

IA847 - Projeto

Sistema de Coordenação do Acesso para Web Labs

Março de 2007

1 Objetivos

O sistema a ser desenvolvido tem por objetivo coordenar o acesso por parte de um grupo de usuários a um Web Lab. A coordenação do acesso visa evitar inconsistências causadas por acesso concorrente a funcionalidades do Web Lab, por exemplo, a operação de um robô por dois ou mais usuários simultaneamente. O sistema de coordenação do acesso estabelece também um mecanismo mínimo de comunicação (baseado em texto) visando facilitar a própria coordenação entre os membros do grupo.

2 Requisitos do Sistema

O sistema de coordenação do acesso deve ser aderente ao modelo conceitual de Web Lab apresentado na Figura 1. Os elementos que compõem um Web Lab, bem como as relações entre estes elementos são apresentados a seguir.

Um *participante* pode constituir-se de um *usuário* individual, um *grupo* de usuários ou (recursivamente) de participantes. Um Web Lab agrega *experimentos* e a relação de um Web Lab para com si próprio indica que Web Labs podem ser federados para aumentar a gama de experimentos oferecidos e/ou para beneficiar a uma gama maior de usuários. Experimentos são atividades executadas a distância que utilizam *serviços* como elementos de constituição. Serviços disponibilizam tanto funcionalidades específicas, como manipulação de robôs e câmeras, quanto funcionalidades de propósito geral como controle de acesso e comunicação inter-pessoal. Serviços fazem uso de *recursos*, tanto físicos (robôs, câmeras, etc.) quanto lógicos (pacotes de simulação, de visualização, etc.).

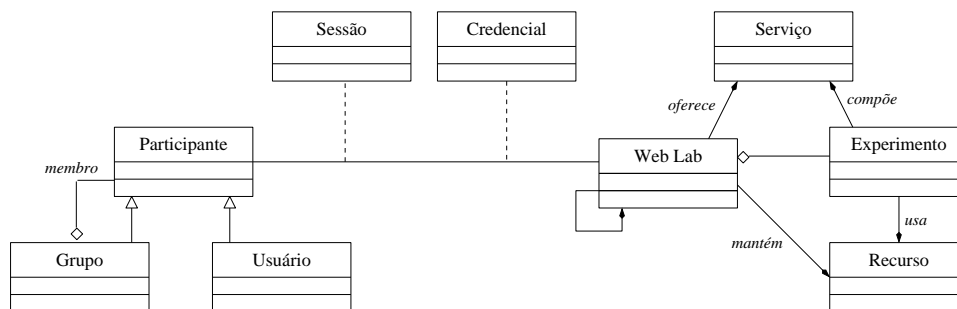


Figure 1: Modelo conceitual para Web Labs.

Cada participante possui uma *credencial* para utilizar um Web Lab. Credenciais contêm papéis, permissões, assinaturas digitais, etc. São exemplos de papéis aluno e professor. Permissões definem atribuições, usualmente associadas a papéis, tais como permissão para cadastrar usuários, para disponibilizar experimentos e para executar experimentos. A interação do participante com o Web Lab é regida por uma ou mais *sessões*. Uma sessão armazena o estado da interação do participante com o Web Lab. Serviços interativos como Web Labs necessitam de pelo menos três classes de sessão:

1. Sessão de acesso: controla o acesso do participante ao Web Lab de acordo com suas credenciais.
2. Sessão de interação: suporta a manipulação dos recursos necessários à realização dos experimentos mantidos pelo Web Lab.
3. Sessão de comunicação: suporta comunicação entre o Web Lab e participantes ou entre participantes.

Requisitos de Acesso

O acesso a experimentos deve ser realizado mediante reserva de horário estabelecida em nome de um participante. O participante detentor da reserva pode identificar um usuário individual ou um grupo de usuários. A autenticação (*login*) é sempre efetuada por um usuário individual. No caso da reserva ter sido efetuada para um grupo, o usuário terá acesso apenas no caso de ser membro do grupo.

