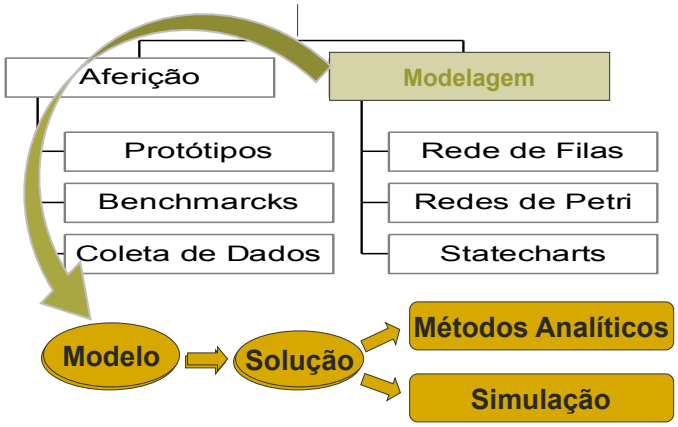
Simulação

Mário Meireles Teixeira
Departamento de Informática, UFMA
mario@deinf.ufma.br

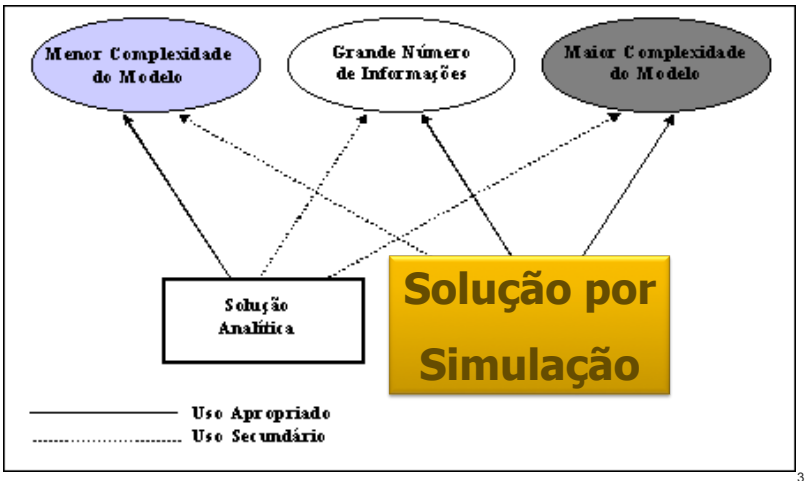
Técnicas de Modelagem

- Técnicas de Avaliação de desempenho



2

Soluções para o Modelo



3

Uso preferencial das técnicas de modelagem

Soluções para o Modelo

- **Solução por Simulação**

Modelos de simulação são programas de computador nos quais a operação de um sistema e sua carga são descritas por meio de algoritmos apropriados

4

Solução por Simulação

- **Simulação - o que é?**
  - construção de um programa computacional que implemente um modelo;
  - supõe-se que o modelo seja uma representação válida do sistema em estudo.

5

Solução por Simulação

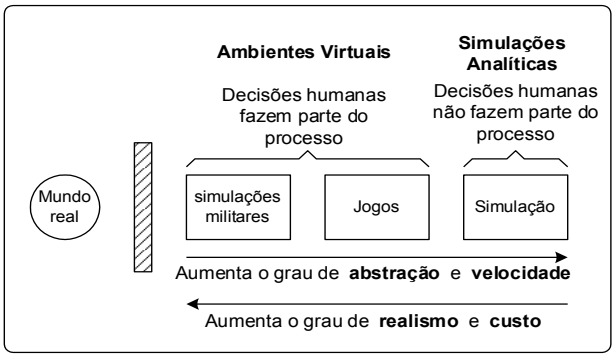
- Simulação é uma técnica bastante útil para a análise de desempenho de sistemas computacionais

- Especialmente:
  - Se o sistema não estiver disponível
  - Para prever o desempenho de diversas alternativas
  - Facilidade de realizar comparações para uma maior variedade de cargas e ambientes computacionais

