Uma Arquitectura Web para Serviços Web

Sérgio Nunes[1] e Gabriel David[2]

[1] FLUP
[2] FEUP/INESC-Porto
Universidade do Porto
Porto, Portugal

Resumo A evolução dos Serviços Web foi, nos últimos tempos, controlada pelas principais empresas da indústria dos computadores. O conceito de Serviço Web (SW) está associado à disponibilização de funcionalidades através de interfaces programáveis via Internet. Na definição apresentada pelo W3C, é referido o uso de URL, HTTP e XML em "conjunto com outras normas relacionadas com a Web". No entanto, apesar do recurso às designadas tecnologias Web, o estilo arquitectural proposto para os SW não segue aquele desenvolvido para a WWW.
Neste artigo, evidenciam-se estas diferenças e descreve-se uma nova abordagem para o desenho de SW, com base no estilo arquitectural REST. Este estilo está na base do desenho da WWW e apresenta qualidades importantes para sistemas distribuídos de grande dimensão. No final, é apresentado um exemplo concreto de um SW, desenvolvido com base nestes conceitos.

1 Introdução

Com o desenvolvimento da Internet e das tecnologias paralelas, surgiram novas abordagens ao problema da interoperabilidade entre sistemas de informação. A ubiquidade da "rede das redes", "indiferente"a fronteiras empresariais ou tecnológicas, surge como uma alternativa para a camada de comunicação. Por outro lado, a evolução tecnológica e a adopção generalizada da eXtensible Markup Language (**XML**), como linguagem de estruturação de documentos, veio permitir o desenvolvimento de novas arquitecturas, actualmente designadas de Serviços Web (**SW**).

Os Serviços Web representam uma tecnologia emergente que tem sido alvo de atenção por parte de toda a indústria dos computadores [1]. Empresas como a Microsoft, IBM e Sun têm apostado e investido neste conceito. As primeiras iniciativas remontam a 1998, com o trabalho desenvolvido pela Microsoft em parceria com a Userland Software, que resultou numa primeira publicação do protocolo SOAP [2]. A partir de 2001, o desenvolvimento de especificações passou a ser realizado em consórcios empresariais, como o W3C e a OASIS.

De uma forma simplificada, os Serviços Web procuram facilitar a integração de aplicações distribuídas utilizando XML sobre a World Wide Web (**Web**). No entanto, de um ponto de vista prático, as estratégias para a implementação deste

conceito divergem, resultando nalguma ambiguidade a este nível. Não existe uma noção clara do que é a implementação de um Serviço Web. Implementações com diferenças importantes ao nível da arquitectura e das tecnologias utilizadas têm sido catalogadas como "Serviços Web".

Um outro aspecto que importa referir é o facto de, associado ao trabalho desenvolvido pelos consórcios empresariais, terem sido produzidas inúmeras especificações. Em finais de 2004, existem mais de 30 especificações (p.e.: WS-Federation, WSDL, BPEL4WS, WS-I Basic Profile) relativas à visão dos Serviços Web proposta pelos principais consórcios! Este aspecto tem contribuído para um maior afastamento entre os programadores, que desenvolvem as implementações reais, e as especificações propostas pelas grandes empresas.

Neste artigo, é feita uma comparação entre a visão tradicional dos Serviços Web e uma outra que tem por base o modelo arquitectural REST. É feita uma comparação em termos da arquitectura das duas abordagens. Uma arquitectura deste tipo determina como os elementos do sistema são identificados e distribuídos, como interagem por forma a constituir um sistema, qual a granularidade da comunicação necessária para interacção e quais os protocolos de interface utilizados para comunicação [3].

2 Serviços Web

A Figura 1 representa o modelo apresentado pelo W3C [4] relativo ao processo geral para estabelecimento de um Serviço Web.

