UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA

INF01151 – Sistemas Operacionais II N
Grupo 2 - Webservices e CORBA

1. Qual a vantagem da CORBA em relação à RMI? (samuel)

Diferente da RMI, CORBA é independente da linguagem, de modo que aplicações escritas em linguagens de programação diferentes podem operar entre si pelo acesso a um núcleo CORBA comum.
fonte: Deitel

2. Por que um usuário escolheria RMI em vez de CORBA? (samuel)

Se o usuário estiver trabalhando estritamente em Java, a independência de CORBA em relação à linguagem será desnecessária. Usar CORBA também requer que o usuário aprenda IDL, para que cliente e servidor possam se comunicar adequadamente.
fonte: Deitel

3. O que é um WebService? (Maitê e Eduardo)

Um WebService é uma solução que permite a comunicação entre sistemas desenvolvidos em plataformas, linguagens e épocas diferentes. Cada sistema opera na sua linguagem, e depois toda a informação a ser compartilhada é traduzida na língua "oficial", o formato XML. As mensagens então são enviadas através da Web para a(s) outra(s) aplicação(ões), para que efetue(m) tarefas simples ou complexas. O envio é feito através de um protocolo de transporte, normalmente o HTTP ou HTTPS (não é especificado o protocolo).

4. O que é CORBA? (Alessandro e Samuel)

Arquitetura Intermediária Comum de Requisição a Objetos (do inglês Commom Object Request Broker Architecture), é um padrão aberto de arquitetura para sistemas distribuídos, seu foco está em prover a interoperabilidade entre programas em sistemas heterogêneos (e também homogêneos). Para que possa prover tal funcionalidade, CORBA precisa ser independente de linguagem e sistema, assim ele é desenvolvido através de uma IDL (***ver questão 13***) onde o desenvolvedor define os procedimentos que seu objeto tem permissão de utilizar (sua interface). Uma visão mais ampla da arquitetura pode ser vista neste diagrama: http://goo.gl/FCokz.

5. Como outros sistemas podem interagir com os web services? (Luiza)

Um web service fornece uma interface de serviço que permite a interoperabilidade com outros sistemas, os quais podem interagir com o web service através de uma troca de mensagens SOAP (Simple Object Access Protocol), normalmente transportadas pelo protocolo HTTP com os dados serializados no formato XML (*Extensible Markup Language*). SOAP é um protocolo destinado à troca de informações em um ambiente distribuído e descentralizado. Ele define um esquema para uso de XML para representar o conteúdo da mensagem e como processá-la. Além de um conjunto de regras de codificação que expressam os tipos de dados definidos pela aplicação e uma convenção para representar as chamadas e respostas a procedimentos remotos. Inicialmente era baseado apenas em HTTP, mas a versão atual é projetada para usar uma variedade de protocolos de transporte incluindo SMTP, TCP ou UDP.

Muitos servidores web comerciais, como por exemplo Amazon, Yahoo, Google e eBay, oferecem web services que permitem aos clientes manipular os seus recursos web. Por

exemplo, o web service oferecido pela Amazon.com fornece operações que permitem aos clientes obterem informações sobre produtos, adicionar um item em um carrinho de compras ou verificar o status de uma transação. Isso permite que aplicações de outros fornecedores construam serviços com valor agregado sobre aqueles fornecidos pela Amazon.com. Por exemplo, uma aplicação de controle de inventário poderia pedir o fornecimento de mercadores da Amazon.com, à medida que eles fossem necessários, e controlar automaticamente a mudança de status de cada requisição. Outro exemplo de aplicação que exige a presença de um web service é a que implementa o *sniping* em leilões do eBay. *Sniping* significa fazer um lance durante os últimos segundos antes que um leilão termine. Embora as pessoas possam fazer isso através da interação direta com a página web, elas não conseguem fazer isso com tanta rapidez.

6. Faça uma comparação de web services com CORBA. (Luiza)

A principal diferença entre web services e CORBA é o contexto no qual eles se destinam a serem usados. O CORBA foi projetado para uso dentro de uma única organização, ou entre um pequeno número de organizações colaboradoras. Já o web service foram concebidos com o enfoque de integração e oferta de serviços pela web, não estando limitados a um subconjunto de organizações.

