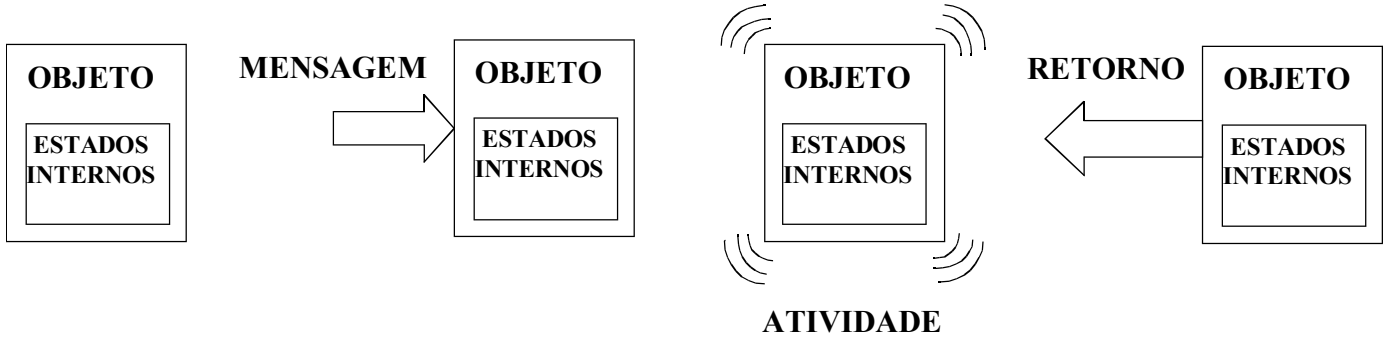
Agentes e Objetos

- Objetos (em software)
  - Metáforas computacionais que emulam características dos objetos do cotidiano
    - possuem estados internos que o descrevem
    - podem receber mensagens, a partir das quais realizam uma atividade, que é determinada conforme as mensagens enviadas e seus correspondentes parâmetros
    - podem ser classificados hierarquicamente em classes, que basicamente descrevem os estados internos e as mensagens que um objeto de uma determinada classe pode receber

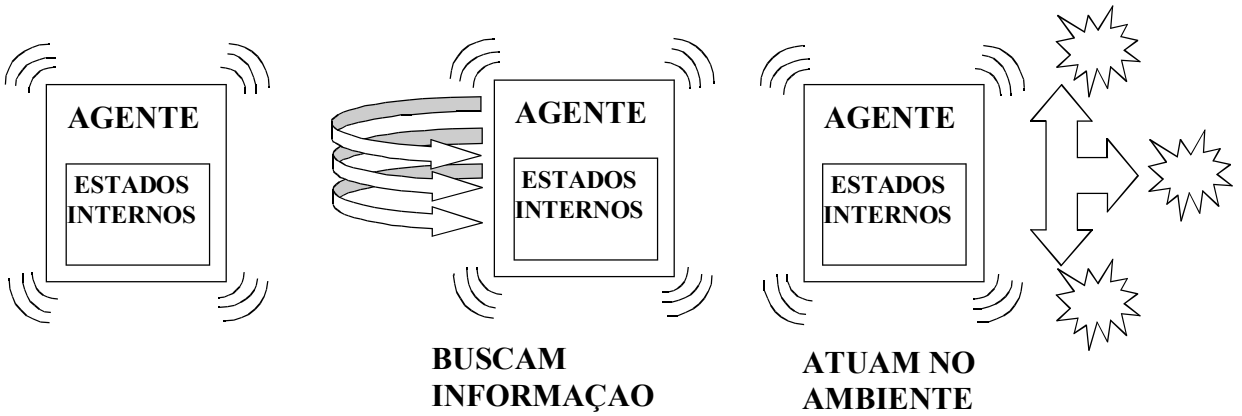


Agentes e Objetos

- Objetos = Máquinas
  - alimenta com mensagens
  - obtém o comportamento prescrito pela mensagem
- Características
  - previsibilidade do comportamento do objeto
  - cada mensagem = comportamento desejado
  - objeto não age por si só
    - responde a uma requisição de serviço
  - enquanto não está "em serviço"
    - inerte
  - objeto não "busca" mensagens ... só as recebe

Agentes e Objetos

- Agentes (em software)
  - metáforas computacionais que emulam comportamento de agentes do cotidiano
    - possuem estados internos que o descrevem
    - podem extrair dados de seu ambiente por meio de seus sensores e atuar sobre o ambiente por meio de seus atuadores
    - possuem um ciclo de vida interna por meio do qual sensoreiam e atuam