Cada grupo define um usuário *líder* que deterá inicialmente todos os *tokens* de acesso. Um token estabelece privilégios de acesso sobre determinado conjunto de recursos. Por exemplo, um experimento pode definir dois tokens, um para acesso ao robô e outro para acesso à câmera panorâmica.

Uma *sessão de acesso* é estabelecida quando o usuário acessa um experimento do Web Lab fornecendo suas credenciais (por exemplo, nome e senha). Caso a sessão seja estabelecida com sucesso, uma interface de controle de acesso é aberta. Esta interface possui um botão de encerramento do acesso e informa o tempo restante da sessão (atualizado periodicamente). Caso a reserva seja em nome de um grupo, a interface de controle do acesso deve possuir controles que permitam um membro do grupo reconhecer os demais membros conectados ao experimento, identificar os membros detentores de tokens, solicitar tokens e repassar tokens aos demais membros do grupo.

Um *serviço de acesso* suporta a gerência de sessões de acesso.

Requisitos de Comunicação

Uma *sessão de comunicação* deverá ser estabelecida entre os membros do grupo. Esta sessão suporta comunicação textual segundo o modelo *bate-papo (chat)*. A sessão de comunicação possui características ponto-multiponto no sentido que a informação gerada pelo participante é propagada para os demais participantes. Um *serviço de difusão* suporta a sessão de comunicação entre os participantes. Este serviço mantém uma lista de participantes, sendo responsável pela distribuição de informação entre os participantes. Esta informação pode ser tanto

de controle (por exemplo, solicitação de token) quanto de comunicação inter-pessoal (por exemplo, mensagem de bate-papo).

3 Atividades de Projeto

O projeto seguirá uma metodologia orientada a visões, por exemplo, Pontos de Vista do ODP ou Multi-Visões da SEI. Inicialmente, um levantamento de requisitos será conduzido pelo aluno. A seguir, um projeto de arquitetura será elaborado segundo as visões estabelecidas na metodologia. Finalmente, o aluno implementará a arquitetura utilizando uma tecnologia de sua escolha.

Integração com um Web Lab Existente

A implementação do aluno será testada em um Web Lab de robótica móvel existente. Para tanto, pontos de interação entre as implementações devem ser definidos. Dois pontos de interação serão estabelecidos:

1. interface WSDL da sessão de acesso existente;
2. protocolo de notificação de aquisição e cessão de tokens na interface do experimentos.

A Figura 2 ilustra estes pontos de interação.