6

Simulação – Emprego

- Criação de ambientes virtuais
- Avaliação de desempenho de sistemas complexos



7

Simulação - Ambientes Virtuais

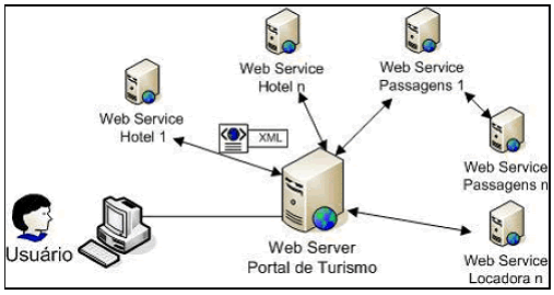
Análise Comportamental

Jogos

Simuladores

Simulação - Avaliação de Desempenho

- **Exemplo:** Simulação de um ambiente que faz reservas de passagens e hotéis e consegue adaptar-se a cargas de trabalho variáveis



Pode-se avaliar:

- Adequabilidade de um índice de medição da carga
- Utilização de diferentes arquiteturas de back-end
- Utilização de diferentes políticas de escalonamento

9

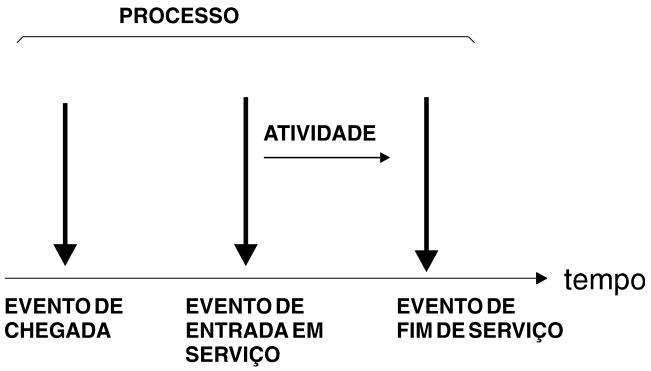
Técnicas de Simulação

	Simulação Analítica	Ambientes Virtuais
Objetivo	Análise quantitativa de sistemas complexos	Criação realista ou representação para diversão / aprendizado
Tempo	O mais rápido possível	Tempo Real
Interação humana	Observador externo	Integral, a fim de controlar os objetos
Precisão	Resultados estatisticamente corretos	Sensível à precisão humana

10

Tipos de Simulação

Processos X Eventos X Atividade



11

Tipos de Simulação

Simulação Orientada a Evento

- Simples de ser implementada
- Modelos complexos -> representação complexa

Simulação orientada a Processos

- Ambiente de simulação complexo
- Maior clareza no programa

Simulação orientada a Atividades

- Difícil de ser implementada

12

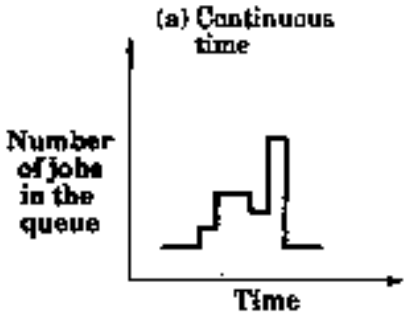
Terminologia

- **Variáveis de estado**
  - Definem o estado do sistema. Se interrompida, a simulação pode ser reiniciada somente se as variáveis de estado são conhecidas. Ex: tamanho da fila de jobs
- **Evento**
  - Mudança no estado do sistema.
    Exemplos:
    - chegada de um job;
    - início de uma nova execução;
    - partida do job.

13

Terminologia

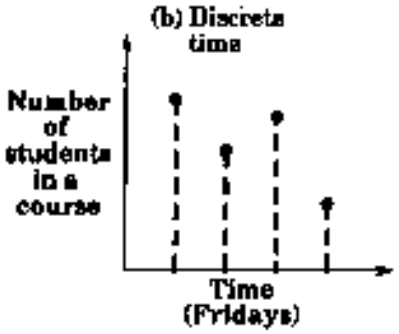
- **Modelo de Tempo Contínuo:** o estado do sistema está definido em todos os instantes.
- Ex: modelo de escalona-mento de uma CPU



14

Terminologia

- **Modelo de Tempo Discreto:** o estado do sistema está definido apenas em instantes particulares
- Ex: número de alunos presentes a cada aula



15

Terminologia

- **Modelos de Estado Contínuo e Estado Discreto**
  - Depende se as variáveis de estado são contínuas ou discretas
  - Ex: - tempo gasto estudando uma determinada matéria
    - tamanho da fila
- Modelo de estado discreto ⇔ Modelo de eventos discretos
- Modelo de estado contínuo ⇔ Modelo de eventos contínuos
- Continuidade de tempo ≠ Continuidade de estado
- Combinações possíveis:
  - estado discreto/tempo discreto
  - estado discreto/tempo contínuo
  - estado contínuo/tempo discreto
  - estado contínuo/tempo contínuo