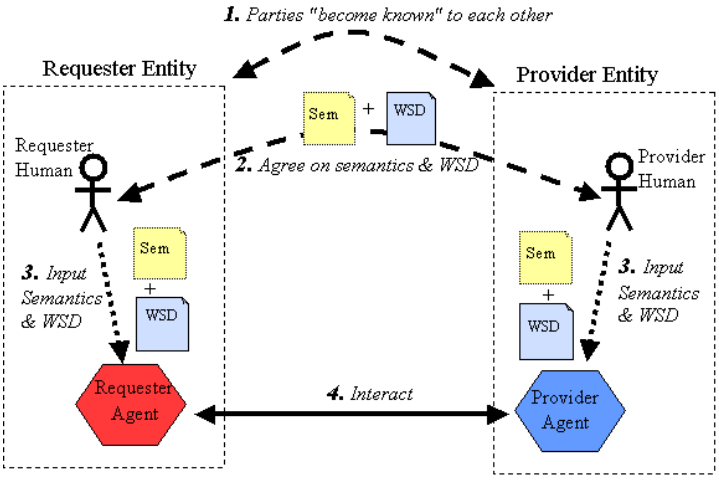

Figura 1. Processo geral para estabelecer um Serviço Web [4]

É possível identificar quatro fases genéricas necessárias [4]: (1) o cliente e o servidor tornam-se conhecidos um do outro (ou pelo menos um conhece o outro); (2) o cliente e o servidor estabelecem um acordo em relação à descrição e semântica do serviço que vai gerir a interacção entre os agentes; (3) são desenvolvidas implementações de acordo com a descrição e semântica acordadas; (4) o cliente e o servidor trocam mensagens que concretizam a interacção entre as partes.

Considerando os objectivos deste trabalho, observa-se com maior detalhe a fase de interacção. De acordo com a visão tradicional para os Serviços Web, a fase de interacção é implementada com recurso à troca de mensagens SOAP. SOAP [5] é uma linguagem de anotação, baseada em XML Schema [6], para a descrição de protocolos de comunicação. Permite a definição de interfaces públicas e dos detalhes associados à invocação e resposta. Em particular, permite a definição dos tipos e estruturas de dados a utilizar. O SOAP pode ser definido como um "*protocol framework*".

A Figura 2 apresenta uma mensagem SOAP correspondente à invocação de um procedimento.

```
<?xml version="1.0" ?>
<soap:Envelope xmlns:soap=" http://www.w3.org/2003/05/soap-envelope">
    <soap:Body>
        <u:getStudentInfo xmlns:u="http://univ.example.com">
            <u:code>123456789</u:code>
        </u:getStudentInfo>
    </soap:Body>
</soap:Envelope>
```

Figura 2. Mensagem SOAP para a invocação do método getStudentInfo com o argumento 123456789

Na Figura 3 apresenta-se a resposta ao pedido. O código aqui apresentado é uma versão simplificada daquele que seria necessário num cenário real.

```
<?xml version="1.0" ?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
    <soap:Body>
        <u:getStudentInfoResponse xmlns:u="http://univ.example.com">
            <u:code>123456789</u:code>
            <u:name>João Silva</u:name>
            <u:age>19</u:age>
        </u:getStudentInfoResponse>
    </soap:Body>
</soap:Envelope>
```

Figura 3. Mensagem SOAP de resposta ao pedido apresentado na Figura 2

O SOAP não define, nem está limitado, a um mecanismo de transporte específico. A especificação permite a integração com diversos protocolos para a transferência das mensagens, por exemplo: HTTP, SMTP e MQSeries. Actualmente, a generalidade das implementações utilizam o protocolo HTTP, em particular o método POST, para a troca de mensagens SOAP. Este cenário é ilustrado na Figura 4.

