DESCRIPTION AND DISCOVERY OF SERVICES

A practical approach

Carlos Alfaiaite, José Delgado
Instituto Superior Técnico, Universidade Técnica de Lisboa, Rua Rovisco Pais nº 1, Lisbon, Portugal
carlos.alfaiaite@ist.utl.pt, jose.delgado@ist.utl.pt

Keywords: Service description, service discovery, SOA, WSDL, UDDI.

Abstract: Although there is some consensus that the conventional Web-services technologies based on XML, WSDL and UDDI can be made to succeed, there is a matching degree of confusion and uncertainty about its future. WSDL is a rigorous, expressive and flexible standard to describe web-services. However, it trades simplicity for expressiveness and flexibility. The service discovery architecture based on centralized UDDI repositories is not having the expected acceptance. This research proposes a different approach to the service description that is not based in XML but it’s compatible with it. It is a service description independent of the format of the data sent, focused in the business information and intended to simplify the work of the programmer of web-services. It is explained how our proposal is used from the service description creation, to its publication and usage and the differences to the Web-services approach. It is demonstrated that this approach is feasible, allows the development of simple, service centric description documents and enables the adoption of a dynamic service discovery architecture that semi-automates the management of the published service description documents.

1 INTRODUCTION

Although there is some consensus that the conventional web-services "stack" based on WSDL and UDDI can be made to succeed, there is a matching degree of confusion and uncertainty about its future. The history of software systems to date teaches that rigid, over-elaborate systems do not survive, not only because they do not have the flexibility to adapt to change but also because people do not like to be constrained by them. Web services in their current state seem to be at the mercy of those who want to control and regulate for every possible eventuality, to such an extent that the whole becomes unmanageable, even incomprehensible [Wright 2005].

Many business problems don’t need the expressiveness and flexibility of the Web-services specifications proposed by W3C and OASIS. It’s desirable to have an approach to service description and discovery that could allow developing simple solutions for simple problems. This investigation we focus on developing a simple and lightweight approach to describe and discover web-services. This approach follows principles defined as essential to ensure that our proposal respect the vision of this investigation. We start by defining our vision and objectives in this chapter. In chapter 2, we present previous work used to develop our proposal. The specification of our proposal to describe and discover services is described in chapter 3. Chapter 4 presents implementation details about the components and mechanisms of the proposals. Finally in chapter 5, are presented the main conclusions of this investigation and the future work.

1.1 Vision

Software applications and infrastructure are organized into a set of interacting service which enables business transactions through the Web and the participation of all the society in the creation and proliferation of new services as a way to interact with each other. The users of the Web have a simple solution to create, describe, find, combine and use their Web-services, which may have a business nature or not. The enterprise sphere, which might have complex
problems to be solved by Web-Services, has a more expressive, robust and complex solution indented to provide stronger guarantees of satisfying the business requirements.

1.2 Strategy

Analyse the characteristics of the current WS-* specifications related to service description and discovery, understand their main components and what can be adapted and re-use. Propose mechanisms to facilitate and semi-automate the creation of Web-services and their service description documents; the adoption of a dynamic architecture that allows their discovery with reduce management effort of the service description documents, ideally being transparent to the user. The focus on reduce the effort of the tasks done by humans and the simplification of the documents necessary to describe Web-services provides a simple solution to the Web users. The extensibility and adaptability of the simple solution allows the development of more complex mechanisms to solve more complex problems.

1.3 Tactic

In the analysis on the service description document proposed by W3C, WSDL we focus on how to separate the business details from the technological details removing the latter from the service description document. Since the technological details are removed we assess what mechanisms, necessary to provide these details, must be supported by the systems implementing the Web-Service. This way we discover the price to pay for the simplicity of the service description document and evaluate compare it with the WS-* proposal. The proposed functionalities necessary to develop Web-Services give the basis for the specification of the necessary tools. Our proposal to the service description document serves the same objectives of WSDL and they should be compatible and the conversion of WSDL documents in our proposed document is very important since WSDL is widely used. Regarding service discovery we analyse the existing architectures and chose the more dynamic and adaptable. Also the UDDI and search engines characteristics are studied and compared taken in account on how well they fit in the service discovery architecture adopted.

