



**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ**  
**CAMPUS DE CURITIBA**  
**GERÊNCIA DE PESQUISA E PÓS-GRADUAÇÃO**  
**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E**  
**INFORMÁTICA INDUSTRIAL – CPGEI**

**JOÃO LUIZ LUGUESI**

**AMBIENTE DE APOIO AO ENSINO E APRENDIZADO**  
**DO ESCALONAMENTO EM SISTEMAS EM TEMPO REAL**

**DISSERTAÇÃO DE MESTRADO**

**CURITIBA**  
**NOVEMBRO - 2006**



**DISSERTAÇÃO**

apresentada à UTFPR  
para obtenção do grau de

**MESTRE EM CIÊNCIAS**

por

**JOÃO LUIZ LUGUESI**

---

**AMBIENTE DE APOIO AO ENSINO E APRENDIZADO  
DO ESCALONAMENTO EM SISTEMAS EM TEMPO REAL**

---

Banca Examinadora:

Presidente e Orientador:

Prof. Dr. Douglas P. B. Renaux UTFPR

Examinadores:

Prof. Dr. Roberto A. Hexsel UFPR

Prof. Dra. Keiko V. Ono Fonseca UTFPR

Prof. Dr. Jean Marcelo Simão UTFPR

Curitiba, 29 de Novembro de 2006.



**João Luiz Lugeski**

**AMBIENTE DE APOIO AO ENSINO E APRENDIZADO  
DO ESCALONAMENTO EM SISTEMAS EM TEMPO REAL**

Dissertação apresentada no Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de “Mestre em Ciências” – Área de Concentração: Informática Industrial.

Orientador: Prof. Dr. Douglas P.B. Renaux

Curitiba

2006

L951a Luguesi, João Luiz

Ambiente de apoio ao ensino e aprendizado do escalonamento em sistemas em tempo real / João Luiz Luguesi. Curitiba. UTFPR, 2006  
XXXIX, 208 f. : il. ; 30 cm

Orientador: Prof. Dr. Douglas P. B. Renaux

Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial. Curitiba, 2006

Bibliografia: f. 139-142

1. Tecnologia educacional. 2. Laboratório remoto. 3. Ensino a distância. 4. Sistemas em tempo real. I. Renaux, Douglas P. B. II. Universidade Tecnológica Federal do Paraná. Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial. III. Título

CDD: 371.3078

„Die Menschen drängen sich zum Lichte,  
nicht um besser zu sehen,  
sondern um besser zu glänzen.“

Friedrich Wilhelm Nietzsche





## **DEDICATÓRIA**

A meus pais João e Herta (*in memoriam*), meus filhos Carlos Augusto e Luiz Fernando, e meus irmãos Carlos Pedro e Martin Augusto, que de certa forma, cada um a seu modo, participaram desta jornada.



## AGRADECIMENTOS

Gostaria de expressar profunda gratidão ao prof. Douglas por ter me aceito como orientado, pela participação de seus conhecimentos, pela seriedade no trabalho, pelo rigor técnico e científico e pela segurança que transmite. Muito obrigado.

Gostaria de agradecer pelo encontro de uma pessoa que posso chamar de amigo, e que muitas vezes disse: "Vá em frente, não desista!". Muito obrigado Nico.

Gostaria de agradecer aos amigos e companheiros de laboratório pelas idéias, pela força, pelo apoio e pela possibilidade. Muito obrigado a todos vocês, em especial, Evan, Andrey, Erik e Jean.

Gostaria de agradecer aos professores do CPGEI, que de uma ou de outra forma contribuem com o aumento e difusão do conhecimento. Muito obrigado prof. Douglas, prof. Keiko, prof. Stadzisz, prof. Emilio, prof. Luiz Allan e prof. Flávio.

Gostaria de agradecer à UTFPR pelo suporte e pela infra-estrutura de laboratório disponibilizados. Muito obrigado.

Enfim, gostaria de agradecer aos meus pais e irmãos pelo empenho, ora material, ora emocional, e a meus filhos cuja abdicação às vezes beirava o sacrifício. De outro modo não teria sido possível ... Muito obrigado.



## SUMÁRIO

SUMÁRIO .....	IX
LISTA DE ALGORITMOS .....	XV
LISTA DE FIGURAS .....	XVII
LISTA DE QUADROS .....	XXI
LISTA DE TABELAS.....	XXIII
TABELA DE REDUÇÕES .....	XXV
TABELA DE SÍMBOLOS .....	XXIX
TABELA DE TERMOS .....	XXXIII
RESUMO .....	XXXVII
<i>ABSTRACT</i> .....	XXXIX
1 INTRODUÇÃO.....	1
1.1 OBJETO DE ESTUDO .....	1
1.2 OBJETIVO E ESCOPO .....	2
1.3 JUSTIFICATIVAS.....	2
1.4 RESULTADOS.....	3
1.5 CONTEÚDO .....	3
2 SISTEMAS EM TEMPO REAL .....	5
2.1 INTRODUÇÃO.....	5
2.1.1 Computação em tempo real .....	5
2.2 GRAUS DE PONTUALIDADE.....	6
2.2.1 Tarefas em tempo real severo .....	7
2.2.2 Tarefas em tempo real rígido.....	7
2.2.3 Tarefas em tempo real fraco .....	7
2.3 MODELO DE APLICATIVOS EM TEMPO REAL.....	8
2.3.1 Ciclo de vida de tarefas .....	8
2.3.2 Ciclo de vida de serviços.....	11
2.3.2.1 Estado do serviço PRONTO.....	11

2.3.2.2 Estado do serviço EXECUTANDO .....	12
2.3.2.3 Estado do serviço SUSPENSO/BLOQUEADO .....	13
2.3.3 Diagramas de Gantt para o ciclo de vida de serviços.....	13
2.3.4 Transações.....	15
2.4 MEDIDAS DE TEMPOS EM SERVIÇOS.....	16
2.4.1 Definição das medidas de tempos .....	17
2.4.2 Valoração das medidas de tempos .....	21
2.5 CLASSIFICAÇÃO DE TAREFAS.....	23
2.5.1 Tarefas periódicas .....	23
2.5.2 Tarefas aperiódicas.....	24
2.5.3 Tarefas esporádicas .....	24
2.6 PRIORIDADES E FILAS.....	25
2.6.1 Prioridade estrita .....	25
2.6.2 Inversão de prioridade.....	26
2.6.3 Impasse de bloqueio .....	26
2.7 AMBIENTES PREEMPTIVOS E NÃO-PREEMPTIVOS.....	27
2.7.1 Escalonamento não-preemptivo .....	27
2.7.2 Escalonamento preemptivo .....	28
2.7.3 Escalonamento com preempção diferida .....	29
2.8 AMBIENTES N-PROCESSADORES .....	30
2.8.1 Acoplamento fraco .....	31
2.8.2 Acoplamento forte.....	31
2.8.3 Ambientes homogêneos e heterogêneos .....	32
2.9 CONCLUSÃO.....	32
3 FUNDAMENTOS DO ESCALONAMENTO.....	35
3.1 INTRODUÇÃO .....	35
3.1.1 Classificação dos escalonadores.....	36
3.1.2 Exemplos de políticas de escalonamento .....	37
3.1.2.1 Primeiro a Chegar Primeiro Atendido.....	37

3.1.2.2 Round Robin.....	37
3.1.2.3 Executivo Cíclico .....	38
3.1.2.4 Prazo Mais Próximo Primeiro .....	40
3.1.2.5 Menor Tempo de Relaxamento Primeiro .....	41
3.1.2.6 Serviço Mais Curto Primeiro.....	42
3.1.2.7 Maior Prioridade Primeiro.....	43
3.2 ESCALONAMENTO ESTÁTICO .....	43
3.3 ESCALONAMENTO DINÂMICO.....	44
3.3.1 Escalonamento dinâmico com prioridade fixa .....	45
3.3.1.1 Escalonamento dinâmico com prioridade fixa não-preemptivo...45	
3.3.1.2 Escalonamento dinâmico com prioridade fixa preemptivo .....	46
3.3.2 Escalonamento dinâmico com prioridade dinâmica.....	47
3.4 ANÁLISES DE ESCALONABILIDADE .....	48
3.4.1 Análise com tarefas independentes .....	49
3.4.1.1 Análise pela utilização do processador .....	49
3.4.1.2 Análise pela carga do processador .....	54
3.4.1.3 Análise pelo tempo de resposta .....	55
3.4.2 Análise com bloqueios .....	56
3.4.2.1 Valoração do tempo de bloqueio.....	58
3.4.2.2 Protocolo de Herança de Prioridade .....	59
3.4.2.3 Protocolo de Teto de Prioridade.....	63
3.4.2.4 Protocolo de Trava Máxima.....	65
3.4.3 Análise não-preemptiva.....	67
3.4.4 Flutuação de liberação .....	68
3.4.5 Deslocamento .....	70
3.4.6 Prazos no período .....	71
3.4.7 Prazos arbitrários .....	71
3.5 ATRIBUIÇÃO DE PRIORIDADES .....	74
3.5.1 Algoritmo de atribuição de prioridades.....	74