Os web services tem uma maior facilidade de uso, uma vez que a infra-estrutura XML e HTTP é bem entendida, conveniente para uso e já está instalada em todos os sistemas operacionais comumente usados. Em contrapartida CORBA é um software grande e complexo, o qual exige instalação e suporte.

Mesmo assim, CORBA pode continuar sendo uma melhor escolha se considerado seu uso dentro de um ambiente corporativo de uma empresa, pois os objetos distribuídos são projetados para troca de objetos em formato binário, de tipos específicos em um processo síncrono de troca de mensagens. Esse fato, aliado à dependência de plataforma, possibilita um alto desempenho. Por outro lado, os web services tendem a não apresentar um alto desempenho, devido ao tamanho e à sobrecarga necessária para o tratamento dos documentos XML. Um estudo realizado por Olson e Ogbuji mostrou que as mensagens de requisição SOAP são 14 vezes maiores do que as equivalentes em CORBA e que a requisição SOAP demora 882 vezes mais que uma invocação CORBA equivalente.

7. Defina ORB e cite algumas implementações. (Rafael Scortegagna)

Em computação distribuída, um ORB (object request broker) é uma peça de middleware que permite que os programadores façam chamadas de um computador a outro via uma rede.

ORB's promovem interoperabilidade entre objetos de sistemas distribuídos pois eles possibilitam que o usuário construa sistemas com "peças" de diferentes linguagens (se comunicam via ORB).

Em linguagem orientada a objetos, um ORB pega a assinatura de um objeto com métodos e habilita conxões com os objetos disponíveis. Depois de um objeto conectar com um ORB, os seus métodos ficam acessíveis para chamadas remotas.

Alguns ORB's, como o CORBA, usam uma IDL (Interface Description Language) para descrever os dados que serão transmitidos nas chamadas remotas.

Algumas implementações de ORB’s são: CORBA, .NET Remoting, Orbix, DCOM, RMI, RPC, ORBit, entre outras.

8- O que é GIOP (definição, tipo de mensagens, etc.)? (Rafael Scortegagna)

Em computação distribuída, GIOP (General Inter-ORB Protocol) é um protocolo abstrato pelo qual ORB’s se comunicam.

O GIOP é dividido em tres partes:

- **CDR** (common data representation): transferência de mapeamento de sintaxe de tipo de dados OMG IDL em uma representação “low-level” para transferência entre pontes ORB’s e inter-ORB’s.

- **IOR** (interoperable object reference): define o formato de uma referência para um objeto remoto.

- **Formato de mensagens definidos**: mensagens são trocadas entre agentes para facilitar a requisição de objetos, localizar a implementação de objetos e controlar os canais de comunicação. As mensagens são: Request, Reply, CancelRequest, LocateRequest, LocateReply, CloseConnection, MessageError, Fragment.

Temos alguns protocolos disponibilizados pelo GIOP, dentre estes citamos os seguintes:

- IIOP (Internet interORB protocol): implementação de GIOP para ser utilizado na internet (sobre TCP/IP).
  - SSLIOP (SSL interORB protocol): implementação IIOP sobre SSL, permitindo criptografia dos dados e autenticação.
  - HTIOP (HyperText interORB protocol): implementação IIOP sobre HTTP, permitindo transparência sob o ponto de vista de “firewalls”.

9. O que é a Web API ou a Web 2.0 (Fernanda)?

É o Web Services baseado em REST.

REST (Representation State Transfer) é uma arquitetura para sistemas distribuídos na qual solicitações e respostas são construídas em torno de **representações** de **recursos** que são acessados por uma **interface** em comum.

Um recurso pode ser essencialmente qualquer conceito coerente que pode ser endereçado como por exemplo um arquivo. Esse arquivo (recurso) pode ser representado por um estrutura que captura seu estado atual.

Podemos, por exemplo, dizer que o arquivo “exemplo” pode ser representado pela URL https:// sisop2.com.br/arquivo

O REST usa uma interface comum para acessar e modificar recursos. Essa interface é baseada no protocolo HTTP e, portanto, usa métodos como GET e POST para modificar e acessar recursos.