1.4 Problem Statement

The problems to be solved are:

- Develop a service description specification that respect the strategy of this thesis and investigate all the necessary components to support the creation, description, composition and execution of Web-services;
- Analyse and adopt a service discovery architecture that better fits the strategy of this thesis.
- Develop a proposal that is simple use to solve simple problems and extensible so that it can be developed to solve complex problems

2 RELATED WORK

This chapter presents some theoretical research previously done to help develop our proposal.

2.1 Web Services

Web Services are the current most promising technology based on the concept of Service Oriented Computing [Weerawarana, 2005 as cited in Papazoglou et al., 2006] and can be considered as an implementation of SOC model [Shen, 2006]. Also at the implementation level, UDDI, WSDL and SOAP provide such capabilities as discovery, deployment and communication, while specifications such as BPEL4WS provide service composition. Web services provide the basis for the development and execution of business processes that are distributed over the network and available via standard interfaces and protocols.

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards [Booth et al., 2004]. A good discussion on the definition of a Web service and Web Services can be found in [Wright, 2005].
Web services specifications suffer from great accumulation of complex specifications that narrow the general acceptability of web services but also form a barrier to interoperability. There is continual overlapping and redundancy while producing different ways of solving the same problems [Wright, 2005].

2.2 WSDL

WSDL is a rigorous, expressive and flexible industry standard to describe web-services. However, it trades clarity for expressiveness and flexibility [Hoschek, 2002]. The verbosity in the example in table A.1 of appendix A is immensely complex and the analysis of the structure of a WSDL document in section 2.3 shows that it has a large degree of redundancy. This makes it very difficult to generate a correct WSDL document manually. The binding element of the WSDL can be considered implementation details and WSDL puts both specification and binding in the same document. UDDI "best practice" recommends that WSDL documents should be in two parts, with binding "including" specification [Newmarch, 2003].

Although being accepted as a standard to describe Web services, WSDL is considered to complex and verbose by the community [Peeters, 2003] [Salz, 2004]. WSDL is considered to be a language that can only be used by specialists in Web-services.

In a WSDL description one can find the definition of operations that the Web-service provides, the interfaces of the operations, the data types of their parameters, the structure of the messages exchanged and the locations of the operations. WSDL is a verbose and complex language difficult to be used by a web service developer who wants to use WSDL to define their services interfaces.

WSDL uses the XML Schema data types to define the messages variables and can define arbitrary data types using XML Schema.

Several attempts to simplify WSDL were made [Hoschek, 2002], but none of them add a practical impact nor followed a different model to describe services.

2.2 Service Implementation Language

The Service Implementation Language (SIL) allows specifying services and its interaction.

It's essentially a language of orchestration but in informatics terms its implementation allows a much lower granularity level than languages like BPEL [Rendeiro, 2008]. It describes services of diverse natures, computer based or not, which means that its implementation can be a computer program or a procedure manual to be executed by a human. In SIL, services are abstract entities that interact by messages.

The language is strongly typified but dynamic and includes rules of structural conformance between types, where in the attribution of a value to a variable that value doesn’t have to include all the components of the data type of the variable.

The USD is extracted from the service implementation and is the part of a service implemented in SIL which describes the interfaces of its operations. A tool should automatically extract the USD from the SIL service each time the service is changed and compiled in order to the USD always reflect the service operations correctly.

The main objectives of Unified Service Description (USD) are: publicise the service description and the data types that it uses; describe the functionality of the service; establish the restrictions of the interface imposed to the clients; allow the validation of a message to a service.

3 ARCHITECTURE

Now we describe our proposal to the service description and discovery of Web-services.

3.1 Service Description

In this section we present our proposal to the description of services.
3.1.1 Objectives

The main objectives of this service description proposal are:

- Reduce the need of business people to know technological details and the need of a programmer to know the business in depth;
- Reduce the creation effort of the Web service programmer in creating and using Web service description documents;
- Increase the flexibility and adaptation to changes;
- Increase the interactivity between the client and the supplier.

3.1.2 Principles

A service description document should be very simple to create and understand, be readable by humans and machines and allowing them to achieve their objectives. Also it should allow one to access the service that is described.

