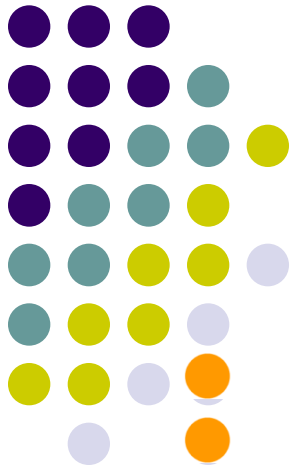
Arquitetura de Software

Web Services & SOA

Web Services e Service-Oriented Architecture

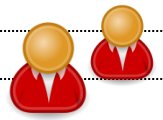
José Motta Lopes
josemotta@bampli.com



Agenda

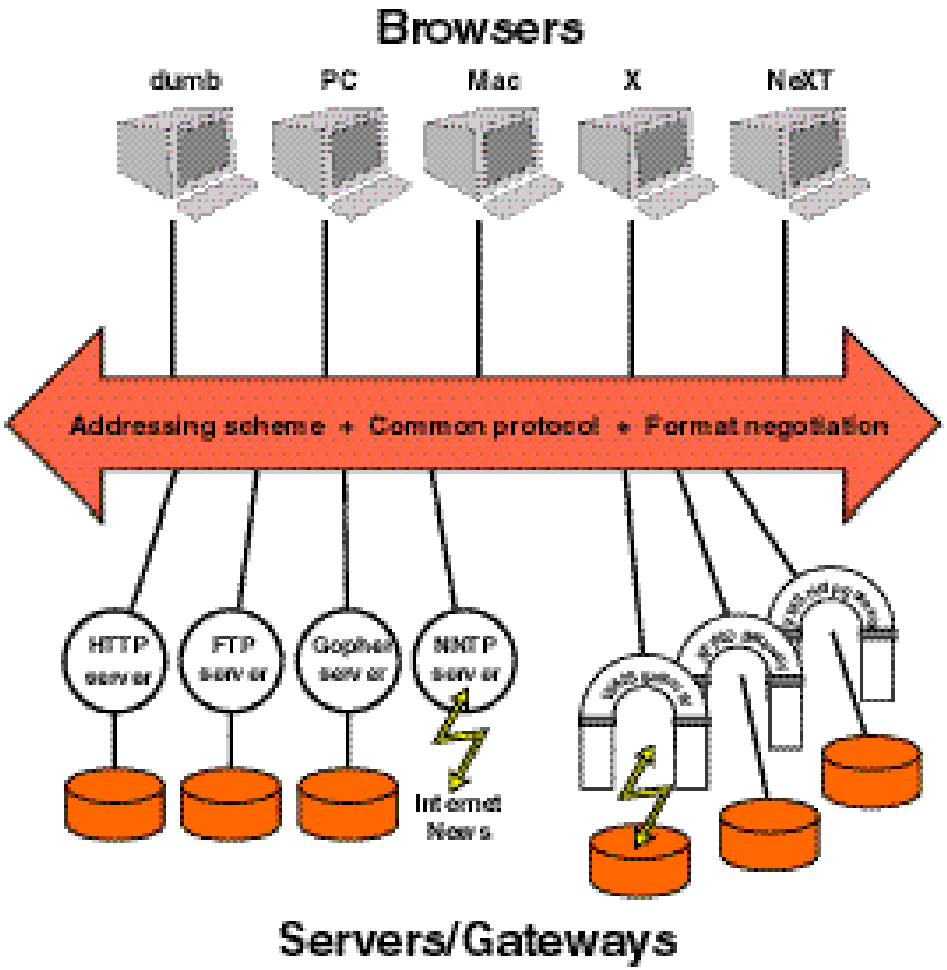
- WWW
- RESTful Web Services
  - REST
  - Interface uniforme
  - CRUD
  - Mensagens auto-descritivas
  - Interações stateful / hyperlinks
  - HATEOAS
  - Identificação por URI
- SOA
  - Open Group
  - Modelo de Referência SOA
  - Princípios SOA
  - Camadas SOA