É importante salientar que REST é uma arquitetura e não somente um protocolo como SOAP. Podemos usar REST como uma abstração de SOAP por exemplo.

REST possui diversas vantagens quando comparado com o SOAP:

1. Posso utilizar diferentes representações: XML, JSON, HTML, PDF.... enquanto SOAP só usa XML em suas mensagens.
2. No SOAP toda a mensagem precisa ser encapsulada no XML o que requer uma maior largura de banda enquanto o REST não.
3. Flexibilidade.

Exemplo: Se a solicitação de uma mensagem pede, por exemplo, para converter uma moeda de dólar para real. No SOAP precisamos precisamos de uma biblioteca XML para gerar a solicitação e para ler a resposta. No REST a solicitação seria uma URL e a resposta seria um número gerado pelo script apontado na URL solicitada.

4. Segurança. SOAP não lida com autenticação, tudo fica a cargo do desenvolvedor, enquanto REST pode se adequar a políticas impostas pelo administrador de sistema.
5. Suporte a outros protocolos: O **Restfull** é uma extensão do REST na qual permite a transmissão de mensagens através de outros protocolos como SMTP e TCP.

Exemplo de REST: AJAX
Ajax é uma tecnologia baseada em REST. Requisiçoes em são enviadas usando objetos XMLHttpRequest. A resposta é usada pelo código JavaScript para mudar a página atualAJAX. Cada XMLHttpRequest é enviado com GET e a resposta é em JSON.

10. Quais sao os principais elementos uma comunicação webservice? (Fernanda)

Os principais elementos para uma comunicação via WebService são 3 protocolos (protocolos de descoberta, de envio de mensagens e da camada de aplicação) e uma linguagem (WSDL).
A quádrupla mais utilizada é HTTP, SOAP, WSDL e UDDI.

11. Cite e caracterize um protocolo de descoberta. (Fernanda)

UDDI (Universal Description, Discovery and Integration)
Quando se constrói serviços na Web, esses serviços necessitam ser acessados, em algum lugar na Web, por uma aplicação-cliente. Uma forma de se acessar um serviço é fazer com que a aplicação-cliente conheça a URI do serviço, desta maneira caracterizando o modo estático de se localizar e acessar um serviço. Entretanto, quando a aplicação-cliente não detém, a priori, a localização de um serviço na Web, esse, pode ser descoberto, antes de ser acessado, caracterizando o modo dinâmico de se descobrir a localização de um serviço.
UDDI é uma especificação técnica para descrever (describing), descobrir (discovering) e integrar serviços na Web. Assim, é portanto, uma parte crítica da pilha de protocolos para serviços Web, habilitando usuários dos serviços a publicarem e descobrirem serviços na Web.
UDDI transmite mensagens via SOAP e e um diretorio que contem informacoes sobre Web Services e suas interfaces descritas pela linguagem WSDL.

12. O que é WSDL (Web Services Description Language)? (Fernanda e Murilo)

É a linguagem baseada em XML, utilizada para descrever as funcionalidades oferecidas por um web service.
A WSDL provê: Como um serviço pode ser chamado, quais os parâmetros esperados e a estrutura de dados retornada (semelhante a assinatura de um método).

- Posso gerar WSDL a partir de classes (**bottom-up**):
  Um desenvolvedor cria as classes (em alguma linguagem de programação), e então usa uma ferramenta de geração de WSDL para expor métodos destas classes como um serviço Web.
- Posso gerar esqueletos de classes a partir de um WSDL (**top-down**):
  Escrevo WSDL e gero um esqueleto com a classes.

13. Cite e defina protocolo(s) de envio de mensagens. (Fernanda e Murilo)

Podemos utilizar RPC, RPC-XML ou SOAP.
XML-RPC se baseia em requests HTTP (o qual traz funcionalidades do protocolo HTTP - como autenticação - na comunicação entre máquinas). A passagem de parâmetros pode ser feita com XML o que abre mais possibilidades, como por exemplo, a passagem de estruturas de dados mais complexas em comparação com o RPC puro.