With this in mind, we propose that a service description should only be oriented to:

1. Understand if the service described can do what a client needs.
2. The direct execution of services and its operations.
3. Exclude technology details from the description

In this perspective the service description document don’t have any technology details containing just information about the service structured in a way to allow the execution of the service using parts of the document as it minimizing document conversions.

In figure 3.1, this perspective is shown using WSDL as an example. A WSDL document contains the description of the data elements required to communicate with the supplier’s system and execute the service. However the data elements are mixed up with technological details, such as XML and SOAP tags, which makes it difficult and tedious to create, understand and use.

We propose that the document to describe a service to contain only business transaction details and the technological details should be requested to the supplier’s system supporting the service.

It is important to realise that we are sending messages to execute operations, in a RPC style, instead of a document to be interpreted, in a document style like XML. However, our SIL-I document is oriented to the execution of operations, like RPC, but allows the execution of a complex set of operations, like XML, because all the operations needed to execute the entire service are describe. In a way, we proposed to send documents with all need information, like XML, but with the information oriented to the execution of operations, like RPC. Instead of passive data that must processed by the system to understand what to do with it, the data sent is active in the sense that the system will be able to used in a more direct way than with XML.

Also like XML the data fields can be in any order in the document and a mechanism to verify if the necessary data is provided.

In this perspective, a SIL-I document follows a hierarchy structure as follows: a service has one or more operations and each operation has zero or more variables and a variable can be an input, an output or an exception. Each service and operation can have a URI address to be executed.

3.1.3 SIL-I

Service Implementation Language – Interface (SIL-I) is developed to demonstrate the concepts of this perspective. A SIL-I document follows a hierarchy structure as follows: a service has one or more operations and each operation has zero or more variables and a variable can be an input, an output or an exception. Each service and operation can have an URI address to be executed.

A variable is an arbitrary data structure that represents data from a domain specific business. This means that a variable is a “business” variable and needs to be arbitrary due to be domain specific. These variables are used to describe the capabilities
of the service and should be defined to be directly converted into user input, like a Web form, or into a message to be sent to the supplier’s system. The mechanism to verify the data types of variables is described later.

Now the SIL-I elements specification is presented in table 1:

<table>
<thead>
<tr>
<th>Element</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service {}</td>
<td>Service declaration</td>
</tr>
<tr>
<td>Operation {}</td>
<td>Operation declaration</td>
</tr>
<tr>
<td>Input {}</td>
<td>Input type variable</td>
</tr>
<tr>
<td>Output {}</td>
<td>Output type variable</td>
</tr>
<tr>
<td>Exception {}</td>
<td>Exception type variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Simple variable</td>
</tr>
<tr>
<td>Variable = [v1, v2, ...]</td>
<td>Composed variable</td>
</tr>
<tr>
<td>Variable = (v1, v2, ...)</td>
<td>Composed variable with specific values</td>
</tr>
<tr>
<td>Variable = “Is a variable”</td>
<td>Description sentence</td>
</tr>
<tr>
<td>Variable = URI</td>
<td>Execution address</td>
</tr>
</tbody>
</table>

Table 1: SIL-I specification elements

The hierarchical structure of a SIL-I document is represented using the “{“ and “}” characters. A simple variable is a variable with a primitive data type. A complex variable is a variable composed by a set of simple variables which means that it can have an arbitrary structure and therefore an arbitrary data type. In table 1, “V1” is an empty variable declared, “ID” is a structured composed of variables “name”, “age” and “address” and “Cities” is simple variable that accepts the values “Lisbon”, “Madrid” and “Paris”. Each of the variables in table X have a correspondent Web form to be presented. SIL-I was design thinking in the automatic creation of the user interface, like Web forms, and some elements are defined to simplify the representation of those interfaces. The variables in SIL-I are directly converted into Web forms and messages exchanged between systems. The element “[ ]” used to define complex variables is irrelevant to create Web forms because the set of simple variables will be converted to Text Boxes whether they are declared in a complex variable or declared separately. However, this element is used to define the structure of the message to send to execute the service, a machine readable element, which is the equivalent to the <message> element of a WSDL document. Also this is useful to allow the dynamic change of the Webpage simply by adding or removing variables to be converted in Web forms.