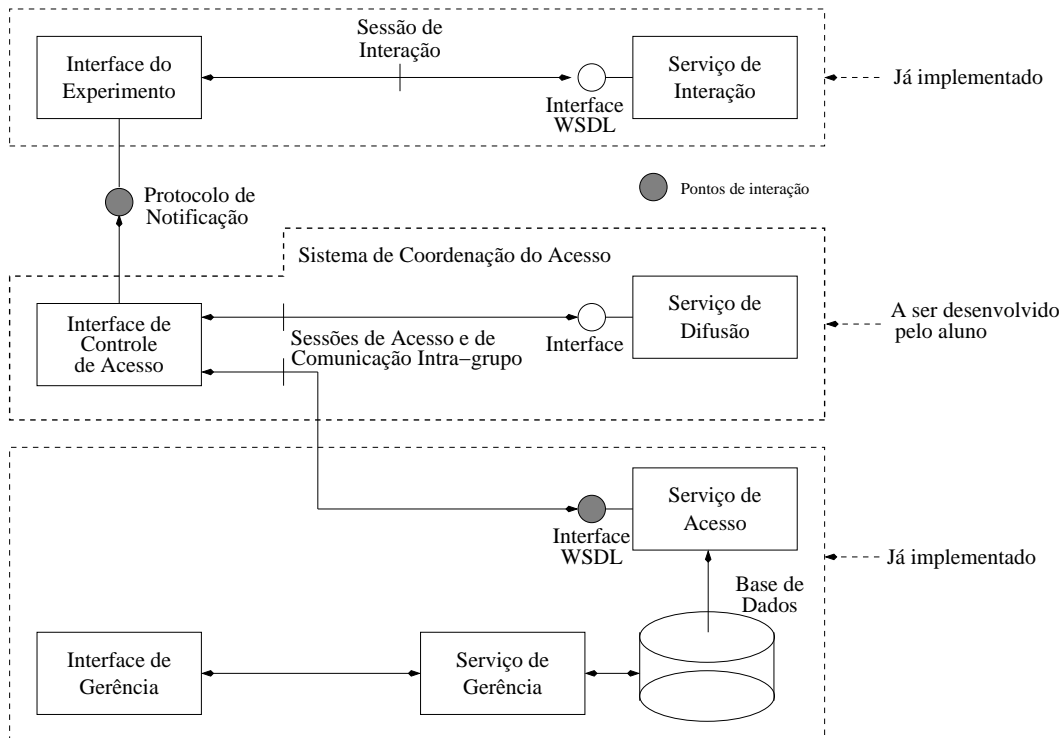


Figure 2: Interação com um Web Lab existente.

Recomendações de Projeto

As seguintes recomendações de projeto devem ser consideradas pelo aluno:

1. Além de utilizar UML como linguagem principal de projeto, o aluno poderá lançar mão de diagramas e figuras em formato livre para auxiliar na documentação do projeto.
2. O aluno deve utilizar uma ferramenta de modelagem UML como ArgoUML, Poseidon ou BOUML.
3. Na documentação do projeto o aluno deverá utilizar um documento distinto por visão.

Recomendações de Implementação

As seguintes recomendações de implementação devem ser consideradas pelo aluno:

1. O lado cliente deve ser o mais leve possível (*thin clients*). Evite invocar operações na interface do cliente (por exemplo, para fins de notificação).
2. Apesar de CORBA e RMI serem soluções aceitáveis para o projeto, lembre que estas tecnologias utilizam protocolos comumente bloqueados por *firewalls* e NAT. Se possível, opte por serviços Web.
3. Recomenda-se que a interface do cliente seja implementada como um processo Java, apesar de interfaces baseadas em navegadores Web serem igualmente aceitáveis desde que não requeiram uso de navegador específico.

4 Serviço de Acesso Existente

O serviço de acesso existente disponibiliza uma interface WSDL com métodos para iniciar, manter e finalizar uma sessão de acesso. Os serviços de acesso possuem a seguinte assinatura:

- *String createSession(String login, String password, int expid, int labid)*
- *long pingSession(String sessid)*
- *boolean endSession(String sessid)*
- *String getSessionInfo(String sessid)*

createSession inicia uma sessão de acesso para o usuário identificado por seu *login* e *password* acessar o experimento identificado por *expid* no Web Lab identificado por *labid*. *createSession* retorna um identificador de sessão (*sessid*).

pingSession deve ser invocada a cada 30 segundos para a sessão identificada por *sessid*. O serviço retorna o tempo restante da sessão em minutos. Este tempo é dado pela reserva efetuada em nome do participante (usuário ou grupo).

endSession encerra a sessão de acesso identificada por *sessid*.

getSessionInfo retorna um conjunto de informações para a sessão de acesso identificada por *sessid*. Estas informações são apresentadas em documento XML contendo:

- o identificador da sessão (sessid);
- a identidade do usuário e credenciais do usuário (papéis e permissões);
- o identificador do Web Lab e do experimento, bem como os tokens definidos para o experimento;
- o grupo que o usuário pertence, seu líder e os membros ativos (que possuem sessões em curso).

Exemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<sessionInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://143.106.50.145:8080/Cursos/IA847/01-07/XML/SessionInfo.xs
<sessid>SID:876567788</sessid>
<user>
  <id>eleri@dca.fee.unicamp.br</id>
  <name>Eleri Cardozo</name>
  <nickname>Eleri</nickname>
  <credentials>
    <role>user</role>
    <role>manager</role>
    <permission>execute</permission>
    <permission>update</permission>
  </credentials>
</user>
<weblab>
  <id>3</id>
  <experiment>
    <id>1</id>
    <tokens>
      <token>CameraToken</token>
      <token>RobotToken</token>
    </tokens>
  </experiment>
</weblab>
<group>
  <name>Group-1</name>
  <leader>eleri@dca.fee.unicamp.br</leader>
  <activeMembers>
    <user>
      <id>eliane@cenpra.gov.br</id>
      <name>Eliane Gomes</name>
      <nickname>Eliane</nickname>
      <credentials>
        <role>user</role>
        <permission>execute</permission>
      </credentials>
    </user>
  </activeMembers>
</group>
</sessionInfo>
```

A interface de controle de acesso implementada pelo aluno é responsável pela gerência da sessão de acesso (devendo, portanto, invocar os serviços de acesso para tal).

Uma vez criada com sucesso uma sessão de acesso, a sessão de interação deve ser iniciada. Esta sessão é iniciada pela interface do experimento implementada pela classe Java *Experimentif*. O construtor desta classe recebe dois parâmetros tipo int, os identificadores do experimento e do WebLab.

É de responsabilidade da sessão de acesso encerrar a sessão de interação. Isto ocorre em duas situações:

1. por ação do usuário (neste caso o serviço *endSession* deve ser invocado);
2. por expiração da reserva (quando *pingSession* retornar zero).

Em ambos os casos o método *endExperiment()* da classe *ExperimentIf* deve ser invocado. Este método encerra a sessão de interação e desabilita a interface do experimento.

5 Instancição da Interface de Controle de Acesso

A interface de controle de acesso é instanciada pelo *Java Web Start* (JWS). JSW recebe um documento JNLP (Java Network Launch Protocol) que especifica a classe a ser instanciada (*AccControllf*) e os parâmetros a serem passados no método *main*. São passados dois parâmetros: o identificador do experimento (int) e o identificador do Web Lab (int). A Figura 3 ilustra este esquema.