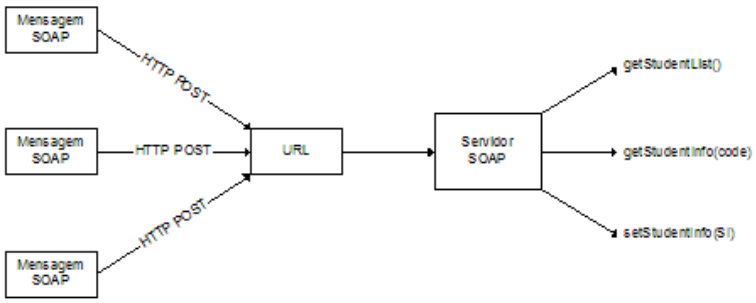

Figura 4. Arquitectura de um Serviço Web tradicional (baseado em [7])

Não é um requisito da especificação a utilização do mesmo URL para o envio de todas as mensagens. No entanto, esta é uma estratégia comum entre os promotores do SOAP. A generalidade dos IDE e das bibliotecas disponíveis geram código segundo esta arquitectura.

3 Representational State Transfer

Representational State Transfer, ou simplesmente REST, é um termo definido por Roy Fielding [8] para descrever um estilo arquitectural de sistemas de informação distribuídos.

A primeira edição do REST, originalmente designada por "HTTP Object Model", foi desenvolvida entre 1994 e 1995, como um meio para transmitir os conceitos que se encontram na base da Web durante o desenvolvimento do protocolo HTTP. Fielding [3] afirma que "*the name "Representational State Transfer"is intended to evoke an image of how a well-designed Web application behaves: a network of Web pages forms a virtual state machine, allowing a user to progress through the application by selecting a link or submitting a short data-entry form, with each action resulting in a transition to the next state of the application by transferring a representation of that state to the user.*".

Resumindo, REST representa um modelo de como a Web deve funcionar. A existência deste modelo permite identificar áreas onde os protocolos existentes falham, comparar soluções alternativas e assegurar que novas extensões não violam os princípios nucleares que contribuíram para o sucesso da Web. O REST

não é uma norma nem uma especificação, mas sim um conjunto de restrições que induzem determinadas propriedades nos sistemas.

Uma descrição detalhada do modelo REST está fora do âmbito deste artigo, pelo que se referem apenas os aspectos mais relevantes para motivar a discussão a propósito dos Serviços Web. Uma apresentação exaustiva do estilo arquitectural REST pode ser encontrada na proposta original publicada por Roy Fielding [8].

O modelo REST utiliza um conjunto de interfaces genéricas para promover interacções sem estado (*stateless*) através da transferência de representações de recursos, em vez de operar directamente sobre esses recursos. O conceito de recurso é a principal abstracção deste modelo. Entre as restrições definidas pelo REST, destacam-se três e refere-se a aplicação no caso concreto da Web:

- **Identificação global.** Todos os recursos são identificados através do mesmo mecanismo global. Independentemente do contexto de aplicação, tecnologia ou implementação concreta, cada recurso disponível na Web é identificado utilizando um URL [9].
- **Interfaces uniformes.** A interacção entre os agentes é feita com recurso ao protocolo HTTP [10], utilizando um conjunto de métodos pré-definidos: GET, POST, PUT e DELETE.
- **Interacções** *stateless***.** O estado concreto de uma aplicação Web é mantido com base no estado de um conjunto de recursos. O servidor conhece o estado dos seus recursos mas não mantém informação sobre as sessões dos clientes.

O caso particular das interfaces uniformes pode ser comparado à abstracção que se verifica na linguagem SQL com as operações SELECT, INSERT, UPDATE e DELETE. O recurso a um conjunto pré-definido de operações, permite que qualquer interveniente na comunicação entenda a operação e aja de acordo. Existe uma semântica estabelecida ao nível da infraestrutura. No caso da Web, intermediários como as *caches* e *proxies*, podem agir de acordo com a operação executada e contribuir para a escalabilidade de toda a infraestrutura, sem afectar a definição da respectiva operação abstracta. Por exemplo, o método GET, idempotente por definição, pode ser servido por *caches* intermédias sem intervenção do servidor responsável pelo recurso. Por outro lado, os métodos POST, PUT e DELETE só podem ser processados pelo servidor uma vez que alteram o estado do recurso.