Últimos 50 anos de TI



WWW

ARQUITETURA INICIAL DA WEB



RESTful Web Services

SERVIÇOS WEB RESTFUL, CRIADOS PARA FUNCIONAR NA WEB

- **Arquitetura que especifica restrições, com interface uniforme**
- **Serviços com desempenho, escalabilidade e modificabilidade**
- **Dados e funcionalidades são considerados recursos**
- **Operações com restrições REST, simples e bem definidas**
- **Acesso através de URIs (Uniform Resource Identifiers)**
- **Protocolo de comunicação stateless, geralmente http**
- **Aplicações simples, leves e rápidas**

REST

ESTILO DE ARQUITETURA COM RESTRIÇÕES

- **Cliente-servidor**
- **Servidor stateless melhora escalabilidade, visibilidade e confiabilidade, simplificando a implementação do servidor.**
- **Cache**
- **Interface uniforme**
  - **Identificação de recursos**
  - **Manipulação de recursos através de representações**
  - **Mensagens auto-descritivas**
  - **Hypermedia como máquina do estado da aplicação**
- **Sistema em camadas**

Interface uniforme

APLICAÇÕES SIMPLES, LEVES E RÁPIDAS.

- **Principal restrição que distingue REST de outros estilos**
- **Cliente e servidor podem evoluir independentemente**
- **Componentes desacoplados**
- **Identificação de recursos por meio de URI**
- **Manipulação de recursos através de representações**
- **Mensagens auto-descritivas com formatos tipo JSON e XML**
- **Interação com estado através de hyperlinks**
  - **Hypermedia as the engine of application state (HATEOAS)**

CRUD

OPERAÇÕES CREATE, READ, UPDATE & DELETE

- **Recurso manipulado por conjunto fixo de 4 operações:**
  - **GET recupera o estado atual de um recurso**
  - **PUT cria um recurso novo**
  - **POST transfere um novo estado para um recurso**
  - **DELETE exclui recurso**

.NET C#

```
/// <summary>
///
/// </summary>
/// <remarks>flashes ir code simulating the remote control</remarks>
/// <param name="remote">Lirc remote</param>
/// <param name="code">ir code</param>
/// <response code="200">response</response>
[HttpPost]
[Route("/motta/home/1.0.1/remotes/{remote}/{code}")]
[ValidateModelState]
[SwaggerOperation("SendRemoteCode")]
[SwaggerResponse(200, typeof(ApiResponse), "response")]
public virtual IActionResult SendRemoteCode([FromRoute]string remote, [FromRoute]string code)
{
    string example = (@"/usr/bin/irsend send_once " + remote + " " + code).Bash();

    return new ObjectResult(example);
}
```

Mensagens auto-descritivas

RECURSOS DISSOCIADOS DE SUAS REPRESENTAÇÕES

- **Conteúdo do recurso acessado em vários formatos**
- **JSON, HTML, XML, texto simples, PDF, JPEG, etc.**
- **Disponibilização de metadados sobre o recurso**
  - **Armazenamento em cache**
  - **Deteção de erros de transmissão**
  - **Negociação do formato mais apropriado**
  - **Autenticação e controle de acesso**

Interações & Hyperlinks

TODA INTERAÇÃO COM RECURSO É SEM ESTADO

- **Mensagens autônomas de solicitação**
- **Interações baseadas na transferência de estado explícita**
- **Estado pode ser incorporado à mensagem de resposta**
- **HATEOAS aponta para estados futuros válidos da interação**
- **Propriedadades de visibilidade, confiabilidade e escalabilidade**
- **Várias técnicas para troca de estado**
  - **Reescrita de URI**
  - **Cookies**
  - **Campos de formulário ocultos**

HATEOAS

HYPERTEXT AS THE ENGINE OF APPLICATION STATE

Links indicam estado do sistema e caminhos possíveis a seguir:

```
GET /account/12345 HTTP/1.1 HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">100.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
    <link rel="withdraw" href="/account/12345/withdraw" />
    <link rel="transfer" href="/account/12345/transfer" />
    <link rel="close" href="/account/12345/close" />
</account>
```

Conta bancária no negativo, não apareceria opção para saque!

```
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">-25.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
</account>
```

Identificação por URI

ENDEREÇAMENTO E ROTEAMENTO NA WEB

- **Serviço da web RESTful expõe recurso**
- **Espaço de endereçamento global para descoberta de recursos**
- **Templates para caminhos URI**