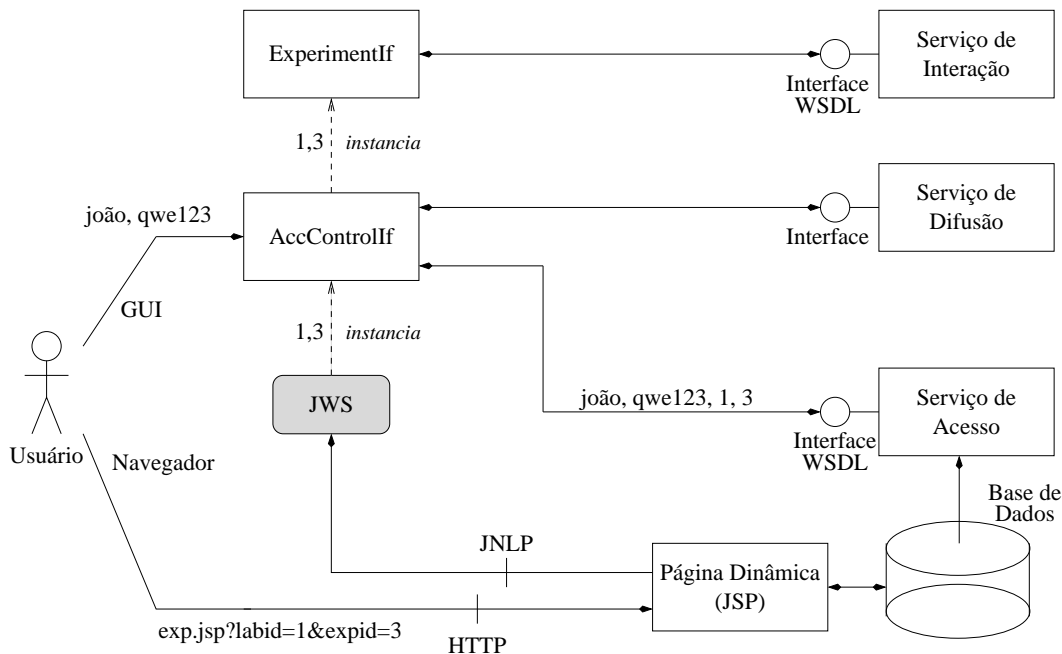


Figure 3: Instancição das interfaces.

Para facilitar os testes, o aluno irá instanciar local e manualmente (sem o JWS) sua classe *AccControllf*, passando o identificador do experimento e o identificador do Web Lab como parâmetros. Será disponibilizada ao aluno a classe *ExperimentIf* para que a mesma possa também ser instanciada localmente. Esta classe possui três métodos:

- *void endExperiment()*
- *void tokenIn(String token)*
- *void tokenOut(String token)*

O método *tokenIn* é invocado quando o controle de acesso recebe um token (o nome do token é passado como parâmetro). No caso do líder do grupo, este método deve ser invocado para cada token associado à sessão. O método *tokenOut* é invocado quando o controle de acesso devolve um token.

6 Modelo de Interação

Um modelo de interação que não exige a instalação de servidores no terminal do cliente utiliza difusão de mensagens tipo *pull*. Para simplificar o serviço de difusão, pode-se empregar a seguinte estratégia:

1. consumidores geram mensagens com um tempo de vida (*TTL* - tempo de vida);

2. o serviço armazena a mensagem por um período de tempo dado pelo parâmetro *TTL*, e acrescenta no cabeçalho um número de seqüência;
3. a informação é considerada *soft state* e deve ser confirmada a critério do produtor em intervalos de $\frac{TTL}{2}$;
4. consumidores de eventos executam operações *pull* no serviço em intervalos de $\frac{TTL}{2}$ passando como parâmetro uma expressão XPath;
5. a operação *pull* retorna todas as mensagens que satisfazem a expressão XPath.

Uma estrutura possível para o serviço de difusão é dado na Figura 4.

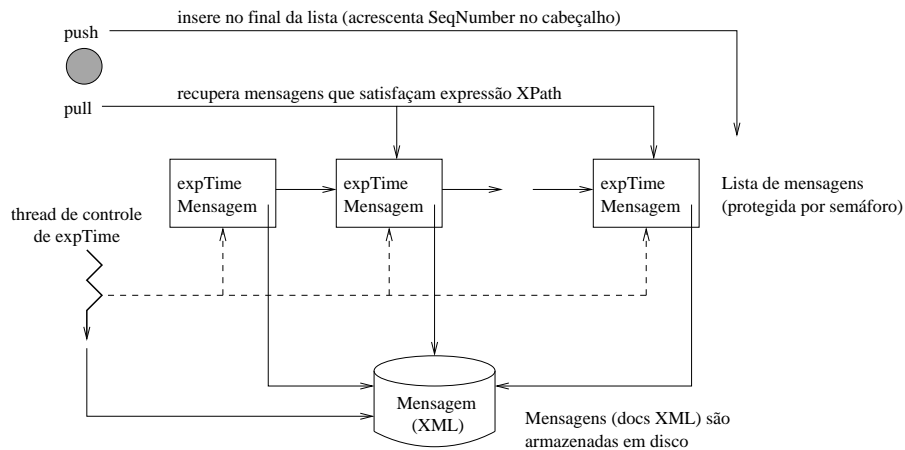


Figure 4: Possível estruturação do serviço de difusão.

Para submeter uma mensagem no serviço de difusão, o produtor invoca o seguinte método:

- *boolean push(String message)*

Os tipos de mensagens difundidas são:

- Mensagens de *chat*. Exemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<message>
  <header>
    <subject>chat</subject>
    <from>elери@dca.fee.unicamp.br</from>
    <ttl>10</ttl>
    <seq>21</seq>
  </header>
  <body>
    Precisamos alinhar o robô com a fita
  </body>
</message>
```