Assim, quando um agente do sistema encontra um URL ou um método HTTP sabe o que fazer para o processar. O facto de existirem acordos públicos, partilhados por todos os interveniente, contribui de forma positiva para a escalabilidade e heterogeneidade do sistema global.

A Web é o maior sistema de informação distribuído em funcionamento. O REST procura identificar e sistematizar as características arquitecturais que permitiram o crescimento exponencial verificado na última década.

4 Serviços Web segundo o modelo REST

Observando as especificações actuais associadas aos Serviços Web [11] e, em particular, os modelos de programação induzidos pelos IDE [12], é possível iden-

tificar um modelo diferente daquele seguido na implementação da Web. Nesta secção, apresenta-se uma arquitectura alternativa para o desenho de Serviços Web tendo por base os pressupostos estabelecidos com o modelo REST.

Como se viu na Secção 2, segundo a abordagem tradicional ao desenvolvimento de Serviços Web, os recursos estão escondidos atrás de métodos que podem ser invocados. Num Serviço Web compatível com a arquitectura REST os recursos a disponibilizar devem ser expostos e ter uma identificação global.

No cenário utilizado como exemplo, pretende-se disponibilizar informação sobre alunos. Os recursos correspondem a documentos XML com a informação sobre os alunos e são disponibilizados com a estrutura apresentada na Figura 5. Cada URL identifica de forma única a informação sobre um estudante.

```
...
http://univ.example.com/studentInfo/123456787
http://univ.example.com/studentInfo/123456788
http://univ.example.com/studentInfo/123456789
...
```

Figura 5. Recursos expostos de um Serviço Web com uma arquitectura REST

Depois de expostos os recursos, a interacção é realizada utilizando as operações genéricas do protocolo HTTP. Assim, para obter a representação da informação relativa a um estudante, utiliza-se o método GET sobre o URL concreto. Na Figura 6 ilustra-se essa situação.

```
GET http://univ.example.com/studentInfo/123456789
```

```
<?xml version="1.0" ?>
<u:StudentInfo xmlns:u="http://univ.example.com">
    <u:code>123456789</u:code>
    <u:name>João Silva</u:name>
    <u:age>19</u:age>
</u:StudentInfo>
```

Figura 6. Pedido da representação de um recurso num Serviço Web com uma arquitectura REST e respectiva resposta

Os restantes métodos do protocolo HTTP permitem: pedir a criação de uma representação para um novo recurso (POST); actualizar a representação de um recurso existente (PUT); apagar um recurso (DELETE). Enquanto que os métodos DELETE e PUT operam directamente sobre um recurso identificado por um URL, o método POST é aplicado sobre um URL disponibilizado pelo servidor para a criação de novos recursos (ex: http://univ.example.com/studentInfo).

Apesar de não serem relevantes para esta discussão, é importante referir que na especificação HTTP são definidos mais métodos para além dos quatro aqui apresentados. De notar que, na navegação Web, apenas os métodos GET e POST são implementados pelos navegadores.

Uma outra característica que advém da adopção do modelo REST é a possibilidade que o cliente tem de navegar entre os recursos. Utilizando a especificação XLink [13] e devido à existência de um mecanismo de identificação global (URL), são estabelecidas ligações entre os recursos disponíveis. É da responsabilidade do cliente navegar de recurso em recurso, reunindo a informação que necessita ou activando os estados que pretende. Desta forma, através da utilização de referências, é possível evitar a repetição de informação e todos os problemas associados a esta opção. Por outro lado, é possível implementar serviços mais sofisticados envolvendo a partilha de recursos entre aplicações diferentes.

