Sistemas Distribuídos

10 Eventos e Notificações

- **Características**
- **Arquitetura de Eventos e Notificações**
- ***Callback* x *Polling***

Profª Ana Cristina B. Kochem Vendramin
DAINF / UTFPR

Introdução

- Idéia por trás do uso de eventos é que um objeto pode reagir à uma mudança ocorrida em outro objeto.
- Eventos causam mudanças nos objetos que mantêm o estado da aplicação.
- SDs baseados em eventos possibilitam que objetos clientes em diferentes locais registrem o seu interesse em receber notificações quando da ocorrência de determinado evento em um outro objeto.

Eventos e Notificações

- Sistemas distribuídos baseados em eventos usam o paradigma ***Publish-Subscribe***.
  - Um objeto gera eventos e os publica aos objetos que tenham se registrado para receberem notificações desses tipos de eventos.
- Notificações podem ser armazenadas, enviadas em mensagens, inquiridas e aplicadas a vários fins.
- Subscrever para um tipo de evento é também chamado de registrar interesse (registering intrest).

Características de SD baseados em eventos

- **Heterogêneo**
  - Componentes em um SD que não foram projetados para interoperar, podem trabalhar em conjunto.
  - Notificações de eventos são usadas como um meio de comunicação entre objetos distribuídos.
  - Permite conectar componentes heterogêneos na Internet.
- **Assíncrono**
  - Notificações são enviadas de forma assíncrona.
  - **Publishers** e **subscribers** são desacoplados porque diferentes usuários estão ativos em diferentes tempos.

Arquitetura de Eventos e Notificações

- Proposta por Rosenblum e Wolf [1997]
- Os principais componentes são:
  - **Um Serviço de eventos**
    - Mantém uma base de dados de eventos publicados e de assinantes interessados
  - **Objetos de interesse (OI)**
    - Publicam seus eventos
    - Objeto que experimenta mudanças de estado como resultado de suas operações serem invocadas
    - É considerado parte do serviço de evento se transmitir notificações – chamado **publisher**.

Arquitetura de Eventos e Notificações

- Os principais componentes são:
  - **Assinantes**
    - Objetos que informam ao serviço de eventos os tipos de eventos em outros objetos aos quais estão interessados
    - Quando o evento ocorre no OI uma notificação é enviada aos assinantes interessados naquele tipo de evento
  - **Objeto observador**
    - O maior propósito deste objeto é desacoplar o OI de seus assinantes
    - Pode atuar como **publisher** gerando notificações de tipos de eventos
    - Um OI pode ter vários assinantes com diferentes interesses

Arquitetura de Eventos e Notificações

- Os principais componentes são:
  - **Objeto observador (cont.)**
    - Por exemplo, assinantes podem ser diferenciados em relação aos tipos de eventos em que estão interessados ou compartilhar os mesmos eventos, mas com diferentes atributos.
    - Poderia se tornar muito complicado para o OI se ele tivesse que fazer toda a lógica para distinguir entre as necessidades dos assinantes
    - Um ou mais observadores podem ser interpostos entre o OI e os assinantes

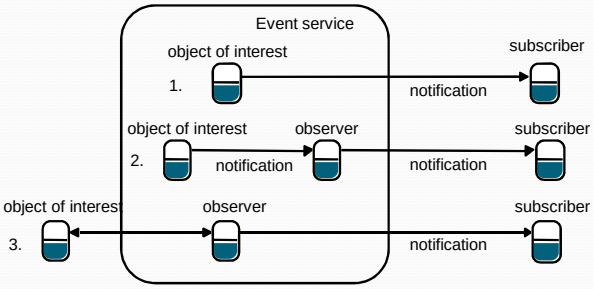
Arquitetura de Eventos e Notificações

- Papéis do **Objeto observador**
  - **Forwarding**
    - Envia todas notificações aos assinantes em nome de um ou mais OI
      - O OI fica livre da função de distribuir a informação, resolver questões relativas ao tipo de evento que o objeto assinante quer receber, etc.
    - OI passa ao observador informações sobre interesses de seus assinantes
  - **Filtering**
    - Para reduzir o número de notificações recebidas de acordo com algum predicado no conteúdo de cada notificação

Arquitetura de Eventos e Notificações

- Papéis do **Objeto observador** (cont.)
  - **Pattern of Events**
    - Um pattern especifica uma relação entre vários eventos
    - Um assinante pode estar interessado na correlação de eventos em vários OI. Ex.: notificar quando um número de OI geraram eventos
  - **Mailbox**
    - Notificações são recebidas em nome de um assinante, somente passando-as quando o assinante estiver pronto para recebe-las.
    - Assinante estabelece um mailbox quando se registra com um OI
    - Pode ser utilizado se sofrer conexões de falhas ou quando for passivado e se tornar ativo novamente.

Arquitetura de Eventos e Notificações



Arquitetura de Eventos e Notificações

- Um OI dentro do serviço de eventos sem um observador: ele manda a notificação diretamente aos assinantes.
- Um OI dentro do serviço de eventos com um observador: o OI envia a notificação via observador.
- Um OI está fora do serviço de eventos: neste caso, um observador inquire o OI a fim de descobrir quando eventos ocorrem. O observador envia as notificações aos assinantes.

Callbacks X *Polling*

- A idéia dos *callbacks* é que, ao invés do cliente fazer *polling* no servidor para descobrir se um evento ocorreu ou não, o servidor notificará seus clientes sempre que um evento ocorrer.
- O uso de *callback* evita as seguintes desvantagens do *polling*:
  - Degradação do desempenho do servidor ao receber *pollings* constantemente;
  - Clientes podem não notificar seus usuários sobre mudanças de uma maneira imediata.

Callbacks

- Desvantagens do *callback*:
  - Servidor precisa manter uma lista atualizada dos clientes interessados.
    - Como os clientes nem sempre informam ao servidor quando deixam um grupo, o servidor pode ficar com listas incorretas.
    - Isto pode ser contornado com o uso de *leasing*.
  - Necessidade do servidor realizar uma série de invocações de métodos remotos nos clientes da lista.

Referências Bibliográficas

- Coulouris, George; Dollimore, Jean; Kindberg, Tim. Distributed Systems: concepts and design. Third Edition. Addison-Wesley 2001.
- Coulouris, George; Dollimore, Jean; Kindberg, Tim; tradução João Tortello. Sistemas Distribuídos: conceitos e projeto. 4. ed. Bookman 2007.