SOAP é um XML-RPC com suporte a transferência de estruturas mais complexas (document-level), e pode ser usado com várias linguagem de programação (desde que estas lidem com XML) e sobre vários protocolos (HTTP, SMTP e TCP) e, obviamente, com suporte a WSDL.

14. Cite protocolo(s) da camada de aplicação. (Fernanda e Murilo)
HTTP, SMTP ou FTP.

15. O que é um IIOP? (Luiza)

IIOP (Internet InterOrb Protocol) é um protocolo que torna possível programas distribuídos escritos em diferentes linguagens de programação se comunicarem através da Internet. Ele é uma parte crítica do CORBA, que passou a ser adotado na versão 2.0. Ele possibilita que uma empresa possa escrever programas que possam se comunicar com os programas de outras empresas sem ter que compreender nada sobre o outro programa sem ser o seu serviço e o seu nome.

Os programadores e usuários nunca são obrigados a interagir com IIOP de qualquer forma, ele é invisível para eles. O IIOP permite que seus programas possam interagir de forma transparente durante a execução, de modo que não se tem que escrever programas específicos para IIOP.

16. O que é o DCOM? (Luiza e Samuel)

DCOM (*Distributed Component Object Model*) é um produto comercial, desenvolvido pela Microsoft na década de 90, que vem sendo incluída no Windows desde a versão Windows 95. Ele possibilita a chamada de procedimentos remotos através do qual as aplicações COM (*Component Object Model*) podem comunicar-se empregando DCOM como protocolo. O DCOM é o principal concorrente do CORBA em computação de objeto distribuído.

Empregando a abordagem *stub* e skeleton, o DCOM expõe através de interface definida os métodos de objetos COM que podem ser invocados remotamente através da rede. O *stub* encapsula a localização na rede do servidor onde se encontra o objeto COM. Ele também se comporta como um proxy no lado cliente. Os servidores podem conter ou hospedar vários objetos COM, que devem se registrar no servidor, tornando-se visíveis e disponíveis para todos os clientes. Outra vantagem é que um único objeto DCOM pode interagir de maneira distinta com vários processos, devido a possibilidade do uso de várias interfaces.

17. Como funciona o DCOM?(Luiza e Samuel)

No DCOM cada interface é associada a um GUID (Globally Unique Identifier), que é o identificador de interface (IID). A aplicação cliente informa o GUID do objeto desejado e, opcionalmente, o endereço do servidor que será responsável pela sua execução.

Para invocar um método remoto o cliente faz uma chamada de *client stub.* O client stub chama a camada de protocolo do DCOM afim de enviar a requisição ao server stub que repassa a requisição ao objeto COM específico. Tanto o stub do cliente quanto do servidor são criados pelo IDL e são conhecidos, respectivamente, por *proxy* e por *stub*.

18. O que é IDL? (Alessandro)

Linguagem de Descrição de Interfaces (do inglês: Interface Description Language). É uma linguagem de especificação que permite a comunicação entre componentes de software que são desenvolvidos em linguagens diferentes (e consequentemente sobre hardwares diferentes). Utilizada na maioria das vezes em RPC's. Existem diversos mapeamentos feitos de IDL para as diferentes linguagens existentes. Sua estrutura de definição é parecida com uma struct da linguagem C. Uma nova IDL criada é um novo tipo para CORBA.

19. Qual a diferença entre STUB e DII? (Rodrigo)

Os objetos de CORBA são comumente acessados via stub, que são automaticamente gerados por um compilador IDL. Os stubs, entretanto, são gerados em tempo de compilação e são estáticos, ou seja, eles não podem ser alterados em tempo de execução. Mas o que fazermos se o tipo da interface de um objeto não é conhecido no tempo de compilação? O único caminho possível de acessar objetos nesse caso é usar o DII. Essa interface pro ORB oferece a possibilidade de invocar operações cuja assinatura não é conhecida no tempo de compilação.

20. Como são transportadas as mensagens no padrão CORBA? (Rodrigo)

São transportadas pelo componente ORB, normalmente utilizando um protocolo IIOP.

ORB (Object Request Brokers) são componentes de software que mediam a transferência de mensagens de um programa para um objeto localizado num servidor remoto da rede. O ORB esconde do programador a complexidade da comunicação de rede.