Na Figura 7, ilustra-se o resultado de um pedido da lista de estudantes disponíveis. O pedido é enviado utilizando o método GET num URL disponibilizado pelo servidor. Reparar no recurso à especificação XLink para representar as ligações para a representação da informação de cada estudante. Com esta informação, o cliente pode efectuar operações sobre os recursos "descobertos".

```
GET http://univ.example.com/studentInfo

<?xml version="1.0" ?>
<u:StudentInfos  xmlns:u="http://univ.example.com"
                 xmlns:xlink="http://www.w3.org/1999/xlink">
     <u:StudentInfo id="123456787" xlink:href="http://univ.example.com/StudentInfo/123456787"/>
     <u:StudentInfo id="123456788" xlink:href="http://univ.example.com/StudentInfo/123456788"/>
     <u:StudentInfo id="123456789" xlink:href="http://univ.example.com/StudentInfo/123456789"/>
     ...
</u:StudentInfos>
```

Figura 7. Pedido de uma lista de recursos num Serviço Web com uma arquitectura REST e respectiva resposta

Como se procurou ilustrar, com recurso a tecnologias estabelecidas e aptas a serem utilizadas (URL, HTTP, XML Schema e XLink), é possível desenvolver Serviços Web com uma arquitectura compatível com o modelo REST. Nesta secção foi feita uma apresentação resumida desta abordagem alternativa. Mais informações sobre o desenvolvimento de Serviços Web compatíveis com o modelo REST podem ser encontradas em [14], [15], [16] e [17].

5 Discussão e Análise dos Resultados

Na definição apresentada pelo W3C para Serviços Web [4], refere-se o "[...] recurso a mensagens SOAP, tipicamente transmitidas sobre HTTP e codificadas em XML, em conjunção com outras especificações Web.". No entanto, para além das tecnologias concretas, um aspecto fundamental da Web é a arquitectura que está na base deste sistema distribuído. Durante o desenvolvimento das principais

tecnologias (nomeadamente URL e HTTP), os conceitos que regem a arquitectura base da Web estiveram presentes de forma tácita. Em 2000, Roy Fielding [8] sistematizou e publicou esta arquitectura com a designação REST.

De uma forma resumida, REST representa um modelo de como a Web deve funcionar de forma a garantir os requisitos inicialmente estabelecidos, em particular as propriedades de escalabilidade e heterogeneidade. No entanto, o trabalho desenvolvido pelos principais consórcios e empresas, no âmbito dos Serviços Web, não tem tido em conta as restrições definidas pelo estilo arquitectural REST.

Na realidade, as especificações actuais para os Serviços Web recorrem às principais tecnologias Web de forma preferencial mas opcional, permitindo a integração com outros protocolos, como o SMTP. Uma designação mais fiel da situação real seria a de "*Serviços Internet*".

É também importante referir que, apenas na segunda revisão da especificação SOAP (versão 1.2), foi possível passar a utilizar a linguagem numa forma consistente com a arquitectura REST. A versão 1.1 do SOAP, a especificação nuclear dos Serviços Web, não era compatível com o modelo REST [4].

A estratégia actual representa uma evolução das abordagens já existentes baseadas no conceito de *Remote Procedure Call* (RPC). A principal diferença introduzida pelo REST é a uniformização da interface de comunicação [8]. Ao contrário do que acontece com o SOAP, o modelo REST não permite a definição de interfaces específicas.

Como resultado, no cenário tradicional, temos duas camadas específicas em cada sistema distribuído: (1) protocolo de interacção (SOAP) e (2) representação dos dados (XML). Num modelo REST, a camada de interacção é normalizada. Assim, apenas a representação dos dados é específica em cada implementação. Com a Figura 8 procura-se ilustrar esta diferença.