.NET C#

```
/// <summary>
///
/// </summary>
/// <remarks>returns all ir codes from remote</rem
/// <param name="remote">Lirc remote</param>
/// <response code="200">All the codes</response>
[HttpGet]
[Route("/motta/home/1.0.1/remotes/{remote}")]
[ValidateModelState]
[SwaggerOperation("GetRemoteCodes")]
[SwaggerResponse(200, typeof(List<string>), "All the codes")]
public virtual IActionResult GetRemoteCodes([FromRoute]string remote)
{
    string example = (@"/usr/bin/irsend list " + remote + @" """""").Bash();

    return new ObjectResult(example);
}
```

JAVA

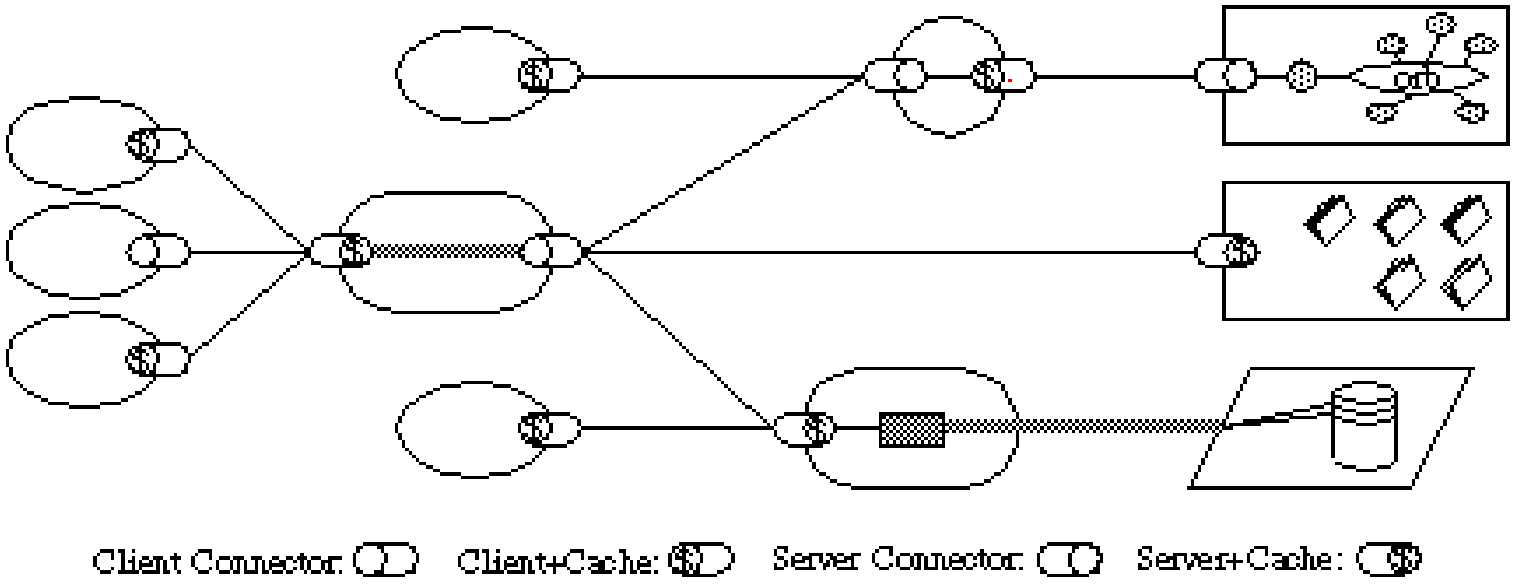
```
@Path("/users/{username}")
public class UserResource {

    @GET
    @Produces("text/xml")
    public String getUser(@PathParam("username") String userName) {
        ...
    }
}
```

Sistema em camadas

NÍVEIS HIERÁRQUICOS RESTRIGEM AÇÃO DE COMPONENTES

- **Componentes não enxergam além da camada que interagem**
- **Camadas encapsulam serviço legado, protegendo novos serviços**
- **Intermediários melhoram escalabilidade e balanceamento de carga**
- **Caches intermediários e compartilhados melhoram performance**