Um ORB te deixa criar um objeto de padrão de software cujas funções podem ser invocadas pelos programas clientes localizados em qualquer lugar da rede. Um programa que contém instâncias de objetos CORBA é frequentemente conhecido como servidor. Entretanto, o mesmo programa pode ele mesmo invocar chamadas em outros programas servidores e assim se comportar como
cliente.

Quando um cliente invoca uma função de um objeto CORBA, o ORB intercepta a chamada da função. O ORB, então, redireciona a chamada pela rede para o objeto alvo. O ORB então coleta os resultados da chamada da função e retorna ao cliente.

22. O que significa OODB? (Oggo)

Nada mais é do que um banco de dados que mostra suas informações em forma de objetos (como utilizado em linguagens de programação orientada a objetos).

23. Cite uma vantagem e uma desvantagem do OODB: (Oggo)

Vantagem: Benchmarkings entre OODB's e bancos de dados relacionais, mostraram que os OODB's tem uma certa superioridade em certos tipos de consultas.

Desvantagem: Técnicas baseadas em ponteiros (que é a técnica do OODB, um ponteiro que aponta para um objeto) pode tornar uma consulta extremamente complexa se comparada aos bancos relacionais.

24. Escolher REST ou SOAP? (Oggo)

Rest:

Como REST utiliza HTTP, é muito mais simples sua utilização e sua documentação é mais fácil de compreender. Ele permite vários tipos de dados, o SOAP permite apenas xml. Podemos também definir um mesmo nome de um método, mudando apenas o tipo de passagem de dados e retorno, um exemplo de código se encontra abaixo:

```
@Path("/Order/")
    public interface OrderInfo {
```

```
@GET
@Produces ("application/xml")
@Path("{orderId}")
public Order getOrder(@PathParam ("orderId") int officeId);

@GET
@Produces ("application/xml")
@Path ("All")
public OrderList getAllOrders();
}
```

O REST também é mais rápido e pode usar cache, SOAP não utiliza cache e normalmente ocorre overhead.

SOAP:

SOAP suporta SSL (assim como o REST) mas também suporta WS-Security, que adiciona mais segurança à passagem de dados (como assinatura nos XML's e também encriptografa o XML). Também se o usuário necessitar de transações atômicas (garantia de todas serem executadas) ele deverá utilizar o SOAP, já que o REST não possui essa feature.

E então, qual usar?

Em suma, depende da aplicação que você vai desenvolver. Já trabalhei com um sistema de NF-e (Nota fiscal eletrônica) e mandávamos para a fazenda lotes de notas via SOAP, pois precisávamos de transações atômicas para este tipo de passagem de dados e uma maior segurança. Quando utilizávamos WebServices internos, da própria ferramenta no mesmo servidor, utilizávamos REST (pela sua simplicidade e velocidade e menor volume de dados).

25. Quais os principais mecanismos de segurança utilizados nos WebServices? (Murilo)

A segurança é um dos pontos fracos apresentados por esta tecnologia. Não pela falta de mecanismos de segurança, mas sim pela falta de consenso em qual deve ser utilizado. Como não existe um suporte único definido para segurança nos Web Services, isto faz com que cada projeto utilize diferentes soluções para resolver este problema o que se torna incompatível com a promessa de implementar uma normalização a nível global.

As questões mais importantes que devem ser levadas em consideração quanto a segurança nos WebServices é a autenticidade, privacidade e integridade das informações. Entre os principais mecanismos de segurança utilizados pelos WebServices estão o SSL, Xml Signature, Xml Encrypton, Ws-Security e SAML.