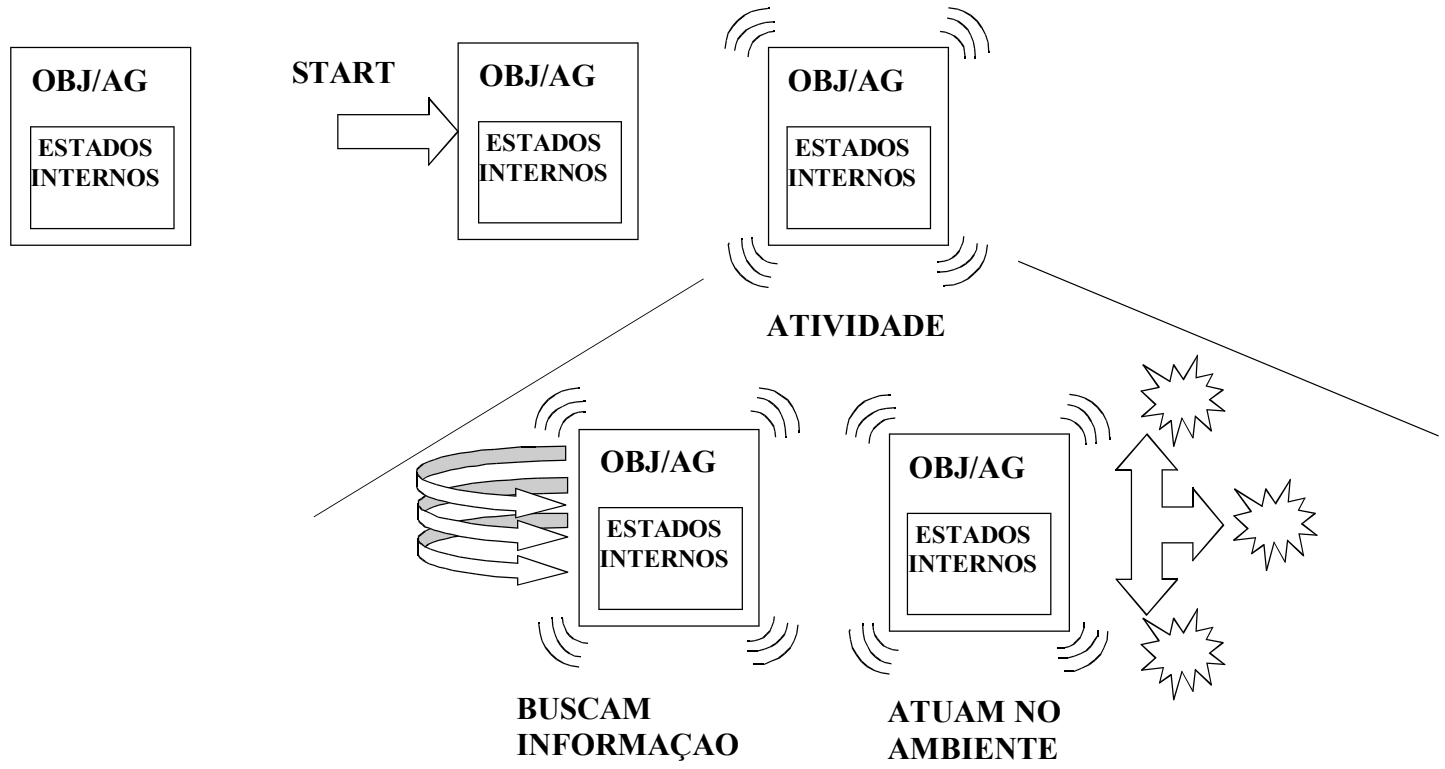


Agentes e Objetos

- Agentes = Organismos
  - mantém atividade incessante de busca por informação e atuação no ambiente
  - comportamento é determinado pela constituição do agente
- Características
  - nunca cessa atividade
  - comportamento pode ser imprevisível
  - agente age por si só
  - agente não "recebe comandos", mas busca por mensagens enviadas na forma de informações do ambiente, que são decodificadas, podendo influenciar o comportamento do agente

Agentes e Objetos

- Objetos podem ser agentes ?