A valid example of a SIL-I document would be:

```java
Service {
    Name = “Airline Agency”
    Description = “Allows to reserve flights”

    Input { Origin City, Destination City, Date, Number of Children }
    Output { Response }

    Reservation = http://www.airline.com/Reserve
    Operation {
        Name = “Check Price”
        Description = “Checks the price of the flight.”
        Output { Price }
        Check = http://www.airline.com/Check
    }
}
```

The Webpage generated from this SIL-I example is illustrated in figure 3.2.

![Figure 3.2: Webpage generated from the SIL-I example.](image)

3.1.4 Web-service development
When creating a Web-service, a programmer can choose between one of two development paths illustrated in figure X.
Figure 3.3: Two development paths the programmer can follow to create the Web-service

As illustrated in figure 3.3, a programmer might start by first implement the service and then, with the aid of a tool, semi-automatically generate the services description documents, the SIL-I documents, the Service Documentation, the Data Types document and the Webpage. Alternatively, the programmer might start to define the SIL-I document and use a tool to generate the service implementation stubs and the remaining documents. All the documents in figure 3.3 except the Service Code and the Code stubs are used in the process of describing a service and we will now explain their content.

The Data Types document lists the variables of a service and their data types. A valid example of a SIL-I document would be:

Service “FlightReservation”
Input
 Origin City : String
 Destination City : String
 Date : Date
 Number of Children : Integer
Operation “CheckPrice”
Output
 Price : Integer

The Service Documentation document contains a human readable description of what the service do, its non-functional properties, and restrictions regarding the supplier of the service. This document is useful when a programmer need more detailed information about the service that doesn’t make sense to be described in the SIL-I document.

The Webpage represents the service description seen by the user of the service in its Web-forms, which describes the information necessary to execute the service.

The elements of these three documents can be semi-automatically generated by a tool with the SIL-I document but ultimately the programmer may need to complete or correct them.

3.1.5 Technological Details

The data types of variables are mandatory because the information provided by the client through the Web form must be converted to its data types in order to create a message that the supplier system understand allowing it execute the service. In particular, the data sent in the message must have the expected data format and type.

To solve this point, a mechanism of data type request must be provided so that the data type of a variable, which can be composed of arbitrary fields each one with a data type, can be requested in run time to allow the creation a message to be sent in order to execute the service. When a request to know the type of the variables describe in the SIL-I documents is presented to the supplier’s system it replies the data types associated with those variables.

Consider the scenario where a service implementation has a complex variable “Identification” composed of three variables: Name, Age and Address each one with its data type. That service description is generated and published on the Web. After that the service implementation changes and the variable “Identification” changes as shown.

Now a client found that service description and intents to provide the necessary data and request the execution of the service, by clicking a button in the Web form. The Web page is presented in the browser to the client with the respective Web form. When the client requests the execution of the service a message is created and sent to the service system in order to execute the service. If the service description had the technology details to create the message, it would create it and send it to the service system. However, the system would not recognize the message because the characteristics of the variable “Identification” have changed.

What will happen is that when the requests the execution of the service, that invocation will send a message requesting the data types of the variables being used to execute the service, in the variable “Identification” composed of the variables “Name”, “Age” and “Address”. The system will receive that request, recognize the structural differences of the
variable “Identification” and return a message with the details of the new version of the variable so that the browser can recreate a message with the correct format and the service system will be able to recognize it.

In figure 3.4, is illustrated the behaviour of the service system when a client tries to use an old version of the service description to execute the service. In 1) the data types of the variable “Identification” are requested to the service system. In 2) the service system recognizes that the variable “Identification” to which the client refers has changed in its structure and data types. In this case, the data types of variable “Age” changed from Integer to Date, the variable “Address” is no longer necessary to execute the service and a new variable “BI” is needed. When the service is implemented, one of the generated documents has all the service variables, its data types and labels to enable the client to request that information and a simple comparison can be done. The service system in 3) will load the most recent webpage which will reflect the correct structure and data type of the variable “Identification” allowing that in 4) the user re-enter the data and request the service execution with success.

This mechanism allowed several adaptations to a situation that otherwise would result in the failure of the service execution:
- If the variable changed in its structure the service system will recognize it inform the requester of that fact providing the correct data types;
- The order of the variables in the message is irrelevant, the system searches for their names and their just have to be declared and have the same structure and data types;
- If the system receives a message with a set of variables and in that set are the variables it needs to execute the service than it will ignore the other variables.