16

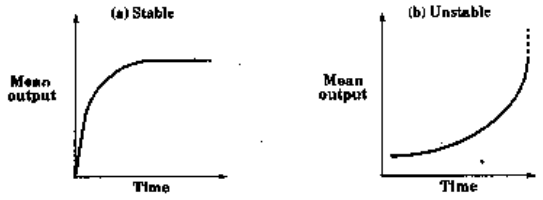
Terminologia

- **Modelos Determinísticos e Probabilísticos**
  - Nos modelos determinísticos, os resultados podem ser previstos com certeza
  - Nos modelos probabilísticos, diferentes repetições para os mesmos parâmetros de entrada produzem resultados diferentes
- **Modelos Estáticos e Dinâmicos**
  - Modelos estáticos são aqueles nos quais o tempo não é uma variável.
    Ex: $E = mc^2$ vs. Modelo de escalonamento de CPU
- **Modelos Lineares e Não-Lineares**
  - output = *f* (input)

17

Terminologia

- **Modelos Abertos e Fechados**
  - Nos modelos abertos a entrada é externa ao modelo e independente do mesmo. Nos fechados, não há entrada.
- **Modelos Estáveis e Instáveis**
  - Estável – atinge regime permanente (equilíbrio)
  - Instável – muda continuamente de comportamento



18

Modelos de Sistemas Computacionais

- Tempo contínuo
- Estados discretos
- Probabilístico
- Dinâmico
- Não-linear
- Aberto ou fechado
- Estável ou instável

19

Tipos de Simulação – Avaliação de Desempenho

- Emulação
  - Utiliza-se hardware ou firmware
  - Ex: emulador de terminal, de processador
  - Envolve basicamente questões de projeto de hardware
- Simulação de Monte Carlo
- Simulação Dirigida por Traces
- Simulação de Eventos Discretos

20

Método de Monte Carlo

- Simulação estática (sem eixo do tempo)
- Usado para modelar fenômenos probabilísticos cujas características não mudam com o tempo
- Necessita de números pseudo-aleatórios, assim como as simulações dinâmicas
- Também pode ser usado para avaliar expressões não probabilísticas através de métodos probabilísticos.
  Ex: cálculo de integrais

$$I = \int_0^2 e^{-x^2} dx$$

aproximada por:

$$x_i \sim \text{Uniforme}(0,2)$$

$$y_i = 2e^{-x_i^2}$$

$$I = E(y) = \frac{1}{n}\sum_{i=1}^{n} y_i$$

21

Simulação Dirigida por Traces

- **Trace:** registro de eventos de um sistema ordenado de acordo com o tempo.
  - Uma simulação dirigida por *traces* utiliza *traces* para gerar sua entrada
  - Exemplos: *trace* de escalonamento de jobs, de instruções de processador, de páginas referenciadas no disco, logs de servidores web, de proxies
- Método comum na avaliação de desempenho de sistemas computacionais, p.ex., para analisar algoritmos de gerenciamento/escalonamento de recursos.
  - Exemplos: paginação, caches, escalonamento de processador, de requisições HTTP, prevenção e detecção de deadlocks, balanceamento de carga

22

Simulação Dirigida por Traces: Vantagens

- Credibilidade: resultados "fáceis" de vender
- Validação fácil: pode-se comparar o resultado da simulação com a aferição do sistema em estudo
- Carga de trabalho precisa: preserva correlações e interferências na carga de trabalho
- *Trade-offs* detalhados: carga de trabalho detalhada permite verificar os efeitos de pequenas alterações nos algoritmos

23

Simulação Dirigida por Traces: Vantagens

- Menor aleatoriedade
  - *Trace* é uma entrada determinística
  - Menor número de repetições para alcançar o mesmo nível de confiança estatística
- Comparação justa: pode-se comparar diferentes alternativas usando a mesma entrada (melhor do que entrada aleatória)
- Semelhança com a implementação real: o modelo dirigido por traces é semelhante ao sistema sendo modelado