3.6 LEVANTAMENTO DE FERRAMENTAS .....	75
3.7 CONCLUSÃO.....	77
4 DESCRIÇÃO DA FERRAMENTA.....	79
4.1 REQUISITOS PARA A FERRAMENTA .....	79
4.2 DESENVOLVIMENTO DA FERRAMENTA .....	81
4.2.1 Módulo de criação e edição de fontes .....	83
4.2.1.1 Editor de fontes textuais.....	83
4.2.1.2 Editor de fontes XML .....	85
4.2.1.3 Linguagem <i>AnimaTi</i> .....	86
4.2.1.4 Construtos para algoritmos.....	89
4.2.1.5 Mutantes .....	90
4.2.2 Módulo de compilação e carga de arquivos fonte.....	92
4.2.2.1 Compilador.....	93
4.2.2.2 Gerador reverso .....	95
4.2.2.3 Gerador reverso XML .....	95
4.2.2.4 Carregador XML .....	96
4.2.2.5 Visor de fatos de análise.....	96
4.2.2.6 Árvore de sintaxe .....	97
4.2.2.7 Exemplo de segmentação .....	99
4.2.3 Módulo de tradução de árvores de sintaxe .....	101
4.2.3.1 Gerador de grafos de segmentos .....	101
4.2.3.2 Gerador de planilhas de tempos .....	103
4.2.3.3 Grafo de segmentos.....	106
4.2.3.4 Diagrama de segmentos .....	107
4.2.4 Módulo de análise e simulação de escalonamento.....	108
4.2.4.1 Editor de planilhas de tempos .....	109
4.2.4.2 Analisador de escalonamento.....	110
4.2.4.3 Simulador de escalonamento.....	111
4.2.4.4 Visor de detalhes de serviços .....	114



4.2.4.5 Planilha de tempos.....	115
4.2.4.6 Gráfico de tempos.....	116
4.2.4.7 Outros gráficos e saídas.....	118
4.2.5 Módulo de simulação de tarefas e rotinas .....	119
4.2.5.1 Simulador de tarefas e rotinas .....	119
4.2.5.2 Animador de gráficos de tempos.....	119
4.2.5.3 Diário de simulação.....	120
4.3 IMPLEMENTAÇÃO DA FERRAMENTA.....	121
4.4 CONCLUSÃO .....	121
5 ESTUDOS DE CASOS.....	123
5.1 CASO 1: EXPERIMENTO COMPLETO.....	123
5.2 CASO 2: SIMULAÇÃO DE EXEMPLOS DESTE TRABALHO .....	127
5.3 CASO 3: SIMULAÇÃO DE EXEMPLOS DA LITERATURA.....	128
5.4 CASO 4: INVERSÃO DE PRIORIDADE.....	129
5.5 CASO 5: ANÁLISE E SIMULAÇÃO DE ESCALONAMENTO .....	131
5.6 CASO 6: IMPASSE DE BLOQUEIO.....	132
5.7 CONCLUSÃO .....	133
6 CONCLUSÃO.....	135
6.1 RESULTADOS.....	135
6.2 CONTRIBUIÇÕES.....	136
6.3 TRABALHOS FUTUROS.....	136
REFERÊNCIAS BIBLIOGRÁFICAS.....	139
APÊNDICE A LINGUAGEM <i>ANIMATI</i> .....	143
APÊNDICE B DEFINIÇÃO DE ARQUIVOS FONTE XML .....	161
APÊNDICE C ELEMENTOS DO DIAGRAMA DE SEGMENTOS.....	167
APÊNDICE D ELEMENTOS DO DIAGRAMA DE NS.....	175
APÊNDICE E ESQUEMAS DE NODOS PARA GRAFOS DE SEGMENTOS.....	181
APÊNDICE F FERRAMENTAS PARA ANÁLISE E SIMULAÇÃO .....	187



## LISTA DE ALGORITMOS

ALGORITMO 2.1: Algoritmo para tarefas com intervalo de ativação fixo .....	10
ALGORITMO 2.2: Algoritmo para tarefas com ativação aleatória.....	10
ALGORITMO 3.1: Exemplo de um escalonador executivo cíclico .....	40
ALGORITMO 3.2: Algoritmo para atribuir prioridades (sem bloqueios).....	75
ALGORITMO 4.1: Exemplo com o construto <code>bloco...fim</code> .....	87
ALGORITMO 4.2: Exemplo com o construto do ator <code>se...fim</code> .....	94
ALGORITMO 4.3: Exemplo de fonte com divisão em segmentos .....	99
ALGORITMO 4.4: Exemplo de tempos pessimistas/otimistas .....	105
ALGORITMO 4.5: Funcionamento básico do simulador de escalonamento .....	113



## LISTA DE FIGURAS

FIGURA 2.1: Graus de pontualidade em sistemas em tempo real.....	6
FIGURA 2.2: Ciclo de vida de tarefas .....	9
FIGURA 2.3: Ciclo de vida básico de serviços.....	11
FIGURA 2.4: Diagrama de Gantt dos serviços no tempo .....	14
FIGURA 2.5: Explicação da cores para os diagramas de Gantt .....	15
FIGURA 2.6: Exemplo de transação formada por tarefas .....	16
FIGURA 2.7: Principais medidas de tempos em serviços .....	17
FIGURA 2.8: Tempo de flutuação de liberação em serviços.....	20
FIGURA 2.9: Tempo de deslocamento de liberação em serviços .....	20
FIGURA 2.10: Tempo de flutuação de término em serviços.....	21
FIGURA 2.11: Valoração de medidas de tempo.....	21
FIGURA 2.12: Distribuição dos serviços de tarefas no tempo .....	23
FIGURA 2.13: Exemplo de impasse de bloqueio .....	26
FIGURA 2.14: Exemplo de escalonamento não-preemptivo.....	27
FIGURA 2.15: Exemplo de escalonamento preemptivo.....	29
FIGURA 2.16: Relação entre processadores e tarefas .....	31
FIGURA 3.1: Exemplo de escalonamento <i>Round Robin</i> .....	38
FIGURA 3.2: Exemplo de escalonamento com o Executivo Cíclico .....	39
FIGURA 3.3: Exemplo escalonamento EDF preemptivo .....	41
FIGURA 3.4: Definição de tempo de relaxamento .....	41
FIGURA 3.5: Tempos de execução estimados .....	42
FIGURA 3.6: Exemplo de tarefas escalonáveis por encaixe perfeito .....	50
FIGURA 3.7: Exemplo de tarefas não escalonáveis .....	51
FIGURA 3.8: Variação de $U(n)$ em função de $n$ .....	52
FIGURA 3.9: Exemplo de tarefas sem herança de prioridade .....	58
FIGURA 3.10: Situações sob o Protocolo de Herança de Prioridade .....	59
FIGURA 3.11: Exemplo de Protocolo de Herança de Prioridade.....	61

FIGURA 3.12: Distribuição de seções críticas por tarefa .....	62
FIGURA 3.13: Combinações de seções críticas .....	62
FIGURA 3.14: Exemplo de Protocolo de Teto de Prioridade .....	63
FIGURA 3.15: Exemplo de Protocolo de Trava Máxima.....	66
FIGURA 3.16: Exemplo de tarefas em ambiente não-preemptivo .....	68
FIGURA 3.17: Exemplo de tarefas com flutuação de liberação.....	69
FIGURA 3.18: Exemplo de tarefas com deslocamento de fase de liberação .....	70
FIGURA 3.19: Exemplo de tarefas com prazos arbitrários .....	73
FIGURA 4.1: Modelo funcional da ferramenta <i>AnimaTi</i> .....	82
FIGURA 4.2: Visores do editor de fontes textuais .....	84
FIGURA 4.3: Exemplo de edição do ALGORITMO 4.3 .....	85
FIGURA 4.4: Exemplo de modelo de valoração dos mutantes .....	91
FIGURA 4.5: Visores de fatos de análise .....	96
FIGURA 4.6: Classes básicas da árvore de sintaxe .....	97
FIGURA 4.7: Classes do ator <i>se...fim</i> e dos mutantes .....	98
FIGURA 4.8: Árvore de sintaxe simplificada para o ALGORITMO 4.3.....	100
FIGURA 4.9: Exemplos de esquemas de nodos .....	102
FIGURA 4.10: Processo de passagem de fichas.....	104
FIGURA 4.11: Classes do grafo de segmentos.....	106
FIGURA 4.12: Diagrama de segmentos do ALGORITMO 4.3 .....	108
FIGURA 4.13: Visor da planilha de tempos .....	109
FIGURA 4.14: Editor de tarefas da planilha de tempos .....	110
FIGURA 4.15: Visor de parâmetros para análise e simulação de escalonamento.....	111
FIGURA 4.16: Visor de gráficos de tempos .....	112
FIGURA 4.17: Visor de detalhes de serviços .....	114
FIGURA 4.18: Esquema de classes da planilha de tempos .....	115
FIGURA 4.19: Esquema de classes do gráfico de tempos.....	117
FIGURA 4.20: Exemplos de cursores.....	118
FIGURA 4.21: Esquema de classes do diário de simulação.....	120

FIGURA 5.1: Arquivo fonte do caso 1 .....	124
FIGURA 5.2: Planilha de tempos do caso 1 .....	125
FIGURA 5.3: Gráfico de tempos do caso 1 .....	125
FIGURA 5.4: Grafo de segmentos textual do caso 1 .....	126
FIGURA 5.5: Gráficos de tempos do caso 2 .....	128
FIGURA 5.6: Gráficos de tempos do caso 3 .....	129
FIGURA 5.7: Planilha de tempos do caso 4 .....	130
FIGURA 5.8: Gráficos de tempos do caso 4 .....	130
FIGURA 5.9: Planilha de tempos do caso 5 .....	131
FIGURA 5.10: Gráfico de tempos do caso 5 .....	132
FIGURA 5.11: Gráfico de tempos do caso 6 .....	133





## LISTA DE QUADROS

QUADRO 2.1: Classificação dos ambientes n-processadores .....	30
QUADRO 3.1: Classificação das políticas de escalonamento .....	36
QUADRO 3.2: Características das ferramentas pesquisadas .....	76