The information about the execution of the service is independent of the communication protocols used so to execute a service is provided in the SIL-I document an URI. This is used to identify resources on the Web abstracting the naming schemes and access methods. (Berners-Lee et al., 2005).

To generate the Webpage from the SIL-I document, its elements have to be converted to Web-forms allowing the user to provide the information necessary to execute the service. When the programmer generates the Web page also has to manually create a presentation document such as a Cascading Style Sheets (CSS) document to improve the presentation of the Webpage. Following is the conversion table mapping SIL-I elements into Web form elements.

<table>
<thead>
<tr>
<th>SIL-I element</th>
<th>Web form element</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name = “Airline Agency”</td>
<td>Airline Agency</td>
<td>Caption</td>
</tr>
<tr>
<td>Destination</td>
<td>Destination</td>
<td>Text Box</td>
</tr>
<tr>
<td>Check Price = [<a href="http://Airline/CheckPrice">http://Airline/CheckPrice</a>]</td>
<td>Check Price</td>
<td>Button</td>
</tr>
<tr>
<td>Cities = [Lisbon, Madrid, Paris]</td>
<td>List Box</td>
<td>Text Box, Combo Box</td>
</tr>
</tbody>
</table>

In figure 3.5 are the basic elements used to exemplify the proposal. Although it’s unusual to create complicated Web forms, the model supports variables with arbitrary structure. This means that one can declare complex variables inside complex variables leading to mixture of combine combo boxes and text boxes.

### 3.1.6 Compatibility with WSDL

This proposal presents an alternative to the service description layer of the W3C Web Services Stack.
However, the existing proposals have their value and more importantly they are already established as an accepted standard with much technological support already developed.

It’s important that our proposal explore ways to ensure the compatibility between SIL-I and other standards like WSDL which means compatibility with XML. In this section are discussed mechanisms to semi-automatically convert SIL-I documents to WSDL and vice versa.

3.2 Service Discovery

This architecture serves the purpose of allowing the client to find SIL-I documents just like a search engine. The architecture follows the index approach and fits better with the service description proposal than the other approaches in [Booth et al., 2004].

3.2.1 Concepts

The first concept to be defined is the portal. This entity represents a system that does the functions of a search engine allowing services to be discovered. A robot is a software program that crawls the Internet searching for service description documents and is used by the portal to gather them. Another concept is the user. This entity represents a human interacting in the architecture to search, execute and compose services. A supplier is a user who has services and wishes to make them available by creating their descriptions and publishing them. Also he can search for services with the objective of creating a new composite service. A client is a user who searches and executes services.

3.2.2 Architecture Definition

There are two main interactions in the architecture: first, the user uses the portal to search for Web-services, the Webpage or the SIL-I document; second, the client interacts with the supplier through the Webpage of the service being provided.

As one can see in figure 3.6, much of the information in a WSDL document can be automatically used to create the proposed documents. The underlined element “Binding” is not used because contains information about SOAP elements that are abstracted in this proposal. Also the underlined element “Supplier documentation” refers to supplier documentation and non-functional parameters of the service that usually are not detailed in the WSDL document. However that information can be described in the WSDL document through its element <documentation>.

The mapping from the SIL-I, Data types and Service Documentation documents to WSDL document is more limited because many implementation details are not available. Contrary, much of the information necessary to create the proposed documents can be extracted from a WSDL document making it easy to convert existing Web service descriptions to the proposal presented.
found by the robots in the Internet and interact with the client.

Figure 3.8: Second phase of interaction between the entities of the architecture

After interacting with the portal and discover an appropriated service, the client will be forwarded to the service Web page where he can interact and execute the service as shown in figure 3.8.

The Web pages are automatically generated by the supplier’s system each time the service changes and it’s recompiled.

3.2.3 Architecture Benefits
In this section, we briefly summarize the goals of this discovery architecture as follows:

1. More comprehensive coverage
The Web crawler component visits not only public/private UDDI registries, but also the Web to find service descriptions. It’s better to the service requester because he will have access to more service suppliers then if he searched in a centralized registry which only has the service description of the supplies that register in that registry.

