et al., 2005] and to illustrate how they can be used in describing the non-functional aspects of a service. More detailed examples of service non-functional properties should be provided. Within this process we will figure out if WSML can support all requirements for specifying non-functional properties descriptions of services. Once we have a good understanding of this we can think about the second solution proposed in Section 4, which is basically the extension of the WSML with non-functional properties constructs or a new language for non-functional properties.

Acknowledgements

The work is funded by the European Commission under the projects DIP, Knowledge Web, SEKT, SWWS, ASG and Esperonto; by Science Foundation Ireland under the DERI-Lion project; and by the Vienna city government under the CoOperate program and by the FIT-IT (Forschung, Innovation, Technologie - Informationstechnologie) under the projects RW and TSC.

The editors would like to thank to all the members of the WSML working group for their advice and input into this document.
Bibliography