## LISTA DE TABELAS

TABELA 3.1: Dados para as tarefas da FIGURA 3.6 .....	50
TABELA 3.2: Dados para o exemplo da FIGURA 3.7.....	51
TABELA 3.3: Tempo de bloqueio sob o Protocolo de Herança de Prioridade .....	62
TABELA 3.4: Tempos de bloqueio sob o Protocolo de Teto de Prioridade.....	65
TABELA 3.5: Dados para as tarefas da FIGURA 3.16 .....	68
TABELA 3.6: Dados para as tarefas da FIGURA 3.17 .....	68
TABELA 3.7: Dados para as tarefas da FIGURA 3.18 .....	70
TABELA 3.8: Dados das tarefas para a FIGURA 3.19 .....	72
TABELA 3.9: Acompanhamento dos cálculos de tempo de resposta .....	73



## TABELA DE REDUÇÕES

<b>Redução</b>	<b>Definição da redução no idioma original (Veja Termos)</b>
API	<i>Application Program Interface</i>
AS	Árvore de Sintaxe
BCET	<i>Best Case Execution Time</i>
BCxT	<i>Best Case x Time, where x in {Execution, Blocking, Response, ...}</i>
CASE	<i>Computer Aided Software Engineering</i>
CE	<i>Cyclic Executive</i>
CPU	<i>Central Processing Unit</i>
DMA	<i>Deadline Monotonic Analysis</i>
DMS	<i>Deadline Monotonic Scheduling</i>
DO-178B	Padrão de desenvolvimento de aplicativos para a aviãoica
DPCP	<i>Dynamic Priority Ceiling Protocol</i>
E/S	Entrada e Saída (I/O)
EDF	<i>Earliest Deadline First</i>
ELLF	<i>Enhanced Least Laxity First</i>
FCFS	<i>First Come First Served</i>
FIFO	<i>First In First Out</i>
GNU	<i>GNU's Not Unix</i>
GS	Grafo de segmentos
HL	<i>Highest Locker Protocol</i>
HPF	<i>Highest Priority First</i>
HTML	<i>HyperText Markup Language</i>
ICPP	<i>Immediate Ceiling Priority Protocol</i>
I/O	<i>Input Output</i>
IOP	<i>Input Output Processor</i>
JRE	<i>Java Runtime Environment</i>

<b>Redução</b>	<b>Definição da redução no idioma original (Veja Termos)</b>
LLF	<i>Least Laxity First</i>
MDC	Máximo Divisor Comum
MMC	Mínimo Múltiplo Comum
NP-Hard	<i>Non-Polynomial Hard</i>
OCP	<i>Original Ceiling Priority Protocol</i>
PCP	<i>Priority Ceiling Protocol</i>
PCPA	Primeiro a Chegar Primeiro Atendido
PIP	<i>Priority Inheritance Protocol</i>
POSIX	<i>Portable Operating System Interface</i>
PSJF	<i>Preemptive Shortest Job First</i>
QoS	<i>Quality of Service</i>
RMA	<i>Rate Monotonic Analysis</i>
RMS	<i>Rate Monotonic Scheduling</i>
RR	<i>Round Robin</i>
RTCA	<i>Radio Technical Commission for Aeronautics</i>
RTOS	<i>Real-Time Operating System</i>
RTS	<i>Real-Time System</i>
SJF	<i>Shortest Job First</i>
SJFA	<i>Shortest Job First Approximation</i>
SO	Sistema Operacional
SRP	<i>Stack Resource Policy</i>
SRTF	<i>Shortest Remaining Time First</i>
TCET	<i>Typical Case Execution Time</i>
TCxT	<i>Typical Case x Time , where x in {Execution, Blocking, Response, ...}</i>
WCET	<i>Worst Case Execution Time</i>
WCxT	<i>Worst Case x Time, where x in {Execution, Blocking, Response, ...}</i>
UML	<i>Unified Modeling Language</i>

<b>Redução</b>	<b>Definição da redução no idioma original (Veja Termos)</b>
----------------	--

---

XML	<i>Extended Markup Language</i>
-----	---------------------------------





## TABELA DE SÍMBOLOS

Símbolo	Definição do símbolo
$a_i$	Tempo de atraso na liberação de um serviço da tarefa $i$ .
$B$	Tempo de bloqueio de um modo geral.
$B_i$	Tempo de bloqueio sofrido por um serviço da tarefa $i$ .
$csc(i,S)$	Conjunto de índices das seções críticas do recurso S em um serviço da tarefa $i$ .
$C$	Tempo de computação de um modo geral.
$C_i$	Tempo de computação realizada por um serviço da tarefa $i$ .
$C_{i,S}, (C_{i,S})_m$	Tempo de computação realizado por um serviço da tarefa $i$ enquanto detém a trava do recurso S. Um serviço pode conter diversos destes intervalos de tempo instanciados por $m=1,2,\dots$
$D$	Prazo de um modo geral relativo ao instante de sua ativação.
$D_i$	Prazo para um serviço da tarefa $i$ relativo ao instante de sua ativação.
$delay()$	Comando para colocar determinada tarefa em espera por dado intervalo de tempo.
$hep(i)$	Conjunto de tarefas com prioridade maior ou igual à prioridade da tarefa $i$ .
$hp(i)$	Conjunto de tarefas com prioridade maior que a prioridade da tarefa $i$ .
$I$	Tempo de interferência de um modo geral.
$I_i$	Tempo de interferência sofrida por um serviço da tarefa $i$ .
$J_i$	Tempo de flutuação de liberação sofrida por um serviço da tarefa $i$ .
$Jt_i$	Tempo de flutuação de término de um serviço da tarefa $i$ .

Símbolo	Definição do símbolo
$locks(k,i)$	Conjunto de recursos utilizados pela tarefa $k$ e também pelas tarefas com prioridade básica maior ou igual à prioridade básica da tarefa $i$ .
$lp(i)$	Conjunto de tarefas com prioridade menor que a prioridade da tarefa $i$ .
$L_i$	Tempo de relaxamento; diferença entre o prazo e o término estimado de um serviço.
$\max_{i \in \{\dots\}} x_i, \max_{i=1}^n x_i$	Maior valor do conjunto de valores selecionados pela variável $i$ .
$\min\{x_1, x_2, \dots\}, \min_{0 < t < T} f(t)$	Menor valor do conjunto de valores selecionados pela variável $i$ .
$O_i$	Tempo de deslocamento da tarefa $i$ .
$P_i$	Prioridade da tarefa $i$ em relação a um conjunto de tarefas que compõem um aplicativo.
$P_i(t), P(t)$	Valor de probabilidade para o instante de tempo $t$ em relação à tarefa $i$ .
$r_i, r_{i,q}$	Tempo de resposta de um serviço da tarefa $i$ relativo ao instante de sua liberação.
$R_i, R_{i,q}$	Tempo de resposta de um serviço da tarefa $i$ relativo ao instante de sua ativação.
$Rmax_i, Rmin_i$	Tempo de resposta máximo e mínimo de um serviço da tarefa $i$ , respectivamente.
$R_i^{(v)}, r_{i,q}^{(v+1)}$	Notação que representa a versão do valor em fórmulas com recorrência.
$R_i^{(J)}$	Tempo de resposta líquido do serviço ( $J$ ), que acrescido de um deslocamento resulta no tempo de resposta total.
$sleep()$	O mesmo que $delay()$ .
$S_k, S$	Recurso compartilhado $k$ . Recurso compartilhado de um modo geral.

Símbolo	Definição do símbolo
$T_i, T, T_{min}$	Período de tempo entre ativações consecutivas de serviços da tarefa $i$ . Período de um modo geral. Período mínimo.
$\hat{T}$	Transação formada por diversas tarefas encadeadas. Constitui um grafo dirigido de ordem parcial.
$U(n)$	Taxa de utilização do processador em função do número de usuários.
$V_i$	Tempo de suspensão voluntária de um serviço da tarefa $i$ .
$W_i(t)$	Carga do processador imposto pelas tarefas com prioridade maior ou igual à prioridade da tarefa $i$ até o instante de tempo $t$ .
<code>yield()</code>	Comando para ceder o controle do processador para outras tarefas; capitular.
$\Gamma$	Conjunto ordenado total de tarefas segundo suas prioridades.
$\tau_i$	Designação das tarefas, por exemplo, $\tau_a$ e $\tau_1$ .
$\varphi$	Deslocamento de fase. Intervalo de tempo entre as ativações de dois serviços.
$\lceil x \rceil$	Função teto, que retorna o valor arredondado para o menor inteiro maior ou igual a $x$ , por exemplo, $\lceil 4/3 \rceil = 2$ .
$\lfloor x \rfloor$	Função piso, que retorna o valor arredondado para o maior inteiro menor ou igual a $x$ , por exemplo, $\lfloor 5/3 \rfloor = 1$ .
$[a, b), (a, b]$	Intervalo de valores entre $a$ e $b$ aberto à esquerda e fechado à direita e vice-versa, respectivamente.
$i..j$	Denota o conjunto de todos os inteiros $k$ em que $i \leq k \leq j$ .
$\bar{b}$	Função negação, que retorna a negação da expressão lógica $b$ .