- Mensagem de requisição de token. Exemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<message>
  <header>
    <subject>token request</subject>
    <from>elери@dca.fee.unicamp.br</from>
    <ttl>10</ttl>
    <seq>34</seq>
  </header>
  <body>
    CameraToken
  </body>
</message>
```

- Mensagem de passagem de token. Exemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<message>
  <header>
    <subject>token pass</subject>
    <from>eliane@dca.fee.unicamp.br</from>
    <ttl>10</ttl>
    <seq>43</seq>
    <to>elери@dca.fee.unicamp.br</to>
  </header>
  <body>
    CameraToken
  </body>
</message>
```

- Mensagem de preempção de token (apenas o lider pode emitir). Exemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<message>
  <header>
    <subject>token preempt</subject>
    <from>elери@dca.fee.unicamp.br</from>
    <ttl>10</ttl>
    <seq>56</seq>
  </header>
  <body>
    CameraToken
  </body>
</message>
```

Para obter uma mensagem do serviço de difusão, o consumidor invoca o seguinte método:

- *String pull(String xpath)*

O método retorna um documento XML (eventualmente vazio) contendo mensagens que satisfazem a expressão XPath passada como argumento da operação. A título de exemplo, seja o retorno de uma operação *pull("/message[./subject = 'chat' and ./seq > 20]/body/text())*:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<messages>
  <message>
    <header>
      <subject>chat</subject>
      <from>elери@dca.fee.unicamp.br</from>
      <ttl>10</ttl>
      <seq>21</seq>
    </header>
    <body>
      Precisamos alinhar o robô com a fita
    </body>
  </message>
  <message>
    <header>
      <subject>chat</subject>
      <from>jose@dca.fee.unicamp.br</from>
      <ttl>10</ttl>
      <seq>22</seq>
    </header>
    <body>
      Deixe comigo, eu tenho o token
    </body>
  </message>
</messages>
```

Uma possível implementação dos métodos *push* e *pull* é apresentada na Figura 5.

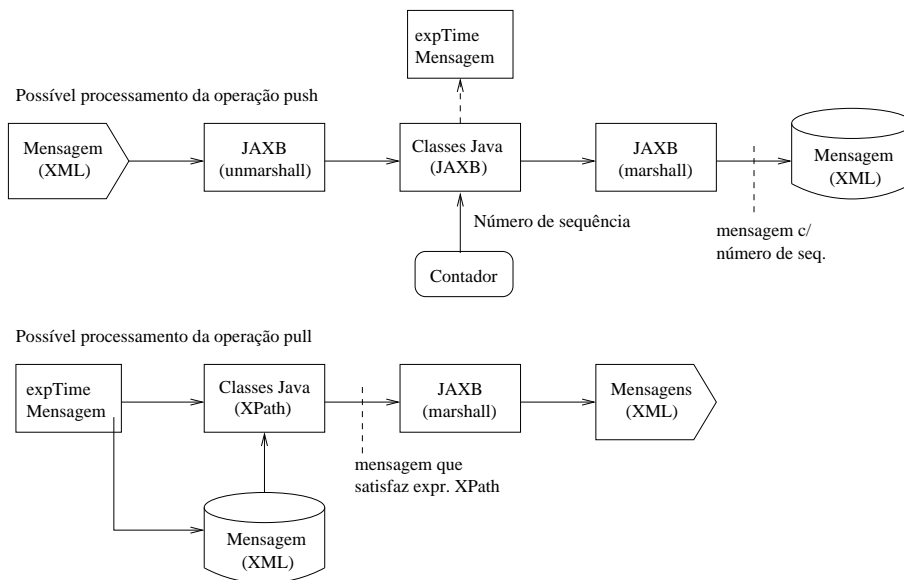


Figure 5: Possível implementação dos métodos *push* e *pull*.