24

Simulação Dirigida por Traces: Desvantagens

- Complexidade: modelo mais detalhado
- Representatividade: as características da carga de trabalho variam com o tempo e o tipo de sistema
- Tamanho do *trace*: os *traces* são seqüências muito longas; poucos minutos enchem um disco
- Ponto único de validação: considerar o uso de mais de um *trace* (um *trace* = um cenário específico)
- Detalhes: um *trace* envolve muitos detalhes que podem atrasar a simulação
- Verificação de *trade-offs*: um *trace* é pouco flexível; difícil alterar a carga de trabalho

25

Simulação de Eventos Discretos

- Baseia-se em um modelo de estados discreto do sistema, i.e., um modelo no qual o estado do sistema assume valores discretos
- Tempo pode ser discreto ou contínuo
- Componentes da Simulação de Eventos Discretos:
  - Escalonador de eventos
  - Relógio da simulação
  - Variáveis de estado do sistema
  - Rotinas para tratamento de eventos
  - Rotinas de entrada de dados, inicialização e trace
  - Geração de relatórios
  - Gerenciamento de memória
  - Programa principal

26

Componentes da Simulação: Escalonador de Eventos

- Mantém uma Lista de Eventos Futuros (LEF) da simulação
- Atribuições do escalonador:
  - Escalonar um evento e no tempo t
  - Cancelar um evento previamente escalonado
  - Suspender um evento durante um intervalo de tempo dt
- Escalonador é um dos componentes mais solicitados durante a simulação ⇨ implicações de eficiência
- LEF pode ser:
  - Lista duplamente encadeada
  - Lista indexada (calendar queues)
  - Árvores binárias; heaps

27

Componentes da Simulação de Eventos Discretos

- **Relógio da Simulação:** Variável global que representa o tempo simulado. Avançado pelo escalonador de eventos.
  - Mecanismos de Avanço do Tempo:
    - baseado em unidades de tempo
    - dirigido a eventos
- **Variáveis de Estado:** variáveis globais que descrevem o estado do sistema.
  Ex: - número de jobs no sistema (global) ≠
  - tempo de CPU gasto por um job (local)

28

Componentes da Simulação de Eventos Discretos

- **Rotinas para Tratamento de Eventos:** cada evento possui uma rotina associada, responsável por atualizar as variáveis de estado e escalonar novos eventos
  - Três eventos:
    - chegada de um job, escalonamento e partida
- **Rotinas de entrada de dados**
  - Obtenção dos parâmetros do modelo
  - Variação dentro de uma certa faixa
- **Rotinas de inicialização**
  - Atribuem os valores iniciais das variáveis de estado
  - Inicializam as sementes

29

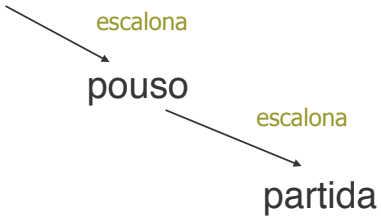
Componentes da Simulação de Eventos Discretos

- **Rotinas de trace** (acompanhamento)
  - Depuração da simulação
  - Mecanismo On/Off
- **Geração de relatórios:** resultado final da simulação
- **Gerenciamento de memória:** coleta de lixo
- **Programa Principal**
  - Inicialização
  - Execução
  - Resultados

30

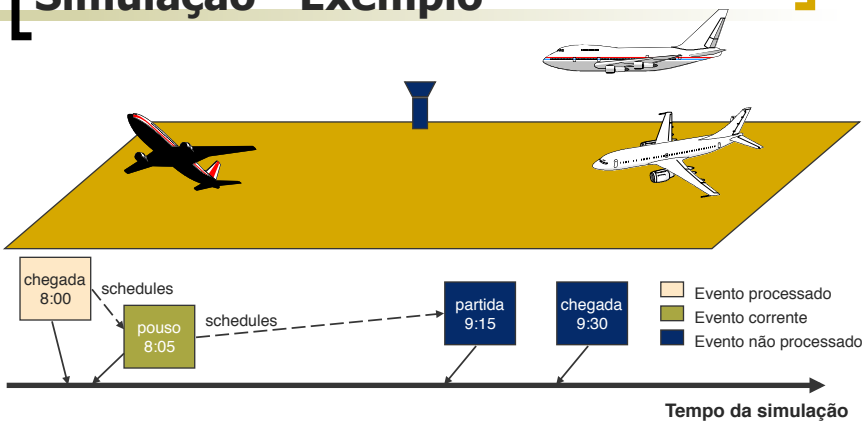
Simulação - Exemplo

- Controle de tráfego aéreo
- Simulação orientada a eventos:
  chegada do avião



31

Simulação - Exemplo



Eventos não processados são armazenados em uma lista de eventos futuros (LEF)
Eventos são processados seguindo a ordem de time stamp