## TABELA DE TERMOS

Termo	Tradução e fixação
<i>Activation</i>	Ativação, lançamento
<i>Application Program Interface</i>	Interface com Programa de Aplicativo
<i>Best Case Execution Time</i>	Melhor Caso de Tempo de Execução
<i>Best Case x Time</i>	Melhor Caso de Tempo de $x$ , para $x$ em {Execução, Bloqueio, Resposta, ...}
<i>Buffer</i>	Área para guarda temporária de dados
<i>Cache</i>	Memória auxiliar de acesso recente
<i>Central Processing Unit</i>	Unidade Central de Processamento
<i>Chip set</i>	Conjunto de circuitos integrados
<i>Computer Aided Software Engineering</i>	Engenharia de Aplicativos Assistida por Computador
<i>Cyclic Executive</i>	Executivo Cíclico
<i>Deadline</i>	Prazo
<i>Deadline Monotonic Analysis</i>	Análise por Prazos Monotônicos
<i>Deadline Monotonic Scheduling</i>	Escalonamento por Prazos Monotônicos
<i>Deadlock</i>	Impasse de bloqueio
<i>Disk Operating System</i>	Sistema Operacional em Disco
<i>Dynamic Priority Ceiling Protocol</i>	Protocolo Dinâmico de Teto de Prioridade
<i>Earliest Deadline First</i>	Prazo Mais Cedo Primeiro
<i>Enhanced Least Laxity First</i>	Menor Relaxamento Primeiro Melhorado
<i>Extended Markup Language</i>	Linguagem de Marcação Estendida
<i>Feasibility</i>	Exeqüibilidade, viabilidade
<i>Firm Real-Time</i>	Tempo Real Rígido
<i>First Come First Served</i>	Primeiro a Chegar Primeiro Atendido
<i>First In First Out</i>	Primeiro a Entrar Primeiro a Sair

<b>Termo</b>	<b>Tradução e fixação</b>
<i>Framework</i>	Arcabouço, coleção de idéias
<i>GNU's Not Unix</i>	GNU Não é Unix
<i>Hard Real-Time</i>	Tempo Real Severo
<i>Highest Locker Protocol</i>	Protocolo de Trava Máxima
<i>Highest Priority First</i>	Maior Prioridade Primeiro
<i>HyperText Markup Language</i>	Linguagem de Marcação de Hiper-Texto
<i>Immediate Ceiling Priority Protocol</i>	Protocolo de Prioridade de Teto Imediato
<i>Input Output Processor</i>	Processador de Entrada e Saída
<i>Invocation</i>	Invocação, serviço
<i>Java Runtime Environment</i>	Ambiente Executivo Java
<i>Jitter</i>	Flutuação, ginga (fr. gingue)
<i>Job</i>	Serviço
<i>Laxity</i>	Relaxamento, laxidão
<i>Least Laxity First</i>	Menor Tempo de Relaxamento Primeiro
<i>Lock</i>	Trava, travar
<i>Non-Polynomial Hard</i>	Não-Polinomial intensivo, NP-intensivo
<i>Offset</i>	Deslocamento
<i>Original Ceiling Priority Protocol</i>	Protocolo de Prioridade de Teto Original
<i>Pipeline</i>	Estrutura seqüencial em que elementos adjacentes se comunicam para frente
<i>Portable Operating System Interface</i>	Interface Portável de Sistemas Operacionais
<i>Preemption, Preemptive, Preempt</i>	Preempção, Preemptivo, Preemptir
<i>Preemptive Shortest Job First</i>	Serviço Mais Curto Primeiro Preemptivo
<i>Prefetch Queues</i>	Fila de Busca Antecipada
<i>Priority Ceiling Protocol</i>	Protocolo de Teto de Prioridade
<i>Priority Inheritance Protocol</i>	Protocolo de Herança de Prioridade
<i>Quality of Service</i>	Qualidade de Serviço

<b>Termo</b>	<b>Tradução e fixação</b>
<i>Radio Technical Commission for Aeronautics</i>	Comissão Rádio Técnica para Aeronáutica
<i>Rate Monotonic Analysis</i>	Análise por Taxas Monotônicas
<i>Rate Monotonic Sheduling</i>	Escalonamento por Taxas Monotônicas
<i>Real-Time Operating System</i>	Sistema Operacional em Tempo Real
<i>Real-Time System</i>	Sistema em Tempo Real
<i>Release</i>	Liberação, liberar
<i>Rendevouz</i>	Encontro
<i>Shortest Job First</i>	Serviço Mais Curto Primeiro
<i>Shortest Job First Approximation</i>	Abordagem por Serviço Mais Curto Primeiro
<i>Shortest Remaining Time First</i>	Menor Tempo Restante Primeiro
<i>Soft Real-Time</i>	Tempo Real Fraco
<i>Stack Resource Policy</i>	Política de Pilha de Recursos
<i>Task</i>	Tarefa
<i>Thread (of control)</i>	Linha (de controle)
<i>Timeliness</i>	Pontualidade
<i>Transaction</i>	Transação
<i>Typical Case Execution Time</i>	Caso Típico de Tempo de Execução
<i>Typical Case <math>x</math> Time</i>	Caso Típico de Tempo de $x$ , para $x$ em {Execução, Bloqueio, Resposta, ...}
<i>Virtual Storage Environment</i>	Ambiente de Memória Virtual
<i>Workbench</i>	Ferramental, bancada, oficina
<i>Worst Case Execution Time</i>	Pior Caso de Tempo de Execução
<i>Worst Case <math>x</math> Time</i>	Pior Caso de Tempo de $x$ , para $x$ em {Execução, Bloqueio, Resposta, ...}
<i>Unified Modeling Language</i>	Linguagem de Modelagem Unificada





## RESUMO

O ensino e o aprendizado necessitam de ferramentas de apoio em que os modelos teóricos são transportados para ambientes de simulação e animação. Isto torna-se especialmente relevante, quando a complexidade dos modelos supera a capacidade de tratamento convencional (lápiz e papel). O programa de computador associado à dissertação pretende justamente disponibilizar um ambiente de experimentação em que o estudante experimenta contextos de tarefas e recursos para verificar a escalonabilidade. O ambiente é instrumentado por uma linguagem para descrição comportamental de tarefas, que imita linguagens de programação de uso comum, por um analisador de tempos de execução, espera, bloqueio, etc., por um analisador de escalonabilidade à luz da teoria de escalonamento clássica e por um simulador e animador de eventos.

O referido programa de computador resgata características de outras ferramentas, acadêmicas e comerciais, especialmente com relação à interface gráfica pessoal-ferramenta. Por outro lado afasta-se de ambientes reais, tais como sistemas operacionais e problemas práticos, de modo que o esforço possa ser dirigido massivamente ao problema em exercício.

Palavras-chave: Sistemas em tempo real; escalonamento de tarefas; suporte ao ensino e aprendizado; simulação; animação.



## **ABSTRACT**

*Teaching and learning activities benefit from supporting tools that simulate and animate theoretical models. This is specially desirable when the complexity of the models hinders using the conventional approach (pen and paper). The computer program developed along this research provides an experimentation framework that students may use to animate complex scenarios with tasks and resources in order to verify their schedulability. A programming language was developed to allow the description of the behavior of tasks; an engine executes the model; which is then analyzed by a waiting and blocking time analyser and a schedulability analyser.*

*The program relies on other tools, academic and commercial, implementing likely graphical user interfaces. Although, it does not require real environments, such as operating systems, hardware kits and real problems. The student may direct his efforts just to the problems in hand.*

*Keywords: Real Time Systems; task scheduling; teaching and learning support; simulation; animation.*



## **1 INTRODUÇÃO**

Sistemas em tempo real tornaram-se corriqueiros nos dias de hoje, com a larga difusão de dispositivos embarcados e seus aplicativos para monitoramento e controle de processos, permeando praticamente todas as áreas de atividade humana.

Por outro lado, nem todos os aplicativos são desenvolvidos com modelagem matemática e práticas de engenharia. Dependendo da complexidade e dos riscos envolvidos, tem-se a noção de que possam ser gerados com práticas convencionais (e artesanais), tais como programação estruturada e validação por testes.

Evidentemente, isto não precisa e nem sempre pode ser assim. Dispõe-se de diversas metodologias para a abordagem de sistemas em tempo real, tais como UML 2.0 e modelos matemáticos para análise de restrições de tempo e exequibilidade de aplicativos. Além disso, o mercado dispõe de diversas ferramentas de análise e simulação de sistemas em tempo real no domínio do tempo.

Pelo exposto, uma das preocupações nos meios de ensino é a formação de pessoal, habilitando os arquitetos, projetistas e programadores nos conhecimentos e técnicas relacionados à construção de sistemas em tempo real.

### **1.1 OBJETO DE ESTUDO**

O estudo realizado neste trabalho abrange as estrutura de sistemas em tempo real, os fenômenos (comunicação, sincronização, travas, etc.) que ocorrem na execução paralela e/ou concorrente de diversas tarefas disputando recursos partilhados e os modelos matemáticos para descrição destes fenômenos.

Além disso, o estudo apresenta ferramentas comerciais e acadêmicas existentes no mercado, para fundamentar a proposta de construção de uma ferramenta para apoio ao ensino e aprendizado, com características de simulação, animação e modelagem analítica.

## 1.2 OBJETIVO E ESCOPO

O objetivo deste trabalho é o desenvolvimento de uma ferramenta essencialmente educativa, para servir de ambiente de experimentação na área de escalonamento de tarefas para sistemas em tempo real. O propósito fundamental é o de auxiliar o aluno na compreensão dos diversos fenômenos tratados pela teoria de escalonamento, através de linguagens gráficas e ambientes interativos.

Como efeito secundário, pretende-se expor e fixar uma terminologia fortemente nacional, uma vez que o conhecimento tem suporte mais abrangente, quando baseado em termos e expressões da língua alvo.

## 1.3 JUSTIFICATIVAS

A análise das principais ferramentas existentes no escopo do escalonamento de sistemas em tempo real permite estabelecer as seguintes justificativas para este trabalho:

- a importância de ferramentas para experimentar exercícios com complexidade além da capacidade de tratamento convencional (lápiz e papel);
- o isolamento do ambiente com relação a aspectos irrelevantes ao contexto, tais como criação de problemas reais expressos em linguagens de programação, conhecimento das APIs de sistemas operacionais e conhecimento de processadores e dispositivos de entrada e saída;
- o agrupamento de diversos processos num único ambiente, tais como descrição comportamental de tarefas, análise de tempos e simulação do escalonamento das tarefas;
- flexibilidade para simular e animar tarefas que não aderem aos modelos da teoria de escalonamento clássica.