SOA

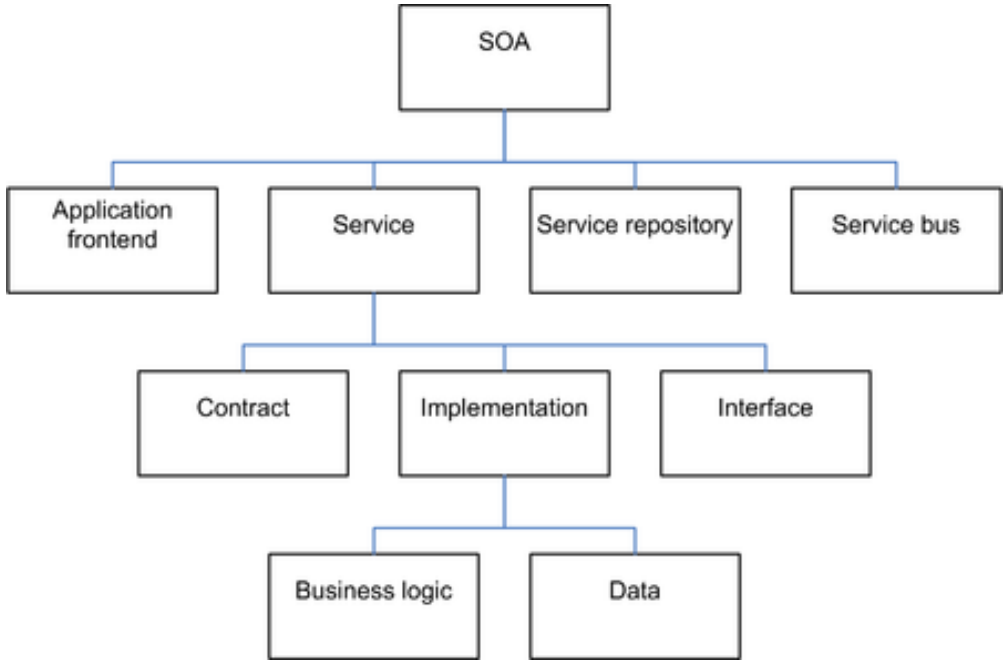
SERVICE ORIENTED ARCHITECTURE

- **Arquitetura Orientada a Serviços é um estilo de projeto de software**
- **Funcionalidades são disponiblizadas em forma de serviço**
- **Serviços são oferecidos pelos componentes de aplicativos**
- **Serviços utilizam protocolos de comunicação em rede**
- **Independente de fornecedores, produtos e tecnologias**
- **Serviço é uma unidade discreta de funcionalidade**
- **Serviço tem acesso e execução remota, de forma independente**