32

Simulação - Exemplo

Modelo do sistema físico

Aplicação
- variáveis de estado
- código modelando o comportamento do sistema
- entrada e saída (interface com o usuário)

Chamadas para escalonamento dos eventos

Chamadas para gerenciamento dos eventos

Independe da aplicação

Simulação
- Gerencia a lista de eventos
- Avança o tempo da simulação

33

Procedimentos para gerenciamento dos eventos

```
Variáveis de estado

Integer: NoAr;
Integer: NaTerra;
```

```
Evento Chegada
{
   …
}
```

```
Evento Pouso
{
   …
}
```

```
Evento Partida
{
   …
}
```

Aplicação

Simulação

Laço de processamento dos eventos

Agora = 8:45

Lista de Eventos Futuros (LEF)

9:00 | 9:16 | 10:10

```
Enquanto (simulação não é encerrada) faça
        E = evento com menor timestamp na LEF
        Retire E da LEF
        Agora = timestamp de E
        Chamada ao manipulador do procedimento
```

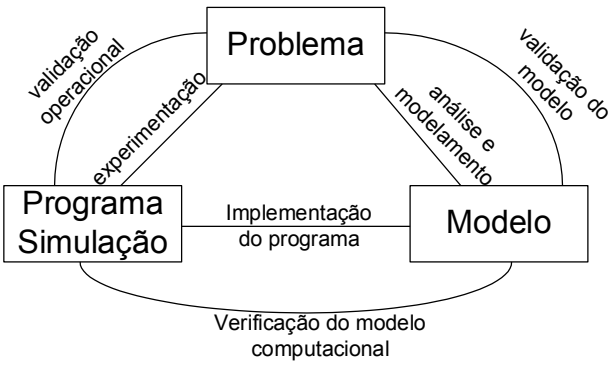
34

Fases de uma Simulação

1. estudo do sistema e definição dos objetivos;
2. construção do modelo;
3. determinação dos dados de entrada e saída;
4. tradução do modelo;
5. verificação do programa de simulação;
6. validação do programa (modelo) de simulação;
7. experimentação;
8. análise dos resultados;
9. documentação.

35

Solução por Simulação



36

Fases de uma Simulação

1. estudo do sistema e definição dos objetivos;
2. construção do modelo;
3. determinação dos dados de entrada e saída;
4. **tradução do modelo**;
5. verificação do programa de simulação;
6. validação do modelo de simulação;
7. experimentação;
8. análise dos resultados;
9. documentação.

37

Desenvolvimento de simulação

- Conhecimentos necessários para o desenvolvimento de uma simulação seqüencial
  - Modelagem
  - Programação / Linguagens para simulação
  - Probabilidade e estatística para análise dos resultados

38

Tradução do modelo

Software para simulação

- **Linguagens de programação de uso geral**
  - Construção do ambiente e do programa
- **Linguagens de simulação**
  - Aprendizado de novas linguagens
  - Oferece suporte para a implementação da simulação
  - Exemplos:
    - SIMSCRIPT (eventos)
    - GPSS, SIMULA (processos)

39

Tradução do modelo

Software para simulação

- **Extensões funcionais**
  - Uso de linguagens conhecidas
  - Implementa o ambiente
  - Disponibiliza os recursos da linguagem hospedeira
  - Exemplos: **SimPy**, SMPL, SIMPACK, Sim++
- **Pacotes de uso específico**
  - Voltados a um ou mais domínios de aplicação
  - Pouco flexíveis
  - Exemplos: NS-2, PeerSim, CloudSim

40

Exemplos de Pacotes de Simulação

Software	ARENA	OPNET Modeler	QueGAUSS	PROVISA	WorkFlow Analyzer
Proprietário	Systems Modeling Corporation	MIL 3, Inc.	Aptech Systems, Inc.	AT&T	Meta Software Corp.
Aplicações Típicas	Ferramenta de Simulação geral, molda--se a muitas aplicações diferentes.	Modelagem de protocolos de comunicação de redes.	Sistemas de filas.	Manufatura com capacidade finita de escalona-mento	Melhorias de processos de negócios
Plataformas	PC' s (DOS) e Estações (UNIX)	DEC-Alpha, HP-UX, IBM, SUN	IBM PC, SUN, IBM RISC.	IBM PC, SUN, HP.	IBM, MAC, SUN.
Preço (U$) Versão Padrão	Contato com o vendedor	25.000	275	Contato com o vendedor	10.000