Finalmente, a ferramenta que resulta deste trabalho possibilita sua extensão para outras áreas de experimentação e integração com outros modelos e métodos de análise da escalonabilidade como, por exemplo, redes de Petri.

## 1.4 RESULTADOS

Deste trabalho resultou a especificação de um ambiente de apoio ao ensino e aprendizado do escalonamento e a implementação de um protótipo da ferramenta *AnimaTi*, que implementa partes da especificação.

O arcabouço formado pela especificação da ferramenta cobre diversos aspectos relacionados com o escalonamento, tais como a escrita de algoritmos abstratos, a simulação destes algoritmos e a extração de tempos para análise e simulação do escalonamento.

## 1.5 CONTEÚDO

No Capítulo 2 apresenta-se as principais características de sistemas em tempo real através de uma linguagem consistente, reunindo diversas abordagens presentes na literatura. Aborda-se a modularização dos aplicativos em tarefas, o ciclo de vida típico das tarefas e sua classificação quanto à pontualidade e frequência de execução. Aborda-se a dinâmica das tarefas através de serviços e seus ciclos de vida. Apresenta-se um formato de diagramas de Gantt utilizado ao longo de todo o trabalho para explicar diversos aspectos do comportamento temporal e da interação de tarefas componentes de um mesmo aplicativo. Descreve-se os diversos intervalos e instantes de tempo que caracterizam fenômenos temporais, tais como período, prazo e espera por recursos. Aborda-se, ainda, os ambientes operacionais e suas características com relação ao modo de controle e número de processadores. No Capítulo 3 aborda-se a teoria de escalonamento propriamente dita. Apresenta-se as características de grande número de estratégias de escalonamento, tais como CE, EDF e LLF. Apresenta-se a análise de escalonabilidade do ponto de vista da utilização do processador, da carga do processador e do tempo de resposta com as devidas formulações matemáticas. Trata-se dos diversos protocolos de escalonamento para solução de impasses de bloqueio de recursos e eliminação da inversão de prioridades. Ao final apresenta-se o resultado da pesquisa de ferramentas existentes. No Capítulo 4 apresenta-se a especificação da

ferramenta *AnimaTi*. A ferramenta está dividida em módulos que cobrem os diversos aspectos do escalonamento. Estes módulos são apresentados juntamente com seus componentes, as principais estruturas de objetos e as funcionalidades dos visores. No Capítulo 5 apresenta-se alguns estudos de casos para validar a ferramenta. Os casos constituem cenários fictícios extraídos da literatura e do Capítulo 3 deste trabalho. A validação consiste em obter os mesmos resultados utilizando o protótipo da ferramenta. No Capítulo 6 apresenta-se as conclusões e sugestões para trabalhos futuros.



## **2 SISTEMAS EM TEMPO REAL**

Neste capítulo aborda-se os aspectos relevantes a sistemas em tempo real e prepara-se os conhecimentos básicos necessários para a compreensão da teoria de escalonamento. A abordagem cobre a anatomia de aplicativos em tempo real e as estruturas de ambientes operacionais para suporte aos aplicativos.

### **2.1 INTRODUÇÃO**

Sistemas em tempo real agrupam diversos componentes para um propósito comum. Os componentes podem englobar os mais variados tipos de dispositivos eletrônicos e eletromecânicos, tais como sensores, atuadores, processadores específicos e de propósito geral, barramentos de dados, redes de comunicação e componentes formados por programas computacionais. A coleção de programas está em destaque neste trabalho e será denominada "aplicativo em tempo real". Modernamente os sistemas são orquestrados por seus aplicativos, e a complexidade recai (quase que) totalmente sobre estes. Assim, sua elaboração e manutenção necessita do apoio de fundamentos matemáticos, em particular, da teoria de escalonamento para garantir sua exeqüibilidade e confiabilidade no tempo.

Estes sistemas têm aplicação em diversas áreas, tais como controle de processos, chaveamento de trilhos, telecomunicações, tráfego aéreo, aplicações militares, automobilística, rede de distribuição de energia elétrica, equipamentos hospitalares e robótica. De modo geral, sistemas em tempo real se fazem presentes em praticamente todas as áreas.

#### **2.1.1 Computação em tempo real**

Nesta seção revisa-se as principais características de aplicativos com restrições de tempo críticas (FIDGE, 2002), a saber: a) a correção do aplicativo depende do tempo em que os resultados são produzidos, além de sua correção lógica; b) as medidas dos

tempos (não só de execução) no aplicativo estão relacionadas a eventos do ambiente externo real (BUTTAZZO, 1997); c) não é somente computação rápida, pois aumentar a velocidade de processamento não resolve todos os problemas de tempo; d) não é somente uma consideração para boa vazão ou desempenho de caso médio, pois, isto não provê garantias de atendimento aos prazos; e) não é somente uma questão de temporização que em geral é necessária para a programação tolerante a faltas (STANKOVIC, 1988); f) são reativos, interagindo repetidamente com seu ambiente; g) em geral são componentes de um aplicativo maior que se comunicam via interfaces com dispositivos eletro-eletrônicos; e h) estão comprometidos com funções importantes, devendo ser confiáveis e livres de problemas.

A lista acima mostra apenas os aspectos mais relevantes, omitindo circunstâncias como a necessidade de resposta e a adversidade do meio em que atuam.

## 2.2 GRAUS DE PONTUALIDADE

As tarefas em tempo real podem ser classificadas quanto à pontualidade do cumprimento de seus prazos com relação ao valor de suas respostas em três categorias de acordo com a FIGURA 2.1 (BUTTAZZO, 1997), a saber: tempo real severo, tempo real rígido e tempo real fraco.

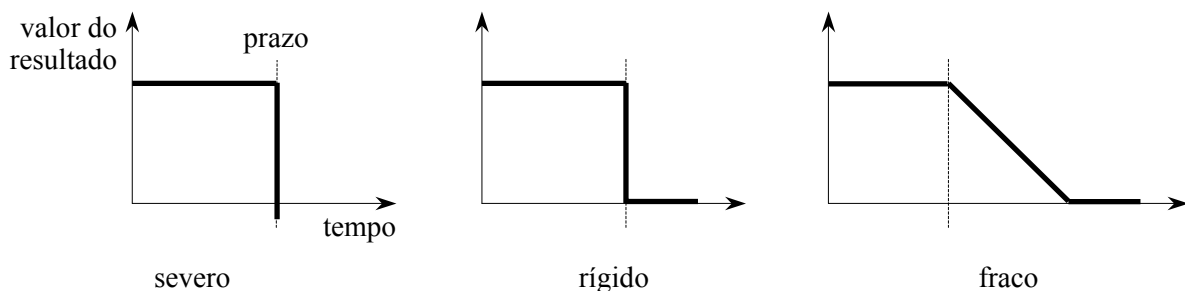


FIGURA 2.1: Graus de pontualidade em sistemas em tempo real

Em geral, os aplicativos para controle de processos operam com restrições de tempo severas, enquadrando-se na categoria de tempo real severo. No entanto, estes não se compõem somente de tarefas em tempo real severo.

A categoria de um aplicativo é estabelecida a partir das categorias das tarefas componentes. Sejam os pesos  $P$  das categorias  $P(\text{severo}) > P(\text{rígido}) > P(\text{fraco})$ , o aplicativo é caracterizado pela categoria de maior peso.

### **2.2.1 Tarefas em tempo real severo**

Tarefas em tempo real severo impõem restrições de tempo tal que o não cumprimento dos prazos torna-se catastrófico, causando danos em sistemas e à própria vida humana. Por exemplo, uma nave realizando um pouso deve acionar os retrofoguetes num instante preciso de modo que não se choque com a superfície de pouso ou que não fique flutuando excessivamente e eventualmente perca o equilíbrio.

Tarefas em tempo real severo podem, ainda, ter uma subcategoria de tempo real real que caracteriza as tarefas com tempos de resposta extremamente curtos, por exemplo, os tempos de resposta presentes em guias de mísseis (BURNS; WELLINGS, 2001).

### **2.2.2 Tarefas em tempo real rígido**

Tarefas em tempo real rígido impõem restrições de tempo menos críticas e o não cumprimento dos prazos implica em esbanjar recursos. Os resultados produzidos perdem o valor imediatamente. Por exemplo, a consulta a uma base de dados torna-se inútil após o usuário perder a paciência e abandonar a questão (FIDGE, 2002).

### **2.2.3 Tarefas em tempo real fraco**

Tarefas em tempo real fraco ainda possuem prazos, mas o não cumprimento destes reduz gradualmente o valor dos resultados produzidos. Por exemplo, o cálculo da previsão do tempo para o dia seguinte deve estar disponível ao anoitecer. Se isto não acontecer, a previsão do tempo perde cada vez mais o valor até a hora do café da manhã, quando perde o valor totalmente (FIDGE, 2002).

## 2.3 MODELO DE APLICATIVOS EM TEMPO REAL

Do ponto de vista da teoria de escalonamento, aplicativos em tempo real consistem nos seguintes componentes (AUDSLEY *et alii*, 1993):

- um conjunto de tarefas computacionais a serem executadas; em geral são sub-rotinas com suas próprias linhas de controle; construtos, tais como servidores de interrupções e escalonadores, que consomem algum recurso computacional podem ser modelados como tarefas;
- um escalonador executivo que implementa as políticas de controle de alocação do processador às tarefas;
- um conjunto de recursos compartilhados utilizados pelas tarefas; em geral, estes recursos são constituídos por variáveis com controle de exclusividade de acesso e por interfaces com dispositivos eletrônicos e eletromecânicos compartilhados, tais como barramentos de dados, sensores, atuadores e dispositivos de entrada/saída.