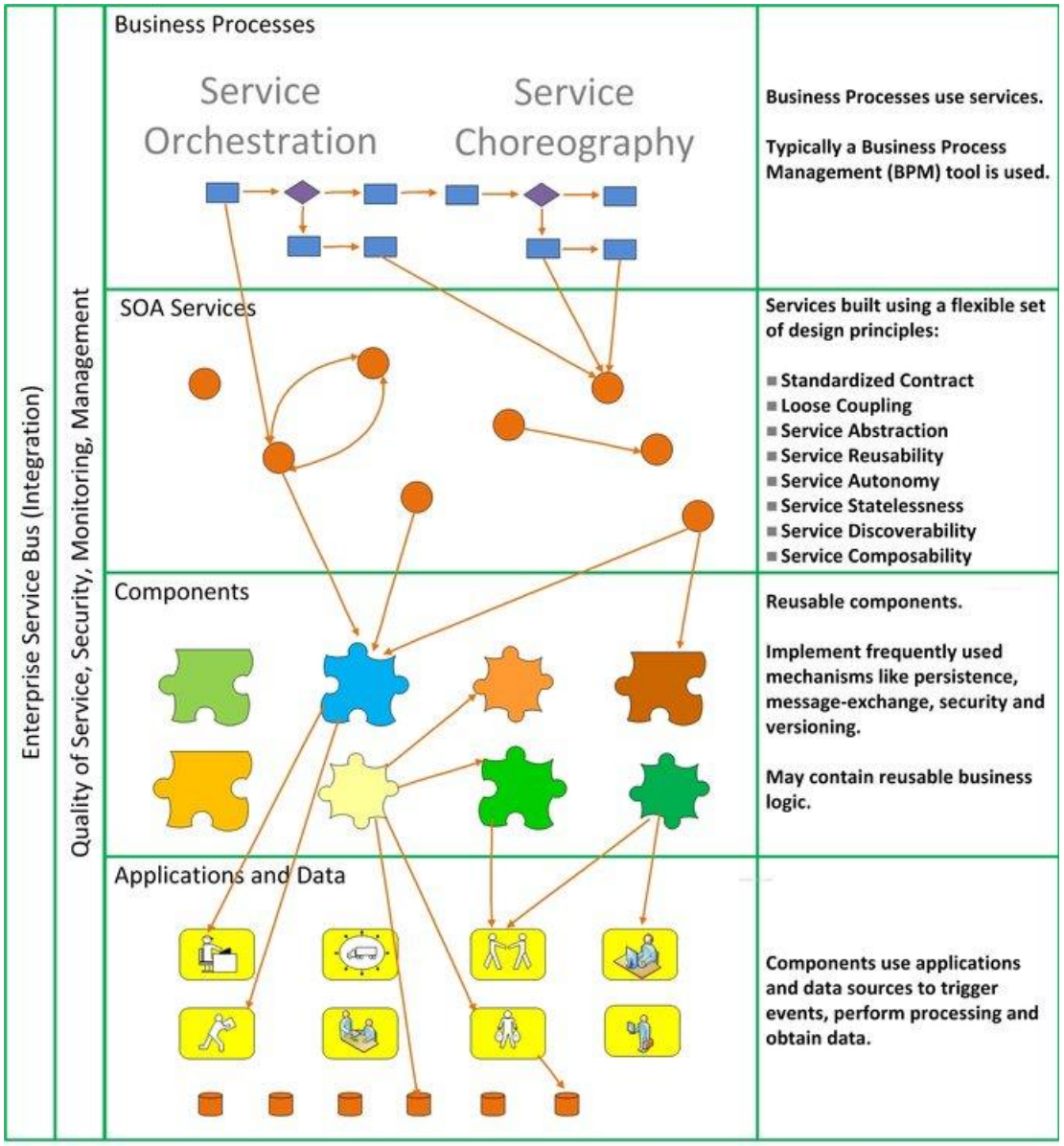
SOA

ORIENTAÇÃO A OBJETOS PARA O MUNDO CORPORATIVO

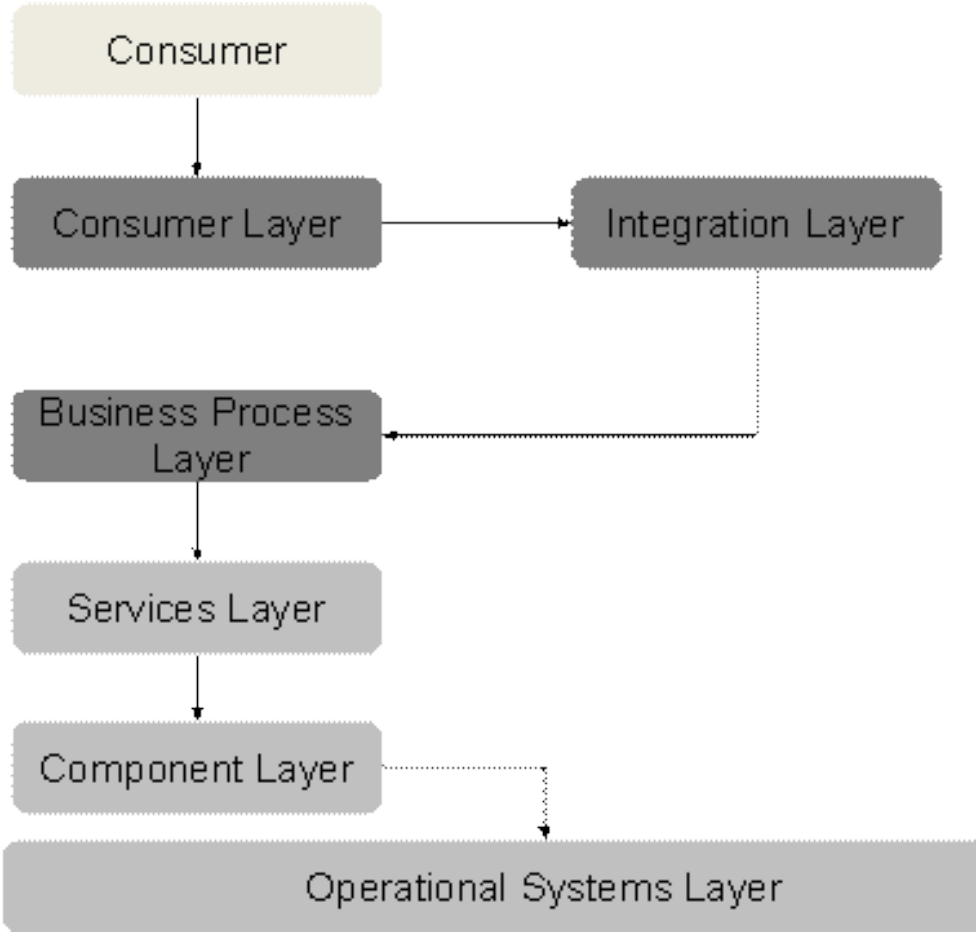
- **Evoluiu dos princípios usados nas linguagens orientadas a objeto**
- **SOA trouxe estes princípios para o mundo corporativo**
- **Estabelece biblioteca reusável de serviços fracamente conectados**