26. Cite alguns produtos que foram implementados utilizando arquitetura CORBA: (Murilo)

1. ChorusORB (Sun)
2. Component Broker/DSOM (IBM)
3. ILU (Xerox Parc)
4. Netscape Internet Service Broker (Netscape)
5. Object Director (Fujitsu)
6. omniORB2 (AT&T Laboratories)
7. ORB Plus (HP)
8. RPC-ORB (Nortel)
9. VisiBroker (Borland)

27. O que significa CCM ? Qual sua funcionalidade ? (Gustavo)

CCM é a sigla utilizada para representar Corba Component Model. Ela é uma extensão que foi introduzida com a CORBA 3. A funcionalidade do CCM é servir como um framework para desenvolvimento de componentes CORBA. A estrutura do CCM é uma espécie de “container” que armazena os componentes do CORBA, e além disso ele fornece serviços aos componentes que o utilizam. Alguns exemplos de servicos são notificação, autenticação, persistencia e processamento de transações.

Os serviços disponibilizados são acessados por interfaces bem conhecidas aos componentes, chamadas de portas. Com esses serviços feitos ao nível de CORBA, a complexidade dos componentes é diminuída muito, evitando que o desenvolvedor de cada componente precise se preocupar com, por exemplo, problemas de comunicação entre diferentes processos.

28. Qual a funcão da IR(Interface Repository) ? (Gustavo)

A Interface Repository serve como um repositório de interfaces de comunicão utilizados por programas clientes e/ou servidores que não conhecam as disponíveis interfaces em tempo de execução. Desta forma, sua principal funcionalidade é resolver problemas de tipagem em tempo de execução. Basicamente para acesso a este repositório temos duas funções, uma de escrita (criar as definições da interface) e uma de leitura (utilizada pelos componentes). As funçoes de escrita sao comunmente executada em conjunto com a IDL, examinando as árvores de código geradas pelo compilador, e adicionando as interfaces com seus respectivos tipos, tipos de variáveis e outras informações. O acesso a este repositório pode ser feito de duas maneiras:

1- passando uma string ao metodo **ORB::resolve_initial_references**, que retorna um ponteiro para um objeto do tipo **CORBA::Repository**, conseguindo assim acesso ao repositório e todas suas definições;

2- utilizando o método **get_interface** de alguma interface do tipo **CORBA::Object**, este método então retorna um objeto do tipo **CORBA::InterfaceDef** que tem acesso a todas definições de tipos do objeto que o requisitou.

29. Como CORBA suporta uma comunicação independente de linguagem? (Marilia)

CORBA primeiramente define uma linguagem padrão de interface chamada IDL, puramente descritiva sem o conceito de classes ou qualquer outra estrutura que dependa de uma linguagem de programação específica. Essa linguagem dispõe de facilidades que provêem a descrição de estruturas, atributos e assinaturas dos métodos que serão implementados pelos objetos remotos. CORBA então especifica o mapeamento da linguagem IDL para linguagens de implementação específicas como C++, Java e etc. Esse mapeamento sofre algumas dificuldades conforme a linguagem, por exemplo, a IDL aceita mais de um retorno para um método enquanto Java só permite um. Já existem mapeamentos padrões para Ada, C, C++, Lisp, Ruby, Smaltalk, Java, Cobol, PL/I e Python.

30. Quais são os modelos de invocação de objetos suportados por CORBA? (Marilia)

O Modelo Síncrono tem uma semântica At-Most-Once e bloqueia o cliente até que uma resposta seja retornada ou uma exceção seja lançada. O Modelo One-Way usa a semântica Best-Effort no qual o cliente continua sua execução imediatamente sem esperar nenhuma resposta do servidor. O Modelo Síncrono-Adiado tem uma semântica At-Most-Once onde o cliente continua sua execução imediatamente mas pode mais tarde bloquear a espera de uma resposta.

31. (Web Services) O que é W3C e qual sua relação com os Web Services? (Arthur)

O World Wide Web Consortium é uma comunidade internacional, formada por empresas, organizações independentes e governamentais, um staff próprio e pelo público em geral. Seu objetivo é desenvolver padrões (ou normas) para a Web (Web Standards). Os princípios dessa comunidade são prover: “Web para todos” e “Web em tudo”. O W3C tem padrões (standards) para vários segmentos da Web, como: design, aplicações, arquitetura, tecnologia XML, dispositivos móveis, browsers e *serviços.* As normas estão sendo revisadas constantemente. Elas pode ser vistas no site do W3C (www.W3C.org).