Toda a sincronização e comunicação entre tarefas se dá através dos recursos compartilhados. Quando se trata de ambientes multiprocessados distribuídos, o modelo básico apresentado necessita de outros mecanismos para viabilizar a sincronização e comunicação.

Nas próximas seções define-se os conceitos de tarefa e serviço e mostra-se o diagrama de estados do ciclo de vida tradicional de tarefas (BURNS; WELLINGS, 2001). O ciclo de vida de tarefas está decomposto em dois diagramas de estados: o primeiro mostra o ciclo de vida de tarefas e o segundo mostra o ciclo de vida de serviços realizados pelas tarefas.

### 2.3.1 Ciclo de vida de tarefas

As tarefas seguem o ciclo de vida representado pelo diagrama de estados da FIGURA 2.2. Os estados à esquerda do diagrama de estados tratam da iniciação da

tarefa e de suas eventuais tarefas filho. Os estados à direita tratam da finalização da tarefa e de suas eventuais tarefas filho.

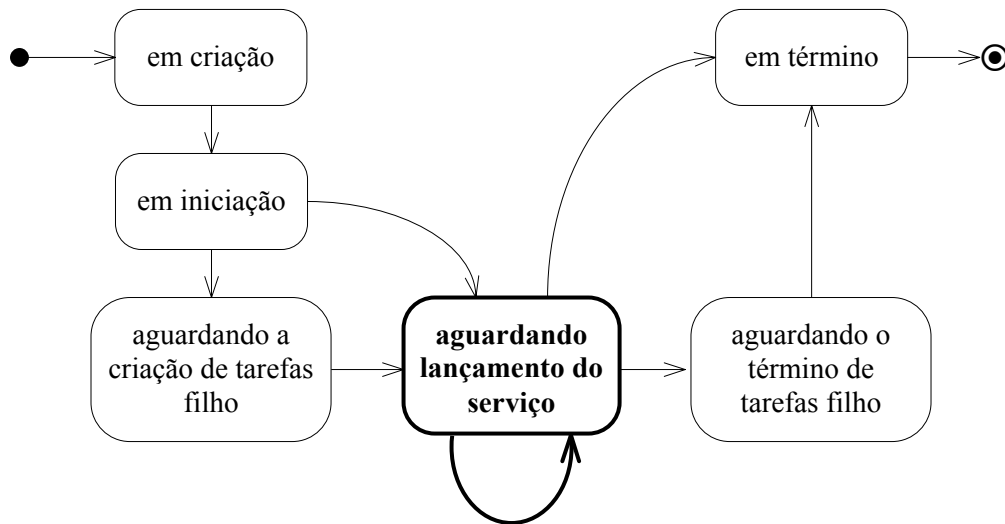


FIGURA 2.2: Ciclo de vida de tarefas

Cada tarefa instancia uma seqüência de serviços idênticos, eventualmente concorrentes, através da auto-transição do estado ressaltado na FIGURA 2.2.

O segmento de algoritmo, que implementa a atividade do serviço, pode estar embutido diretamente no código de controle da tarefa ou pode figurar como uma linha de controle independente. Este último caso aplica-se às tarefas cujo período  $T$  (Seção 2.4.1) é menor que o prazo  $D$  (Seção 2.4.1), quando então dois ou mais serviços da mesma tarefa podem coexistir. Embora as atividades de todos os serviços possam ser implementadas por linhas de controle independentes, deve-se avaliar a sobrecarga de processamento introduzida por este modelo de implementação.

O ALGORITMO 2.1 e o ALGORITMO 2.2 mostram dois segmentos de código fonte típicos para a implementação de tarefas: o primeiro segmento descreve tarefas ativadas a intervalos de tempo fixos, e o segundo segmento descreve tarefas ativadas a intervalos de tempo aleatórios.

---

**ALGORITMO 2.1: Algoritmo para tarefas com intervalo de ativação fixo**

---

```
task Tarefa1 is
end Tarefa1;

task body Tarefa1 is
    horaDaAtivação : Time;
begin
    ...
    -- anota o instante para ativação imediata
    horaDaAtivação := Clock;
    loop
        -- suspende a tarefa até a próxima ativação
        delay until horaDaAtivação;
        -- código do serviço
        ...
        -- calcula o instante da próxima ativação
        horaDaAtivação := horaDaAtivação + T;
    end loop;
end Tarefa1;
```

---

**ALGORITMO 2.2: Algoritmo para tarefas com ativação aleatória**

---

```
task Tarefa2 is
    entry pontoDeEncontro;
end Tarefa2;

task body Tarefa2 is
begin
    loop
        -- espera alguém ativar o serviço através do encontro
        accept pontoDeEncontro do
            -- seção crítica do encontro para
            -- eventual comunicação de dados
            ...
        end pontoDeEncontro;
        -- código do serviço
        ...
    end loop;
end Tarefa2;
```

---

### 2.3.2 Ciclo de vida de serviços

O serviço é uma instância em execução da atividade da tarefa, ou simplesmente da tarefa. A política de lançamento dos serviços e o número de serviços idênticos ativos simultaneamente depende do tipo de tarefa e da relação entre período e prazo da tarefa. O serviço segue o ciclo de vida representado pelo diagrama de estados da FIGURA 2.3.

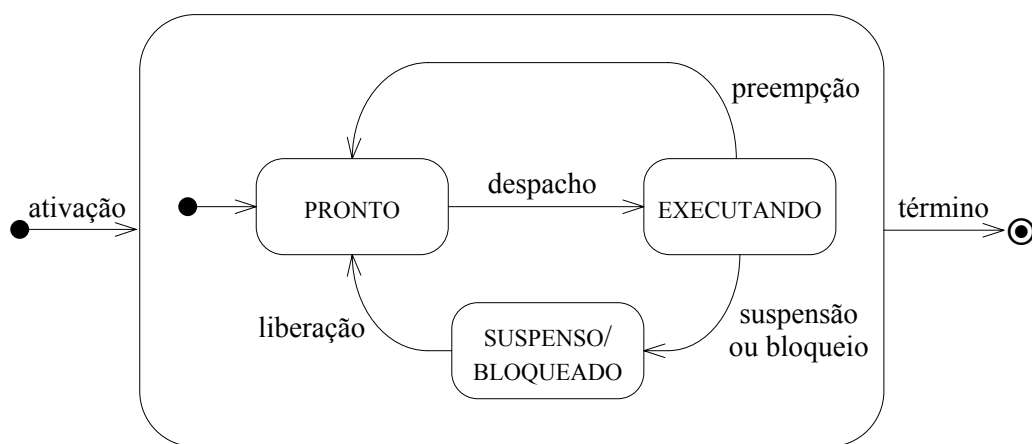


FIGURA 2.3: Ciclo de vida básico de serviços

Os três estados, PRONTO, EXECUTANDO e SUSPENSO/BLOQUEADO representam as situações relevantes para a teoria de escalonamento em que os serviços podem se encontrar.

Para a representação destas situações no tempo, utilizam-se ao longo deste trabalho diagramas de Gantt coloridos de forma a diferenciar os motivos pelos quais os serviços se encontram nos diferentes estados.

#### 2.3.2.1 Estado do serviço PRONTO

O serviço no estado PRONTO aguarda em um esquema de filas sua vez para receber o controle do processador. Cabe ao escalonador executivo determinar o próximo serviço a ser despachado e passar ao estado EXECUTANDO.

Os esquemas de filas devem ser adequados às políticas de escalonamento que levam em conta diversos elementos, tais como prioridade, critérios de desempate para a mesma prioridade e redespacho após preempção (perda compulsória do controle do processador). As políticas de escalonamento são objeto do próximo capítulo.

O serviço entra no estado PRONTO imediatamente após a sua ativação, por preempção ou após a liberação do estado SUSPENSO/BLOQUEADO. O serviço permanece neste estado pelos seguintes motivos:

- a) pelo tempo decorrido entre a ativação e sua liberação;
- b) por estar suspenso preemptido por tarefas mais prioritárias; ou
- c) por estar suspenso bloqueado por tarefas menos prioritárias.

### **2.3.2.2 Estado do serviço EXECUTANDO**

O serviço no estado EXECUTANDO permanece no controle do processador até que seja interrompido por um dos seguintes motivos:

- d) pela cessão voluntária do controle do processador, entrando no estado PRONTO, por exemplo, pela execução da chamada `yield()`;
- e) pela suspensão voluntária do controle do processador, entrando no estado SUSPENSO(/BLOQUEADO) (BURNS; WELLINGS, 2001), por exemplo, pela execução da chamada `delay()`. A teoria de escalonamento pressupõe que serviços não se suspendam voluntariamente (AUDSLEY *et alii*, 1995);
- f) pela perda do controle do processador na tentativa frustrada em obter acesso a um recurso compartilhado, entrando no estado (SUSPENSO/)BLOQUEADO;
- g) pela perda do controle do processador na tentativa frustrada de estabelecer um encontro de sincronização e/ou comunicação, entrando no estado (SUSPENSO/)BLOQUEADO;
- h) pela perda da vez para outra tarefa por preempção, entrando no estado PRONTO; ou
- i) simplesmente quando completa o serviço e termina.



Toda vez em que um serviço é interrompido, o escalonador executivo deve entrar em ação para determinar o próximo serviço a ser despachado.