SOA


Enterprise Service Bus (Integration)
Quality of Service, Security, Monitoring, Management
Business Processes
Service Orchestration
Service Choreography
Business Processes use services.
Typically a Business Process Management (BPM) tool is used.
SOA Services
Services built using a flexible set of design principles:
Standardized Contract
Loose Coupling
Service Abstraction
Service Reusability
Service Autonomy
Service Statelessness
Service Discoverability
Service Composability
Components
Reusable components.
Implement frequently used mechanisms like persistence, message-exchange, security and versioning.
May contain reusable business logic.
Applications and Data
Components use applications and data sources to trigger events, perform processing and obtain data.

Camadas SOA



Consumidores de serviço acionam a camada de integração e com isso, ...

... processos de negócios são ativados na camada de serviços.

Os serviços ativam a camada de componentes, ...

... que por sua vez, utilizam a camada de sistemas operacionais.

Princípios SOA

9 PRINCÍPIOS PARA O PROJETO DE UM SERVIÇO SOA

- **Contrato de Serviço Padrão adere a uma descrição de serviço**
- **Fraco acoplamento minimiza as dependências entre si**
- **Abstração de serviços oculta a lógica que eles encapsulam**
- **Lógica maximiza a reutilização de serviços**
- **Autonomia inclui controle sobre a lógica que serviços encapsulam**
- **Apatidão do serviço, idealmente sem estado (stateless)**
- **Serviços descobertos através de serviço de deteção de serviços**
- **Serviço decompõe problema grande em pequenos problemas**
- **Interoperabilidade de serviços através de padrões adequados**

SOA

HÁ MUITAS DEFINIÇÕES DE SOA

- **Service-Oriented Architecture Standards do OpenGroup**
  - **Formado por grupo de empresas**
  - **Arquitetura Corporativa, Computação na nuvem, Plataformas, Segurança, Ciclo de vida, Segurança, Arquitetura Orientada a Serviço (SOA), Sistemas Tempo Real e Embarcados, IT4IT para gerência de negócios, etc.**
- **OASIS Reference Model for SOA**
  - **Modelo de referência com framework abstrato**
  - **Grupo aberto aborda ecossistema, integridade e implementação**
  - **Sem uma entidade autoritativa única para governança SOA**
- **SoaML Specification do Object Management Group**

Open Group

MEMBROS ASSOCIADOS

- Membros Platina (8)

  DXC, Fujitsu, HCL, Huawei, IBM, Micro Focus, Oracle e Philips

- Membros Gold (57)