2. Less work to the service provider
In the proposed architecture, the service provider just has to create the service description documents and publish them in he’s web server. These documents will be automatically gathered by the robots that search for service description.

3. More up-to-date repository
When a service description document is changed by the supplier, those changes would, in a short period of time, be automatically reflected in the index because the robots will understand that it changed and replace it the index. A centralized UDDI registry always suffers from the possibility of being obsolete, and keeping it up-to-date has to be done by careful planning and tedious programming work using its publish APIs.

4 IMPLEMENTATION

4.1 Unified Service Description
SIL-I is a coherent specification developed in this thesis that can be used, at some extent, in the SIL-I description document. However, SIL-I isn’t complete and must be developed in the future in order to be used to implement the SIL-I documents. To implement the concept explained with SIL-I we propose the Unified Service Description (USD).

The USD is a document that is generated from the source files of a service implemented in SIL. USD is our proposal to implement the concepts of SIL-I for web-services implemented in SIL.

4.2 Systems Communication
Our proposal is independent of the protocol used to codify and transport messages. The execution of services through the exchange of messages should be done using the SOAP protocol or an architectural style like REST using just HTTP to communicate.

REST is a simple way to create web-services to solve simple problems which doesn’t have complex security, transactional or management restrictions. Since only relies on HTTP few problems involving these restrictions can be solved. HTTPS can be used to provide basic security but mechanisms to deal with more complex restrictions would have to be programmed while implementing the service.

The Unified Service Message Protocol (USMP) can be used to support the communication mechanisms referred in this thesis and can be used as an implementation of the messaging details abstracted in our proposal [Sacramento, 2008]

4.3 Webpage Generation
In section 3.2 the interactivity between in the Webpage the client uses to interact with the service were specified as well the mechanisms used. To implement these interactivity one can use technologies like GWT (see http://code.google.com/webtoolkit/) and Ajax to enable the dynamism of the web-forms.
There is a layer of JAVA classes available to create and manage JavaScript code to control the behaviour of the web-forms. Webpage is presented to the client and the client accesses the service through its Web-forms. The Webpage can have business logic regarding the behaviour of its Web-forms in JavaScript allowing the interaction between the client and the service necessary in a service transaction. This interaction can be implemented in a simple way using GWT which converts the behaviour programmed in JAVA into Javascript to be used in the Webpage.

The presentation layer of our proposal to the service description of services can be developed using GWT.

5 CONCLUSIONS AND FUTURE WORK

This investigation addressed the problem of describe and discover Web-services in a simple, lightweight and dynamic way. It showed that at least for Web-services with few requirements for security, transactions and management this model can be used and provide the proposed benefits. Nonetheless, these extra requirements can be introduced at the application level even if the proposed model doesn’t take them in account.

The core principles and requirements of our proposal were presented and we demonstrated that this proposal is worthy of being developed in future works. Further analysis and development of its components as well as adaptations to support more complex mechanisms, respecting the core principles and requirements, can make this proposal eligible to be used to enable business transactions through the Web and the participation of all the society in the creation and proliferation of new services as a way to interact with each other.

To finalize we present the directions for future research.

First, the proposed SIL-I specification is not complete and lacks expressiveness. SIL-I was developed with the purpose of exemplifying how would be the service description documents following the principles in section 3.1.1. SIL-I specification should be enhanced with more expressivity to be able to solve more complex problems.

Second, one of the drawbacks of this proposal is that requires much communication between systems and there is the need to deal with ways to increase the performance. Also the communication mechanisms and the presentation aspects to support the service description proposal should be enhanced and implemented.

Third, the development of the integration between the platform proposal in [Sacramento, 2008] and the service description and discovery architecture proposed in this thesis should be considered for future work.

Fourth, an integrated IDE for the development of the SIL-I documents, showed in section 3.1.3, should be developed to help increasing the programmer’s efficiency.

Finally, the investigation on how to introduce semantic information using Semantic Web technologies in the SIL-I following the same principles used to define SIL-I would be valuable.

REFERENCES

Peeters J., "WSDL Tales From the Trenches, Part 3", August 2003
Salz, R., "WSDL 2: Just Say No", November 2004