### **2.3.2.3 Estado do serviço SUSPENSO/BLOQUEADO**

O serviço no estado SUSPENSO/BLOQUEADO aguarda em um esquema de filas algum evento capaz de remover o motivo pelo qual está neste estado e liberá-lo. O serviço permanece neste estado pelos seguintes motivos:

- j) está suspenso, aguardando o transcurso de um tempo de retardo;
- k) está bloqueado, aguardando a liberação de algum recurso travado; ou
- l) está bloqueado, aguardando o encontro (sincronização/comunicação) com outra tarefa.

A implementação do estado SUSPENSO/BLOQUEADO depende da política de escalonamento. Por exemplo, o esquema de filas para o motivo k) pode ser substituído pelo seguinte: o serviço entra imediatamente no estado PRONTO e quando despachado tenta de novo obter acesso ao recurso (TAFT; DUFF, 1997). A análise de implementações não é objeto deste trabalho.

As notações SUSPENSO(/BLOQUEADO) e (SUSPENSO)/BLOQUEADO referem-se ao estado SUSPENSO/BLOQUEADO, enfatizando apenas o motivo de se estar neste estado: por suspensão ou por bloqueio, respectivamente.

### **2.3.3 Diagramas de Gantt para o ciclo de vida de serviços**

A FIGURA 2.4 mostra um modelo de diagrama de Gantt para representar os serviços, seus estados e alguns eventos relevantes ao longo do tempo (BUTTAZZO, 1997). Cada tarefa possui uma linha que mapeia os serviços no tempo. Cada serviço compõe um retângulo com subdivisões no tempo coloridas de acordo com o estado corrente do serviço. A mudança de cor representa a resposta aos eventos do diagrama de estados da FIGURA 2.3.

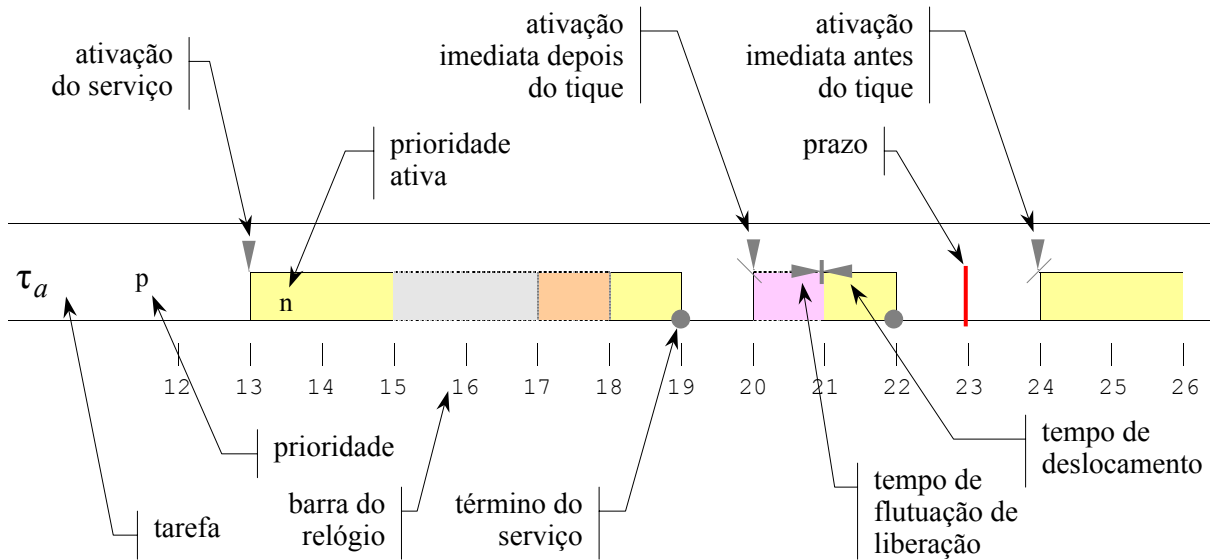










FIGURA 2.4: Diagrama de Gantt dos serviços no tempo

Os símbolos de ativação e ativação imediata antes/depois dos serviços indicam o instante em que um serviço é lançado pelo escalonador. Os símbolos da ativação imediata antes/depois de um instante de tempo diminuem/acrescentam infinitésimos de tempo de modo a desempatar ativações em duas ou mais tarefas diferentes, aparentemente paralelas, de forma determinística (válido somente para os exemplos). Outras anotações, tais como prioridade (básica), prioridade ativa, prazo e flutuação de liberação (BURNS; WELLINGS, 2001) serão explicadas na Seção 2.4.1.

A FIGURA 2.5 relaciona as cores com os estados e os motivos (relacionados em seções anteriores) de permanência nos estados. O estado EXECUTANDO está subdividido em pelo menos duas situações: a) quando está executando independente de outras tarefas; e b) quando está executando em uma seção crítica com algum recurso  $S$  travado. Os exemplos ao longo deste trabalho envolvem no máximo os recursos  $S_1$  e  $S_2$  (suficientes para mostrar exemplos relevantes), de modo que são necessárias três cores para diferenciar as situações do estado EXECUTANDO.

---

	1 Executando independente na sua prioridade natural
	2 Executando em seção crítica com o recurso $S_1$ travado
	3 Executando em seção crítica com o recurso $S_2$ travado
	4 Suspenso preemptido por tarefas mais prioritárias, motivo $b$ )
	5 Bloqueado, aguardando liberação do recurso $S_1$ , motivo $k$ )
	6 Bloqueado, aguardando liberação do recurso $S_2$ , motivo $k$ )
	7 Suspenso por tarefas menos prioritárias, motivo $c$ )
	8 Suspenso por flutuação, deslocamento ou retardo, motivos $a$ ) e $j$ )

---

FIGURA 2.5: Explicação da cores para os diagramas de Gantt

Os serviços podem disputar por e travar mais de um recurso ao mesmo tempo. Para representar este fato, as subdivisões dos retângulos que representam este estado podem ser subdivididas na horizontal e coloridas com as cores 2 e 3, respectivamente.

### 2.3.4 Transações

Determinada funcionalidade de um aplicativo pode ser distribuída entre um grupo de tarefas com restrições de precedência, especialmente em sistemas distribuídos multiprocessados. Estes grupos formam transações que possuem características similares às de tarefas: período, prazo, tempo de resposta e ativação (Seção 2.4.1) (GRIGG; AUDSLEY, 1999).

A FIGURA 2.6 mostra uma transação formada por três tarefas  $\hat{T} = (\tau_1, \tau_2, \tau_3)$ , executando nos processadores IOP e CPU. O serviço  $\tau_1$  executa no processador IOP e no final envia dados ao serviço  $\tau_2$ . O serviço  $\tau_2$  executa no processador CPU e no final envia dados ao serviço  $\tau_3$ . O serviço  $\tau_3$  executa no processador IOP e conclui a atividade da transação. Os dados trafegam por barramentos de dados e/ou por uma rede levando tempos consideráveis. A modelagem deve considerar estes tempos.

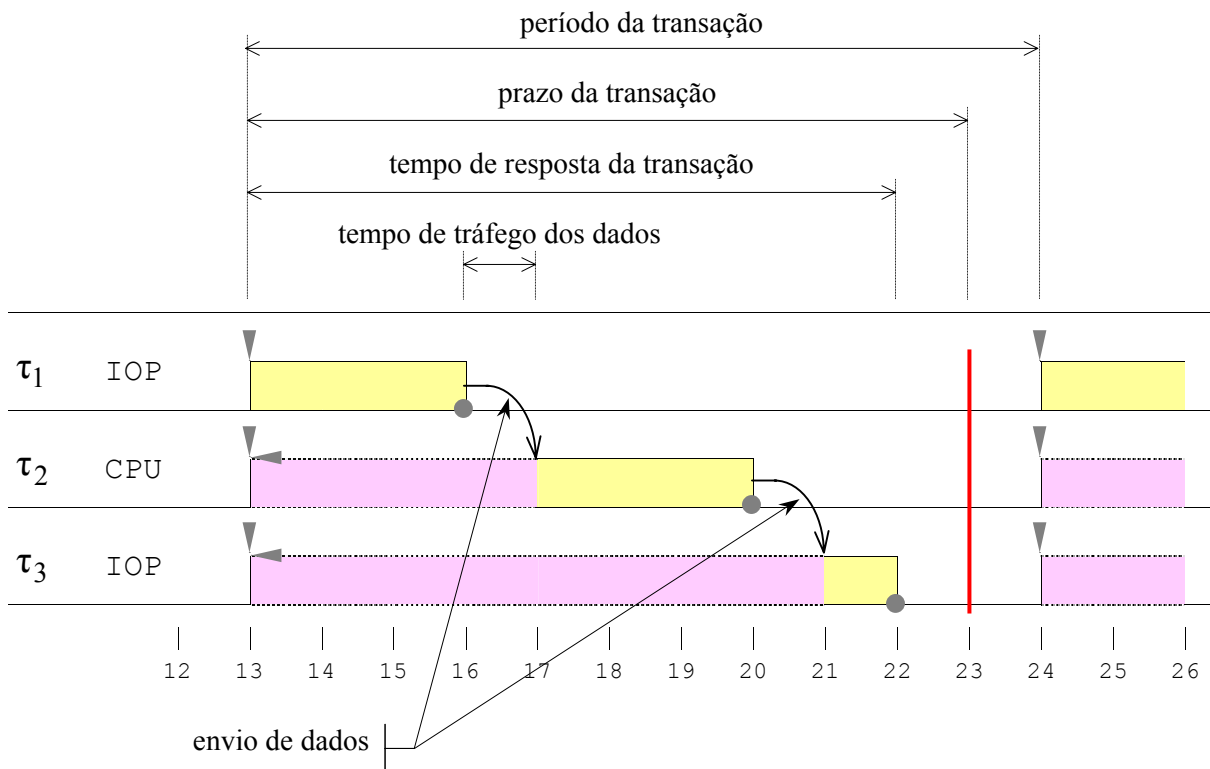


FIGURA 2.6: Exemplo de transação formada por tarefas

A ativação do serviço  $\tau_1$  equivale à ativação da transação. Os pontos de encontro para envio de dados ( $\tau_1, \tau_2$ ) e ( $\tau_2, \tau_3$ ) definem a sincronização entre os serviços e por consequência estabelecem as restrições de precedência. O término do serviço  $\tau_3$  equivale ao término da transação. É importante observar que os serviços sofrem perturbações em cada processador.