41

Tradução do modelo

Software para simulação

- Ambientes para Simulação Automáticos
  - Facilitam a elaboração de programas de simulação
  - Pouco conhecimento de simulação ou programação
  - Usuário deve conhecer as técnicas de modelagem
  - Geram o programa automaticamente
  - Oferecem ferramentas que orientam o usuário na descrição, coleta de dados e análise de resultados

42

Desenvolvimento do Programa

Depende da Abordagem escolhida

1. **Atividade**
2. **Evento**
3. **Processo**

43

Desenvolvimento do Programa

Depende da Abordagem escolhida

1. Processos

- Desenvolvimento de um processo para cada recurso
- Ativação de todos os processos concorrentemente
- Problema: Sincronismo entre os processos

44

Desenvolvimento do Programa

Depende da Abordagem escolhida

2. Atividade

- Loop verificando a cada pulso de clock se existe atividade a ser executada
- Sobrecarga de processamento muito alta
- Inviável por problemas de desempenho

45

Desenvolvimento do Programa

Depende da Abordagem escolhida

3. Eventos

- Define-se uma lista de eventos futuros (LEF)
- Todos os eventos a ser executados devem estar nessa lista em ordem de timestamp
- Dentro de um loop, o próximo evento é retirado da lista, executado e eventualmente são colocados na lista novos eventos ativados pela execução do evento atual

46

Desenvolvimento do Programa

Abordagem mais utilizada para simulação de sistemas computacionais:

1. **Eventos**
   Implementação mais fácil
2. **Processos**
   Sistemas complexos geram programas mais claros

47

Aspectos importantes: Nível de detalhamento do modelo

- Nem sempre um modelo mais detalhado é o melhor modelo
- Problemas com detalhes excessivos:
  - Mais tempo de desenvolvimento do modelo
  - Maior probabilidade de erros e mais tempo gasto para identificá-los
  - Maior tempo de execução da simulação
  - Necessidade de um maior conhecimento dos parâmetros de entrada que, se não estiverem disponíveis, tornam o modelo impreciso
- Melhor introduzir detalhes no modelo aos poucos

48

Aspectos importantes

- Escolha da linguagem
  - Linguagem de simulação (SIMULA, SIMSCRIPT)
  - Linguagem de propósito geral (C, FORTRAN)
  - Extensão de uma linguagem (GASP, SMPL, SIMPACK)
  - Pacotes de simulação (QNET4, OPNET, NS-2)
- Verificação do modelo
  - Implementação do modelo não pode conter erros
- Validação do modelo
  - Modelo tem que representar corretamente o sistema real
  - Cuidado com hipóteses equivocadas!
- Tratamento das condições iniciais
  - A parte inicial da simulação (*warm-up*) geralmente não é representativa do comportamento do sistema em regime permanente (equilíbrio)

49

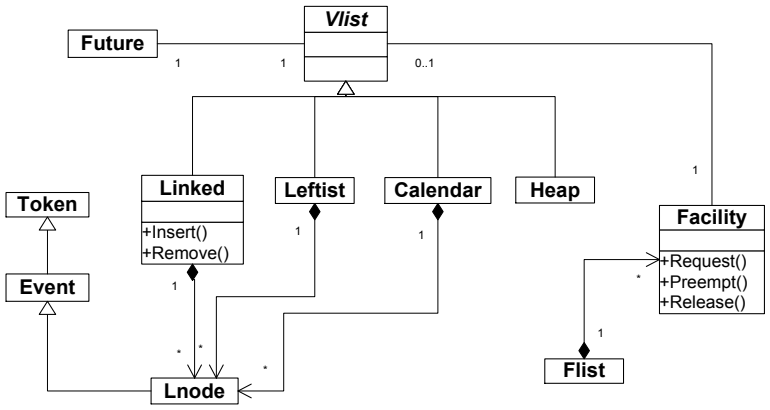
Aspectos importantes

- Duração da simulação
  - Simulações muito curtas são muito dependentes das condições iniciais e podem não representar o sistema real
- Geração de números aleatórios
  - Utilize geradores amplamente conhecidos e analisados!
- Seleção de sementes
  - Escolha cuidadosa evita correlação estatística indesejada entre diferentes rodadas da simulação
- Habilidades essenciais
  - Modelagem Estatística; Programação; Conhecimento do sistema sendo modelado
- Objetivos claros
  - O que se pretende avaliar?