Agentes e Objetos

- Distinção entre Agentes e Objetos
  - Diferentes paradigmas para a modelagem de sistemas
  - Enfoque mais adequado a cada situação
- Conveniência de cada paradigma
  - Depende da complexidade do sistema
  - Disponibilidade de linguagens adequadas para implementação
  - Tipo de comportamento que se deseja modelar
- Modelos x Linguagem
  - Agentes podem ser implementados em linguagens orientadas a objetos ?
  - Agentes demandam linguagens próprias para uma implementação adequada ?
- Modelos Orientados a Objetos x Modelos Orientados a Agentes
  - Modelos Híbridos ?

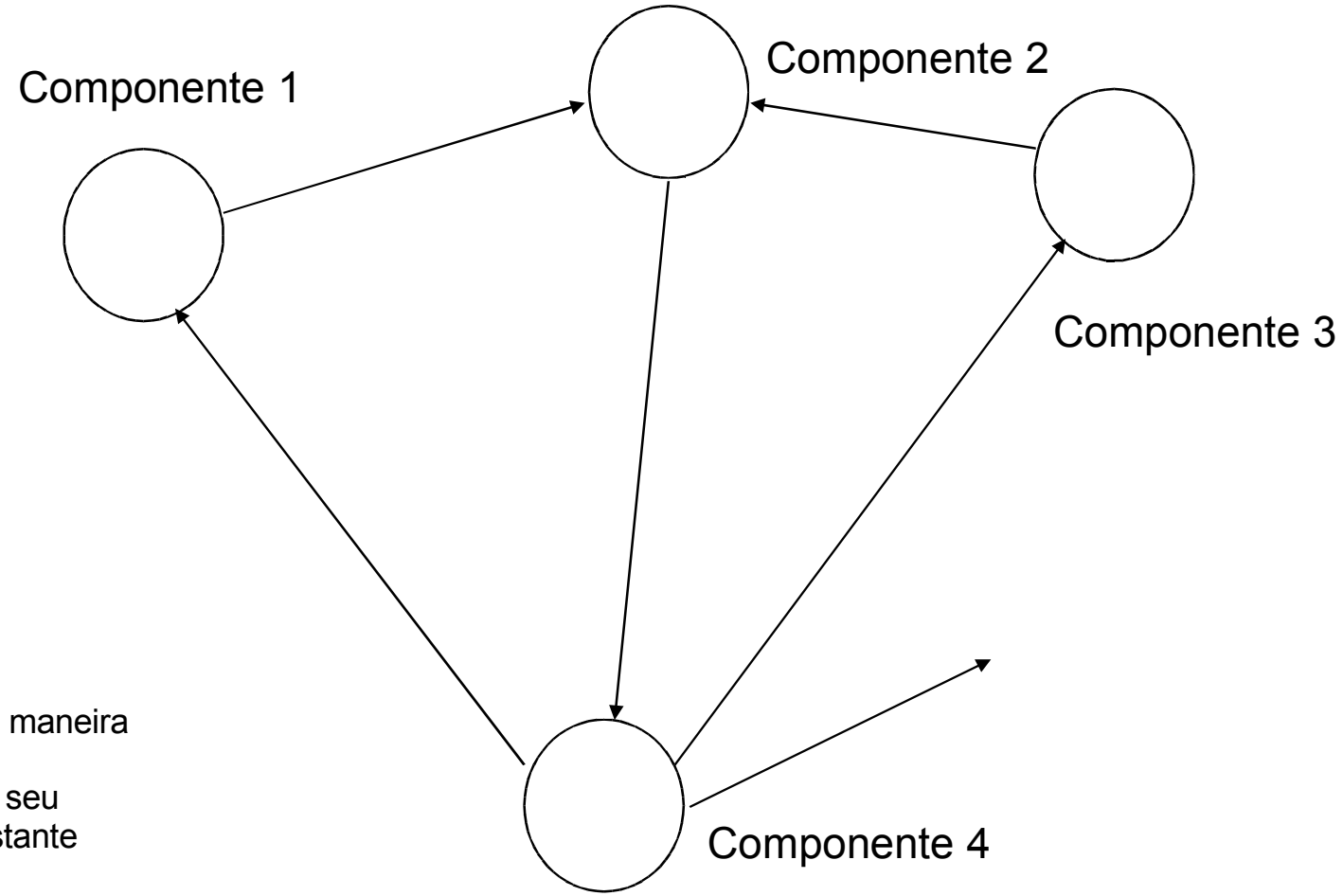
Modelos de Computação

- Modelos de Computação
  - semântica de interação entre módulos ou componentes de um sistema.
  - são usados tanto nos programas de computadores quanto no projeto de sistemas de hardware.
  - podem ser vistos como princípios organizacionais de uma especificação ou modelo de projeto.
- Exemplos
  - Processos Seqüenciais, Máquina de Estados Finitos, Máquina de Turing, Dataflow, Redes de Processos, Simuladores de Eventos Discretos, Linguagens Síncronas, Redes de Petri, Sistemas Heterogêneos, Passagem de Mensagens, etc.