  American Express, Boeing, Microsoft, Ministério Defesa UK, NASA, US DoD, etc

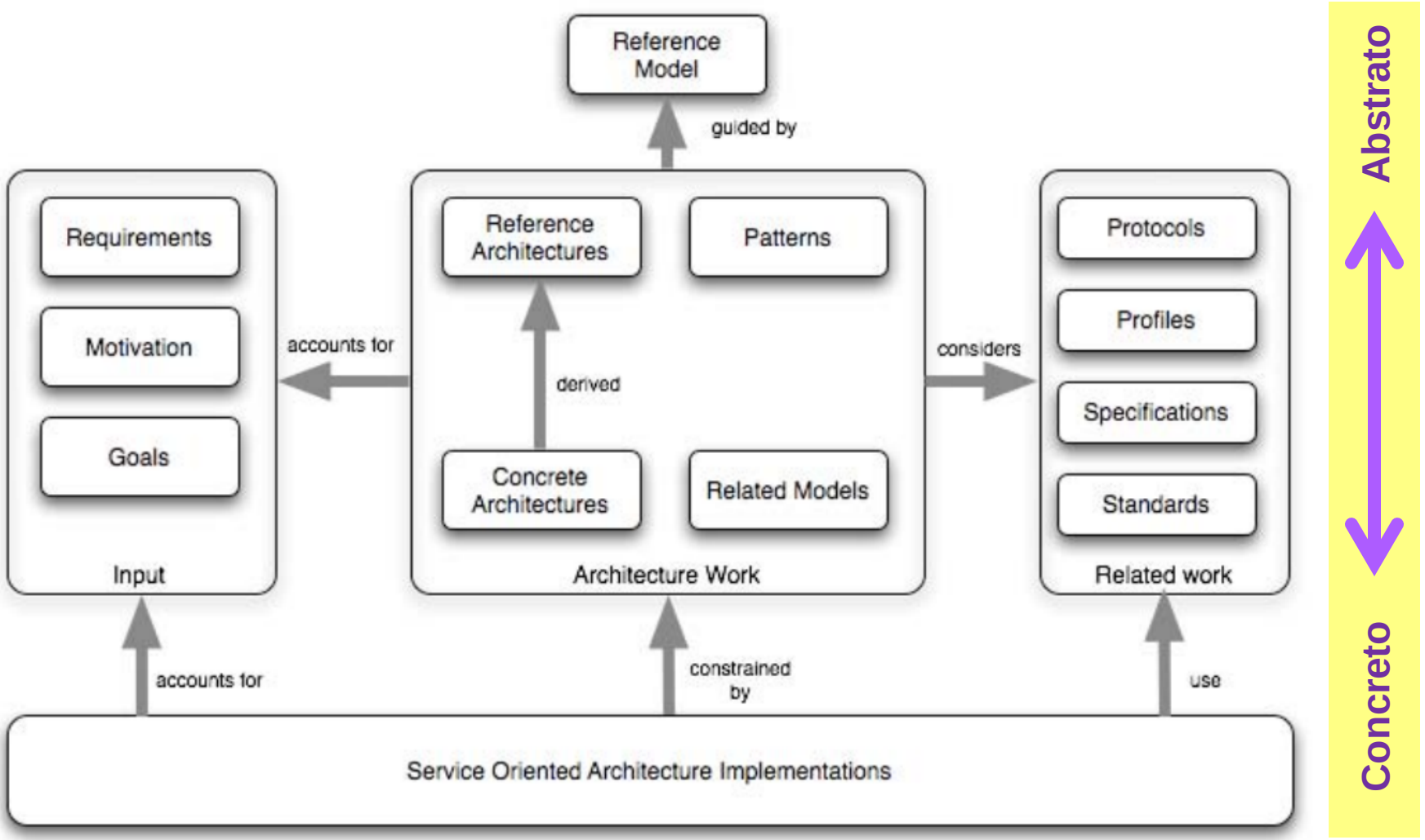
- Membros Silver (396)

  Air China, Apple, BASF, Bayer, Cisco, Dell, DuPont, ExxonMobil, GE, Intel, Japan Aerospace, Rockwell, SAP, Siemens, Société Générale, etc

- Membros Academic (40), entre outros.

DXC.technology FUJITSU HCL HUAWEI IBM MICRO FOCUS ORACLE PHILIPS

Modelo de Referência SOA



Modelo de Referência SOA

A QUEM INTERESSA O MODELO DE REFERÊNCIA?

- **Arquitetos e desenvolvedores que projetam, identificam ou desenvolvem sistemas baseados no paradigma orientado a serviço**
- **Arquitetos de padrões e analistas desenvolvendo especificações que aderem aos conceitos de orientação a serviços.**
- **Tomadores de decisão que buscam um entendimento comum consistente de arquiteturas orientadas a serviço.**
- **Usuários que buscam conceitos e benefícios da arquitetura orientada a serviço.**