50

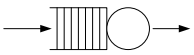
Exemplos

SimPack
Diagrama de Classes



52

Exemplo: Fila M/M/1



```
#include "queuing.h"        // Biblioteca SimPack

enum EventId {ARRIVAL, REQUEST_SERVER, RELEASE_SERVER};
const int TIME_LIMIT = 1000;

Facility * queue = NULL;    // centro de serviço (fila)

// Rotinas de tratamento de eventos
void Arrive   (void);
void RqstSrvr (void);
void RlsSrvr  (void);
```

53

Programa Principal

The image cannot be displayed. Your computer may not have enough memory to open the

```
int main ()
{
    new Future (LINKED);                    // cria LEF
    queue = new Facility ("queue");         // cria fila M/M/1
    Token customer (1);                     // primeiro cliente (token)
    Future::Schedule (ARRIVAL, 0.0, customer);      // escalona chegada

    while (Future::SimTime() < TIME_LIMIT)          // condição de parada
    {                                               //    da simulação
        Estatus es = Future::NextEvent ();          // próximo evento da LEF
        switch(es.event_id) {
            case ARRIVAL          : Arrive ();   break;
            case REQUEST_SERVER : RqstSrvr ();   break;
            case RELEASE_SERVER  : RlsSrvr ();   break;
            default               : ErrXit (1, "event_id invalido");
        }
    }
    Future::ReportStats ();                 // resultados
    return 0;
}
```

54

Tratamento de Eventos (1)


```
void Arrive ()
 {
     Token customer = Future::CurrentToken ();
     Future::UpdateArrivals ();                    // registra a chegada (λ)
     Future::Schedule (REQUEST_SERVER, 0.0, customer);
     customer.Id (customer.Id() + 1);
     // Escalona a próxima chegada
     Future::Schedule (ARRIVAL, expntl(2.0), customer);
 }

void RqstSrvr ()
 {
     Token customer = Future::CurrentToken ();     // obtém token atual
     if (queue->Request (customer) == FREE)        // solicita o recurso
      {
            double sample = expntl (2.0);
            Future::Schedule (RELEASE_SERVER, sample, customer);
      }
 }
```

55

Tratamento de Eventos (2)

```
void RlsSrvr ()
 {
     Token customer = Future::CurrentToken ();
     int who = customer.Id();
     queue->Release (who);                 // libera o recurso
     Future::UpdateDepartures ();          // registra o término (X)
 }
```

56

Resultados

Future constructed: Mon Jun 28 09:55:40 2004

SimPack SIMULATION REPORT

Total Simulation Time: 1000.000000
Total System Arrivals: 501
Total System Departures: 494

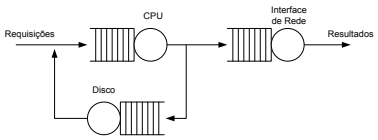
System Wide Statistics
NOTE: facility 1 'queue' has 1 busy server(s)
System Utilization: 97.7%
Arrival Rate: 0.501000, Throughput: 0.494000
Mean Service Time per Token: 1.976938
Mean # of Tokens in System: 10.270485 Largest FEL size was 3
Mean Residence Time for each Token: 20.790456

Facility Statistics
F 1 (queue): Idle: 2.3%, Util: 97.7%, Preemptions: 0, LongestQ: 26

57

Exemplo: Servidor Web com múltiplas classes

```
#include "queuing.h“     // Biblioteca SimPack

// Definição de constantes
const long          TEMPO_SIMUL = 1000;
const int           VISITAS_DISCO = 1;
const double        INTERVALO_CHEG = 0.2;
const int           NUMCLASS = 4;

const double media_cpu[4]     = {0.375, 0.325, 0.2, 0.075};
const double media_disco[4]   = {0.175, 0.175, 0.175, 0.125};
const double media_nic[4]     = {0.2, 0.225, 0.25, 0.525};

const double p[NUMCLASS] = {0.35, 0.85, 0.99, 1.00};

enum TipoEvento {CHEGADA, REQ_CPU, LIBERA_CPU, REQ_DISCO,
                 LIBERA_DISCO, REQ_NIC, LIBERA_NIC};

// Variáveis globais
Facility *cpu   = NULL,     // centros de serviço (filas)
         *disco = NULL,
         *nic   = NULL;

Token cliente;    // token da simulação
```