Modelos de Interação entre Sub-Sistemas

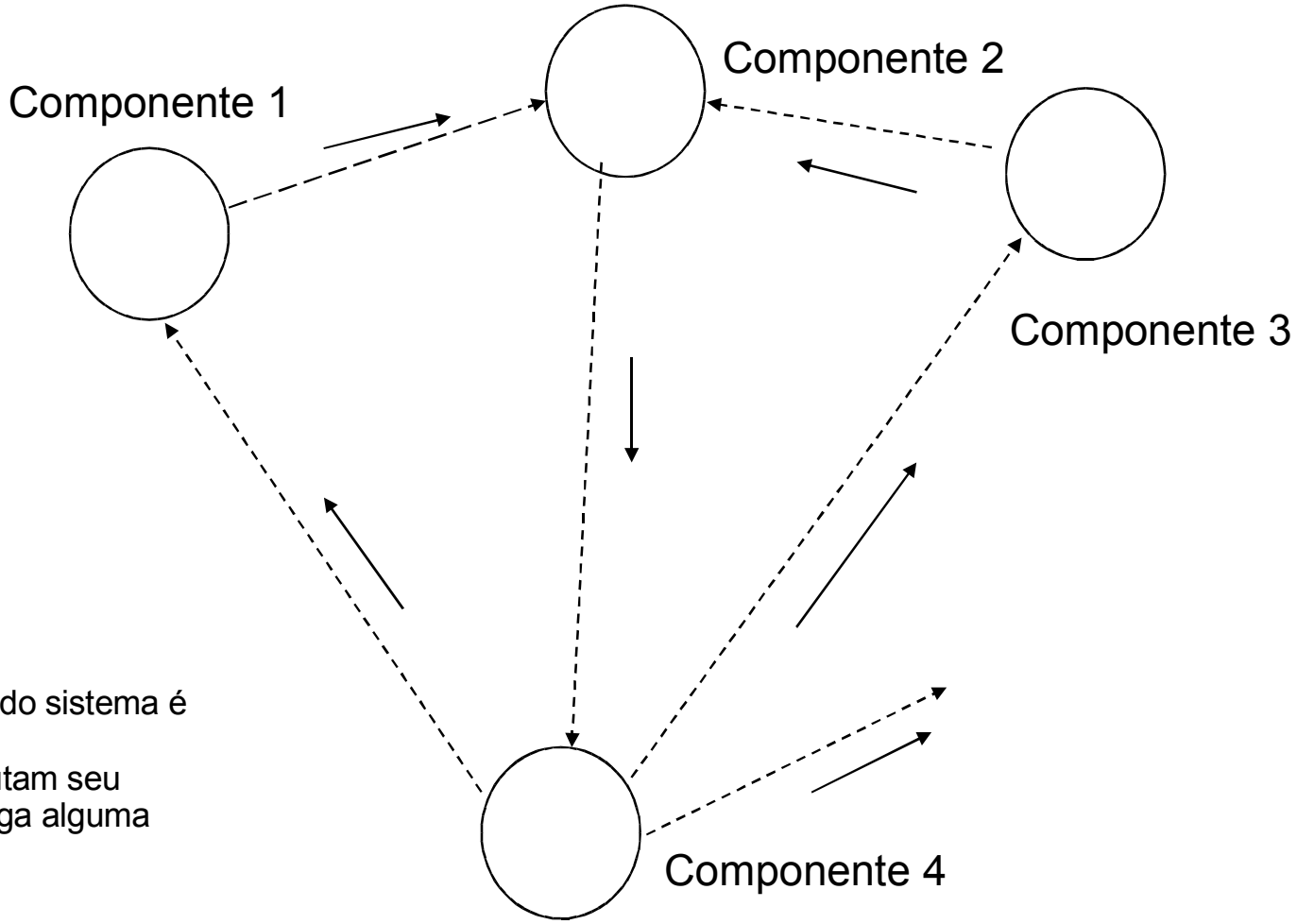
- Distinção
  - Modelo de Computação x Maneira com que o Modelo de Computação pode ser implementado.
    - Sequenciais x Concorrentes
  - Modelo de Computação x Linguagem de Programação
- Modelos de Interação entre Sub-Sistemas
  - **Dataflow**
    - Redes Neurais, Programação Estruturada, Streams
  - **Passagem de Mensagens**
    - Programação Orientada a Objetos, Teoria de Atores (Agha), Redes de Processos (Kahn)
  - **Busca por Mensagens**
    - Espaços de Tuplas (JavaSpace), Linda e Agentes