O W3C define e padroniza os protocolos (HTTP, SOAP), a descrição de serviço (WSDL, SML), a segurança (transferência de dados e autenticação) e a internacionalização dos Web Services.

32. (Web Services) Quais as características dos protocolos HTTP, REST e SOAP, e quais as suas diferenças? (Arthur)

O (Hypertext Transfer Protocol) HTTP/1.1 foi definido pela RFC 2616, de junho de 1999. O HTTP utiliza o modelo cliente-servidor e faz a comunicação com a troca de mensagens. O Cliente envia uma (mensagem de) *requisição* e o Servidor responde com uma (mensagem de) *resposta*.

O (Representational State Transfer) REST é um estilo de arquitetura de software para sistemas distribuídos. Foi introduzido e definido em 2000 por Roy Fielding, que também é um dos principais autores das especificações 1.0 e 1.1 do HTTP. O REST também utiliza o modelo cliente-servidor, pois ele é baseado no HTTP 1.0 e foi desenvolvido paralelamente com esse protocolo.

O (Simple Object Access Protocol) SOAP foi projetado em 1998 e tornou-se uma recomendação da W3C em 2003. Ele é um protocolo para a troca de informação estruturada em Web Services. O formato das suas mensagens são em XML, mas ele se baseia em outro protocolos, como HTTP e SMTP para a transmissão e negociação de mensagens.

O REST é uma arquitetura, que apesar de fundada sobre o HTTP não se limita a este protocolo para a troca de informações. O SOAP é considerado uma alternativa ao REST. O HTTP é (ou pode ser) usado tanto pelo REST quanto pelo SOAP, por ser apenas a forma de comunicação.

33. Compare pelo menos três destes aspectos de CORBA e Webservices:
Modelos de dados, Acoplamento cliente/servidor, Endereçamento de recursos, Sistema de tipos, tratamento de erros, serialização, passagem de parâmetros, síntaxe de transferência, noção de estado, semântica de requisição, Composição em tempo de execução, registro, descoberta de serviço, eventos. **(Jonata)**

Aspecto	**CORBA**	**Web services**
Modelo de dados	**Baseado em objetos**	**Baseado em troca de mensagens**
Acoplamento cliente/ servidor	**Grande**	**Pequeno**
Endereçamento de recursos	**Referência de objetos**	**URL**

Sistema de tipos	**IDL (verificação estática e em tempo de execução)**	**XML schemas (em tempo de execução)**
Tratamento de erros	**Exceções IDL**	**Mensagens de falha SOAP**
Serialização	**Integrada no ORB**	**Definida pelo usuário**
Passagem de parâmetros	**Por referência ou valor**	**Por valor**
Síntaxe de transferência	**CDR (binário)**	**XML (Unicode)**
Noção de estado	**Sim**	**Não (Stateless)**
Semântica de requisição	**no máximo uma vez**	**definido pelo SOAP**
Composição em tempo de execução	**DII**	**UDDI/WSDL**
Registro	**Repositório de interface**	**UDDI/WSDL**
Descoberta de serviço	**Serviço de nomes do CORBA ou registro RMI**	**UDDI**
Securança	**Serviço de segurança CORBA**	**HTTP/SSL, Assinatura XML**
Eventos	**Serviço de Eventos**	**Não tem**

34. O que é POM e PDS? E quala a relação entre eles?
O PDS implementa o mecanismo de persistência de dados. É responsável por levar os dados de um objeto para o meio de armazenamento. Cada PDS suporta um protocolo e um tipo de *datastore*, cabendo ao POM descobrir qual PDS encaixa-se melhor às necessidades de cada objeto persistente. O papel executado por ele é crucial pois, além de localizar os dados dos objetos nos *datastores*, também deve traduzi-los em ambos sentidos (passar do formato usado pelo objeto para o do armazenamento e vice-versa) e obter eficiência nesse acesso.

35. O que é POS? E para o que serve?
O serviço de persistência de objetos permite que um objeto sobreviva ao término da aplicação que o criou ou do cliente que o utilizou.
O POS provê interfaces para os mecanismos que implementam o armazenamento e gerenciam o estado persistente dos objetos, fornecendo suporte a bancos de dados e sistemas de arquivos e, ao mesmo tempo, independência de dados.