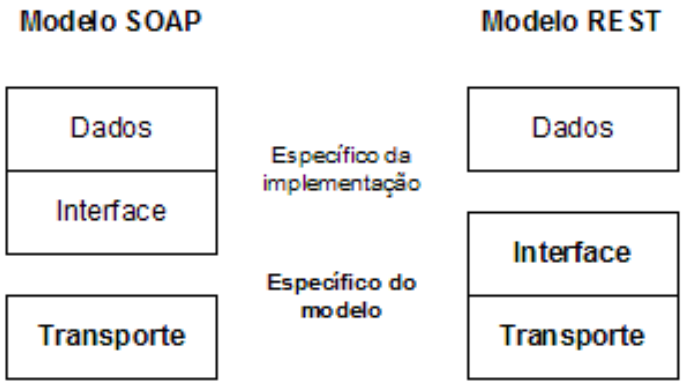

Figura 8. Organização das camadas no modelo SOAP e no modelo REST (adaptado de [20])

No caso do SOAP, as interfaces não são definidas previamente. Para cada implementação é possível definir os modelos de dados e as interfaces a utilizar. Em particular, é possível definir os mecanismos de endereçamento e os métodos disponíveis para invocação. No modelo REST, a interface faz parte da infraestrutura e é estática. Isto é, as implementações concretas apenas diferem na camada de dados.

Esta diferença permite que o SOAP funcione sobre qualquer protocolo, uma vez que os detalhes da invocação estão dentro da mensagem e são independentes da camada inferior. Por outro lado, o modelo REST implica a utilização de um protocolo comum entre todos os intervenientes, incluindo os agentes intermédios (*caches*, *proxies*, *firewalls*). Com esta abordagem, é possível eliminar uma camada de complexidade extra no desenvolvimento das soluções. Os serviços recorrem à interface disponível, neste caso HTTP, evitando a especificação de soluções próprias. Desta forma, reduzem-se as barreiras entre serviços diferentes, aumentando o grau de interoperabilidade.

Por exemplo, numa abordagem SOAP os métodos getStudent, getFaculty, getCourse apenas têm significado semântico num contexto limitado, o dos sistemas que os implementam. Numa abordagem REST, o método GET mantém o mesmo significado apesar de poder ser aplicado a diferentes recursos (estudantes, docentes, disciplinas). A interoperabilidade com outros sistemas está presente por natureza.

Um outro aspecto importante para esta discussão tem a ver com a complexidade das especificações propostas pelos consórcios empresariais. Este facto tem afastado as pequenas empresas e programadores da designada via tradicional. Em relação a este aspecto, a abordagem REST apresenta-se como uma solução mais simples. Por um lado, promove tecnologias testadas e disponíveis a baixo custo, por outro lado baseia-se num paradigma bem conhecido. A propósito desta questão, um aspecto importante referido por Tim O'Reilly [18], é o facto da Amazon disponibilizar serviços Web em ambos os estilos (REST e SOAP) [19] e 85% da utilização ser através da interface REST.

6 Conclusões

Mantendo-se fieis à arquitectura REST, as tecnologias Web (URI e HTTP) foram capazes de se adaptar ao crescimento exponencial que se verificou durante os últimos 10 anos de existência da Web. Outras abordagens para o suporte de sistemas distribuídos, bem sucedidas na integração em redes locais (CORBA, DCOM, RMI), tiveram um sucesso muito limitado quando transpostas para sistemas de grande dimensão, à escala da Internet.

Com este trabalho, procurou-se contribuir para uma reflexão crítica a propósito das recentes estratégias no âmbito da interoperabilidade entre sistemas distribuídos. O paradigma baseado no modelo REST tem por base uma arquitectura e um conjunto de tecnologias que estão na origem do sucesso de um dos maiores sistemas distribuídos desenvolvidos. Confrontando este modelo com aquele que tem vindo a ser proposto pelas grandes empresas e consórcios, as diferenças são

evidentes. No entanto, é ainda cedo para afirmar a superioridade de uma ou outra abordagem para a implementação de Serviços Web.

Neste contexto, importa referir que as duas abordagens não são exclusivas. Desenvolvimentos recentes apontam para uma aproximação entre as duas. Por um lado, recorrer à utilização dos paradigmas introduzidos com o modelo REST, em particular as interfaces genéricas e a exposição dos recursos. Por outro lado, aproveitar o trabalho desenvolvido para o SOAP nas áreas da segurança e orquestração de serviços, entre outras.