Modelos de Interação – Dataflow



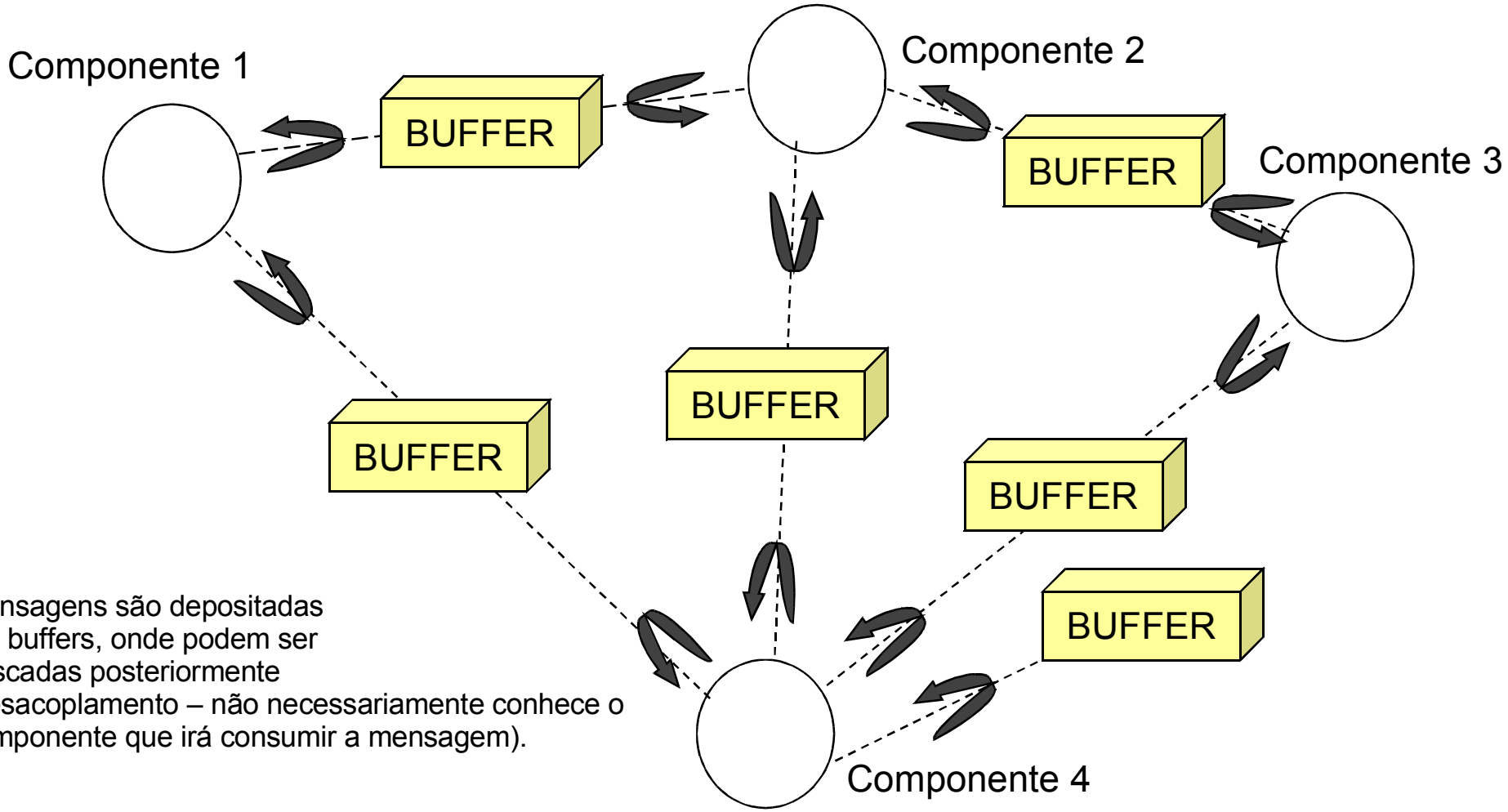
- Saídas de um componente estão diretamente acopladas às entradas de outros componentes
- Interação é realizada de maneira síncrona
- componentes executam seu processamento a todo instante