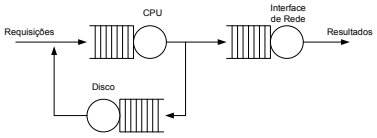


58

Programa Principal

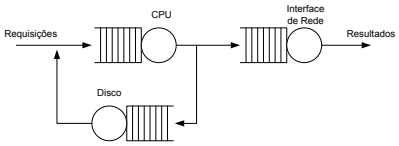
```
int main()
{
   new Future(LINKED);
   cpu = new Facility("CPU", 1);                    // repetir para disco e rede
   . . .
   cliente.Id(1);                                   // primeiro token da simulação
   cliente.SetTokenIattr(0, VISITAS_DISCO);         // attr0 = VISITAS_DISCO
   int classe = GeraClasReq();
   cliente.SetTokenIattr(1, classe);                // attr1 = Classe Requisicao

   Future::Schedule(CHEGADA, 0.0, cliente);         // primeira chegada de cliente
   while (Future::SimTime() < TEMPO_SIMUL)
   {
      Estatus es = Future::NextEvent();   cliente = es.token;
      switch(es.event_id)
      {
          case CHEGADA      : Chegada();      break;
          case REQ_CPU      : ReqCpu();       break;
          . . .
          default           : ErrXit (1, "event_id invalido");
      } }
   . . .
    Future::ReportStats();     // imprime o relatório da simulação
```


Tratamento de Eventos (1)




```
void Chegada()
{
   Future::Schedule(REQ_CPU, 0.0, cliente);
   Future::UpdateArrivals();

    // Escalona a chegada do proximo cliente
    cliente.Id(cliente.Id() + 1);
    cliente.SetTokenIattr(0, VISITAS_DISCO);
    int classe = GeraClasReq();
    cliente.SetTokenIattr(1, classe);
    Future::Schedule(CHEGADA, INTERVALO_CHEG, cliente);
}

void ReqCpu()
{
   if (cpu->Request(cliente) == FREE)
   {
      int classe = cliente.TokenIattr(1);
      double media = media_cpu[classe] / (VISITAS_DISCO + 1);
      Future::Schedule(LIBERA_CPU, expntl(media), cliente);
   }
}
```

Tratamento de Eventos (2)




```
void LiberaCpu()
{
   int id = cliente.Id();
   cpu->Release(id);

   int num_visitas = cliente.TokenIattr(0);      // obtém núm. visitas ao disco
   if (num_visitas > 0)                          // cliente ainda não terminou
      Future::Schedule(REQ_DISCO, 0.0, cliente);  // escalona o disco
   else
      Future::Schedule(REQ_NIC, 0.0, cliente);      // escalona a Intf. de rede
}

. . .

void LiberaNic()
{
   int id = cliente.Id();
   nic->Release(id);
   Future::UpdateDepartures();    // atendimento concluído
}
```

61

Resultado da Simulação (1)

```
Future constructed: Wed Jun 23 10:46:30 2004

SimPack SIMULATION REPORT
-------------------------------------------
Total Simulation Time: 1000.000000
Total System Arrivals: 5001
Total System Departures: 2584

System Wide Statistics
------------------------------
NOTE: facility 1 'CPU' has 1 busy server(s)
NOTE: facility 2 'DISCO' has 1 busy server(s)
System Utilization: 73.8%
Arrival Rate: 5.001000, Throughput: 2.584000
Mean Service Time per Token: 0.285647
Mean # of Tokens in System: 1216.283719    Largest FEL size was 5
Mean Residence Time for each Token: 470.698033
```

62

Resultado da Simulação (2)

```
Facility Statistics
----------------------
F 1 (CPU): Idle:  0.0%, Util: 100.0%, Preemptions: 0, LongestQ: 2414
F 2 (DISCO): Idle: 37.0%, Util: 63.0%, Preemptions: 0, LongestQ: 13
F 3 (REDE): Idle: 41.5%, Util: 58.5%, Preemptions: 0, LongestQ: 9

Parametros da Simulacao (SERVWEB1.CPP)
-------------------------------------
Intervalo entre chegadas....: 0.200000
Classes.....: 0.35 0.85 0.99 1.00
```

63