Referências

1. Wong, W., Kane, M., Ricciuti, M.: The new buzz - "Web services"try to rise above high-tech din CNET News.Com. (2000) `http://news.com.com/2009-1017-275484.html?legacy=cnet&tag=tp_pr` [02-12-2004]
2. Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., Winer, D.: Simple Object Access Protocol (SOAP) 1.1. W3C Note, World Wide Web Consortium. (2000) `http://www.w3.org/TR/SOAP` [02-12-2004]
3. Fielding, R. T., Taylor, R. N.: Principle Design of the Modern Web Architecture ACM Transactions on Internet Technology. 2:2 (2002) 115–150
4. Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.: Web Services Architecture W3C Working Group Note, World Wide Web Consortium. (2004) `http://www.w3.org/TR/ws-arch/` [24-11-2004]
5. Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., Nielsen, H. F.: SOAP Version 1.2 Part 1: Messaging Framework W3C Recommendation, World Wide Web Consortium. (2003) `http://www.w3.org/TR/soap12-part1/` [02-12-2004]
6. Fallside, D. C.: XML Schema Part 0: Primer W3C Recommendation, World Wide Web Consortium. (2001) `http://www.w3.org/TR/xmlschema-0/` [16-05-2003]
7. Costello, R. L.: REST (Representational State Transfer). (2002) `http://www.xfront.com/REST.html` [02-12-2004]
8. Fielding, R. T.: Architectural styles and the design of networked-based software architectures Dissertação de Doutoramento Dept. of Information and Computer Science, University of California, Irvine (2000)
9. Berners-Lee, T., Fielding, R. T., Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax RFC 2396, Internet Engineering Task Force. (1998) `http://www.ietf.org/rfc/rfc2396.txt` [26-05-2004]
10. Fielding, R. T., Gettys, J., Mogul, J. C., Nielsen, H. F., Masinter, L., Leach, P. J., Berners-Lee, T.: Hypertext Transfer Protocol - HTTP/1.1 RCF 2616, Internet Engineering Task Force. (1999) `http://www.ietf.org/rfc/rfc2616.txt` [26-05-2004]
11. Cabrera, L. F., Kurt, C., Box, D.: An Introduction to the Web Services Architecture and Its Specifications White Paper, Microsoft Developer Network. (2004) `http://msdn.microsoft.com/library/en-us/dnwebsrv/html/introwsa.asp` [02-12-2004]
12. Lopes, C. J. F., Ramalho, J. C.: Web Services: Metodologias de Desenvolvimento Actas da XATA 2004 - XML, Aplicações e Tecnologias Associadas. Porto, Portugal. (2004)
13. DeRose, S., Maler, E., Orchard, D.: XML Linking Language (XLink) Version 1.0 W3C Recommendation, World Wide Web Consortium. (2001) `http://www.w3.org/TR/xlink/` [02-12-2004]

14. Prescod, P.: Second Generation Web Services XML.com (2002) http://www.xml.com/pub/a/ws/2002/02/06/rest.html [20-11-2004]
15. Prescod, P.: REST and the Real World XML.com (2002) http://www.xml.com/pub/a/ws/2002/02/20/rest.html [20-11-2004]
16. Baker, M.: RESTWiki. (2004) http://rest.blueoxen.net [20-11-2004]
17. Prescod, P.: Paul's REST Resources. (2004) http://www.prescod.net/rest/ [02-12-2004]
18. O'Reilly, T.: REST vs. SOAP at Amazon. (2003) http://www.oreillynet.com/pub/wlg/3005 [02-12-2004]
19. Amazon.com: Amazon.com Web Services (2004) http://www.amazon.com/webservices [02-12-2004]
20. Baker, M.: Protocol independence. (2004) http://www.markbaker.ca/2002/09/Blog/2004/10/29#2004-10-protocols [20-01-2005]