36. A arquitetura CORBA pode ser vista como uma pilha de protocolos, serviços e linguagens da seguinte forma: TCP/IP -> GIOP/IIOP -> CDR -> Stubs CORBA -> IDL.

Como seria a pilha equivalente em Webservices? (Jonata)
TCP/IP -> HTTP -> XML -> SOAP -> UDDI -> WSDL

37. Quando RESTfull webservices são uma boa escolha? (Barbara)
Quando a banda e os recursos são limitados; quando são usadas operações que não guardam estados (*stateless*) CRUD; quando é necessário o uso de *cache;* quando é necessário expor dados;

38. Quando SOAP baseado em Webservics é uma boa solucão?(Barbara)
Quando a invocação e o processamento são assincronos (se é necessário prover confiabilidade e segurança, o SOAP 1.2 oferece padrões que garantem esse tipo de operação, tais como WSRM - WS - Reliable Messaging, etc); quando produtor e consumidor precisam acordar um formato de troca, o SOAP 1.2 fornece as especificações para esse tipo de comunicação; quando são necessárias operações *statefull,* o SOAP 1.2 oferece operações na estrutura WS* para prover tais operações; quando é necessário expor a lógica.

39. Cite desvantagen(s) no uso de Web Servces.(Gabriel)
As maiores desvanteges no uso de web services em geral estão na sua eficiência. As mesagens são trocadas no formato de texto/XML, o que acaba envolvendo a troca de muita “informação inútil”.

40. Qual tipo de webservice usar quando clientes serão dispositivos móveis?(Paulo)
Dispositivos móveis muitas vezes tem acesso a web por uma conexão limitada. Portanto a melhor opção seria um webservice REST usando dados JSON principalmente pelo menor overhead de informação que no SOAP ocorreria por ser obrigatório o uso de XML. Web services REST oferecem: facilidade de desenvolvimento, performance, uso eficiente da banda, oportunidade para segurança e robustes.

41. Que vantagens o uso do HTTP trás a webservices REST?(Paulo)
As mensagens e códigos de erros do REST são herdados dp HTTP.
Os mecanismos de segurança disponiveis no HTTP/S.

42. A plataforma básica do Webservices XML + HTTP. O que são elas? (Eduardo)
XML (Extensible Markup Language) - É uma linguagem de marcação, como o HTML, que define regras de codificação de um documento para poder ser lido por máquina e pessoas. Diferente do HTML tem uma sintaxe bem definida o que diminue a complexidade dos parsers. É utilizada normalmente para armazenar, estruturar e transportar informações. É baseado em um arquivo de texto simples.
Várias linguagens basea-se em XML como: XHTML (antigo sucessor do HTML 4), MathML, SVG (formato gráfico vetorial), etc.
HTTP (Hypertext Transfer Protocol) - é um protocolo de rede da camada de aplicação e é utilizado para sistemas de informação de hipermedia distribuídos e colaborativos.
Ele basicamente é responsável por tratar requisição e respostas entre cliente e servidor na World Wide Web, ou seja, o cliente envia um pedido ao servidor e servidor processa e envia resposta ao cliente e encerrada a comunicação.
Fonte:
Wikipédia PT - HTTP - XML
Wikipedia EN - HTTP - XML
W3schools

43. Descreva o formato de dados JSON. (Paulo)
JSON é um formato usado para serializar e transmitir dados pela internet oferecendo uma alternativa ao XML. Existem bibliotecas para parser de dados no formato JSON em quase todas linguagens de programação e podem ser conferidas na pagina http://json.org/.
Existem JSON schemas usados para verificar a estrutura e os tipos de dados dentro de um objeto JSON, é semelhante a XML schema.
Os tipos basicos são:

- Números
- Strings: representadas entre aspas
- Boolean: true ou false
- Array: Lista de valores separados por virgula entre conchetes.
- Objeto: conunto não orenado de pares key:value.

Para maiores informações: http://en.wikipedia.org/wiki/JSON

44. Cite, se existe, alguma vantagem do uso de SOAP sobre REST.(Gabriel)
Possui um melhor suporte de ferramentas.