Modelos de Interação – Passagem de Mensagens



- As mensagens podem ser síncronas ou assíncronas.
- comportamento global do sistema é totalmente assíncrono
- componentes só executam seu processamento qdo chega alguma mensagem.

Modelos de Interação – Busca por Mensagens



Mensagens são depositadas em buffers, onde podem ser buscadas posteriormente (desacoplamento – não necessariamente conhece o componente que irá consumir a mensagem).

Programando Agentes em uma Linguagem Orientada a Objetos

- Objetos-Agentes
  - são criados como objetos normais de uma linguagem
    - podem criar objetos-filhos para seu próprio uso
  - são "startados"
  - permanecem em atividade até que "morram", por vontade própria
  - também podem criar agentes-filhos, com os quais mantém algum tipo de contato
- Problema
  - Um método que nunca termina paralisa o sistema
- Solução
  - Técnicas de programação concorrente

Threads e Java

- Algumas linguagens de programação
  - contém diretivas para o uso de threads
- Java
  - contém toda a infra-estrutura para a programação multi-thread
  - interpretador Java emula os threads dentro do processo ao qual pertence
  - Suporte à programação multi-thread é centralizado em torno da classe java.lang.Thread
- Classe Thread
  - permite a criação de objetos do tipo Thread, onde cada objeto terá seu próprio fluxo de controle independente

Agentes e Redes

- Redes de Computadores
  - Nível superior de concorrência que permite o desenvolvimento de aplicações em paralelo
- Modelos de Comunicação Entre Processos
  - Sockets (Passagem de Mensagens)
  - RPC - Remote Procedure Call
  - Objetos Distribuídos (CORBA e DCOM)
- Passagem de Mensagens (via sockets)
  - mecanismo mais simples para comunicação entre/com agentes
- Java
  - provê mecanismos para os três tipos de modelo de comunicação entre processos
    - Sockets, RMI, CORBA

Meu Primeiro Agente (EC1)

- Agente de Mirror
  - dados dois diretórios designados em um computador, o agente de mirror deve se encarregar em manter ambos os diretórios exatamente iguais
  - havendo uma inserção de arquivo ou subdiretório em um dos diretórios, o agente deve inserí-lo no seu diretório mirror
  - havendo modificação nos arquivos de um dos diretórios, o agente de mirror deve copiar os arquivos modificados para o diretório mirror (utilizando a data do arquivo para definir qual é o mais atual)
  - apagando-se um arquivo ou subdiretório de um dos diretórios, ele deve ser apagado também no diretório mirror

Meu Primeiro Agente (EC1)

- Requisitos do Agente de Mirror
  - O agente de Mirror deve funcionar como um daemon
  - Os diretórios que devem ser espelhados devem ser passados ao agente como um parâmetro (não devem ser embutidos no código)
  - Havendo um arquivo com o nome KILL_AG.$$$ em um dos diretórios mirror, o agente deve se matar
  - O agente somente procederá a mudanças nos diretórios quando os arquivos estiverem fechados ... ou seja, enquanto algum arquivo estiver aberto, ele não procederá a nenhuma mudança referente ao arquivo em questão
  - o agente deve utilizar algum mecanismo que evite o consumo de muito tempo de CPU, para não entrar em conflito com outros processos que estejam rodando na mesma máquina

Meu Primeiro Agente (EC1)

- Perguntas a serem respondidas no relatório
  - Qual é o ambiente deste agente ?
  - Quais são os sensores deste agente ?
  - Quais são os atuadores deste agente ?
  - Como é o ciclo operacional deste agente ?
  - Qual a melhor classificação, dentre as estudadas no curso, para este agente ? (Reflexivo, Comportamental, Planejador, Emocional, Comunicativo ou Semiótico ?)
  - Qual a aplicação deste tipo de agente ? (Robótico, Desktop, Internet, Entretenimento, etc ?)
  - Podemos dizer que este agente é inteligente ? Por quê ?