O modelo de transações não se restringe à simplicidade do exemplo acima. As restrições de precedência impostas ao conjunto de tarefas que compõem a transação podem ser mais complexas. Alguns serviços podem ser ativados diversas vezes em circunstâncias distintas.

## 2.4 MEDIDAS DE TEMPOS EM SERVIÇOS

Os serviços possuem características temporais que determinam os tempos de permanência nos seus diversos estados. Pode-se estabelecer dois referenciais de tempo para os serviços: a) o tempo global, que conta os tiques de relógio transcorridos

enquanto o aplicativo e/ou ambiente estiver ativo; e b) o tempo relativo, iniciando em 0, que conta os tiques de relógio a partir do momento em que o serviço for lançado por sua ativação. A noção de tempo global em ambientes multiprocessados implica em relógios dos diversos processadores fortemente sincronizados (FALARDEAU, 1994).

### 2.4.1 Definição das medidas de tempos

O ciclo de vida dos serviços pode ser caracterizado por diversas medidas de tempos, levando em consideração eventos e estados. A FIGURA 2.7 mostra as principais medidas de tempos associadas ao ciclo de vida dos serviços.

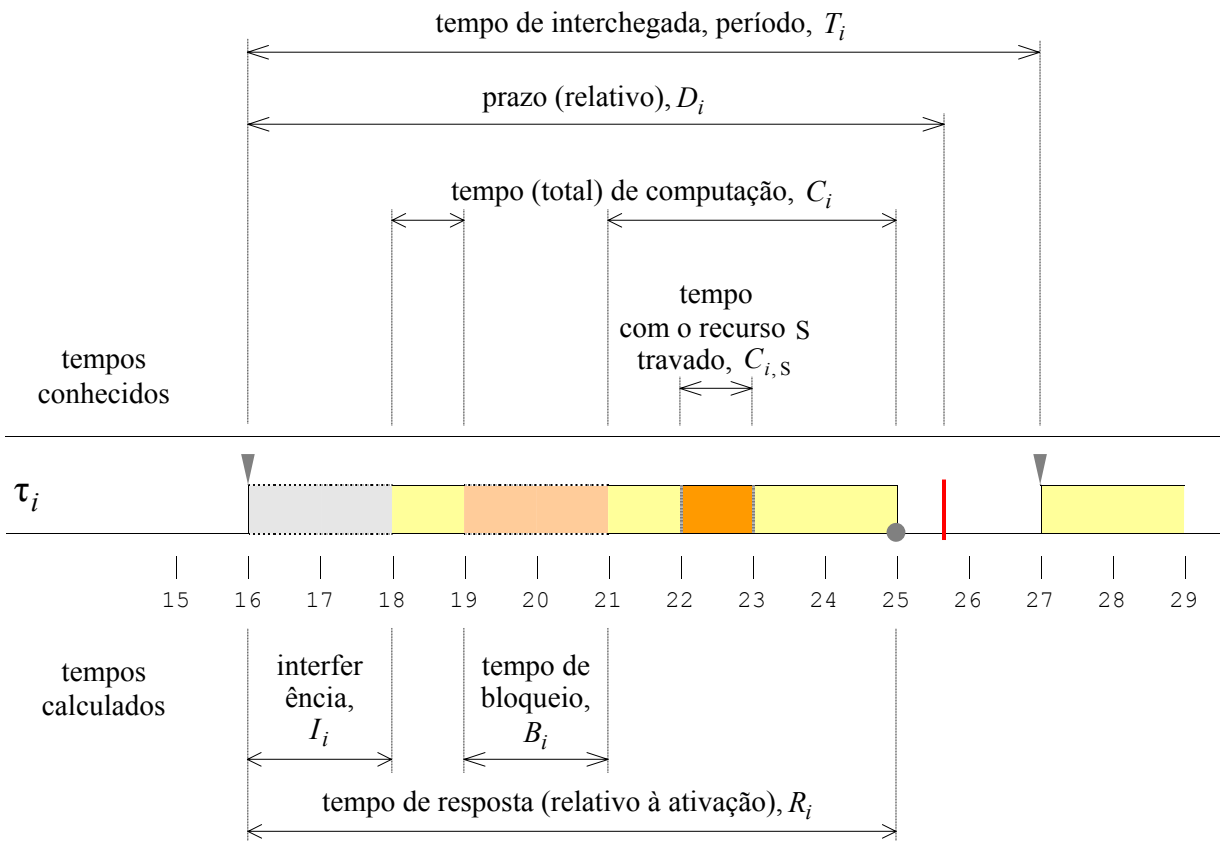


FIGURA 2.7: Principais medidas de tempos em serviços

A barra do relógio mostra o tempo global com tique 0 em algum momento no passado (exatamente quando, não interessa neste contexto). No exemplo

da figura, a tarefa  $\tau_i$  instancia dois serviços sucessivos: a) ativado no instante 16 e terminado no instante 25; e b) ativado no instante 27.

As demais anotações são tempos relativos e podem por sua vez ser divididos em dois grupos: a) tempos conhecidos, estabelecidos pelos requisitos do aplicativo ou medidos pela análise do código dos programas; e b) tempos calculados, obtidos pela aplicação da teoria de escalonamento. A saber:

- a) tempo de interchegada, período  $T_i$ : equivale ao intervalo de tempo transcorrido entre as ativações de dois serviços adjacentes da mesma tarefa; a característica deste tempo depende do tipo da tarefa e pode ser um tempo fixo ou um tempo mínimo (a classificação de tarefas será apresentada na Seção 2.5);
- b) prazo (relativo)  $D_i$ : equivale ao prazo relativo à sua ativação que o serviço tem para disponibilizar os resultados e terminar; existem três modalidades de colocação destes prazos: a)  $D_i = T_i$ , em que o prazo é igual ao período; b)  $D_i < T_i$ , em que o prazo antecede o período; e c)  $D_i > T_i$ , em que o prazo ultrapassa o período; estas modalidades têm relevância na teoria de escalonamento;
- c) tempo (total) de computação  $C_i$ : equivale à soma dos intervalos de tempo em que o serviço está no estado EXECUTANDO; em geral pressupõe-se que os serviços não suspendam o processamento voluntariamente; assim, chamadas do tipo `delay()` contam para o tempo de computação; o tempo  $C_{i,S}$  está incluído no tempo  $C_i$ ;
- d) tempo com o recurso S travado  $C_{i,S}$ : equivale ao intervalo de tempo de computação em que o serviço está em uma seção crítica com o recurso S travado; um serviço pode ter mais de uma seção crítica por recurso; além disso pode ter seções críticas combinadas com, por exemplo, o tempo de  $S_1$  contido no tempo de  $S_2$  ou com seus tempos parcialmente sobrepostos; a notação  $(C_{i,S})_m$  instancia a  $m$ -ésima ocorrência da seção crítica no conjunto das seções críticas do recurso S, e  $csc(i,S)$  retorna o conjunto de índices  $m$  destas seções críticas; o tempo  $C_{i,S}$  está incluído no tempo  $C_i$ ;

- e) interferência  $I_i$ : equivale à soma dos intervalos de tempo em que o serviço está no estado PRONTO, aguardando o processador; a interferência é causada por serviços mais prioritários;
- f) tempo de bloqueio  $B_i$ : equivale à soma dos intervalos de tempo em que o serviço está no estado (SUSPENSO/BLOQUEADO), esperando pela liberação de recursos compartilhados; o cálculo de  $B_i$  depende das seções críticas dos serviços bloqueantes e das políticas de escalonamento;
- g) tempo de resposta (relativo à ativação)  $R_i$ : equivale ao intervalo de tempo transcorrido desde a ativação até o término do serviço; o tempo de resposta determina a escalonabilidade da tarefa; a tarefa é escalonável somente se  $R_i \leq D_i$ , e o aplicativo é escalonável somente se todas as tarefas que o compõem forem escalonáveis.

A FIGURA 2.8 e a FIGURA 2.9 mostram mais algumas anotações de tempos relacionados com suspensão voluntária, tempos de flutuação de liberação e de deslocamento. A saber:

- h) tempo de suspensão  $V_i$ : equivale à soma dos intervalos de tempo em que o serviço está no estado SUSPENSO(/BLOQUEADO) voluntariamente por chamadas do tipo `delay()`; durante este tempo o processador pode ser alocado a outros serviços;
- i) tempo de flutuação (de liberação)  $J_i$ : equivale ao intervalo de tempo máximo entre a ativação e a liberação do serviço; o serviço pode ser liberado imediatamente ou com um atraso de até  $J_i$ ; diversas situações podem ser modeladas utilizando a flutuação de liberação, tais como o atraso gerado pela de estratégia de escalonamento diferido e as restrições de precedência em grupos de tarefas que constituem transações;
- j) tempo de resposta (relativo à liberação)  $r_i$ : equivale ao intervalo de tempo transcorrido desde a liberação até o término do serviço; o tempo de resposta final  $R_i$  resulta tempo de resposta (relativo à liberação)  $r_i$  acrescido do tempo de flutuação de liberação e ou deslocamento;

- k) tempo de deslocamento  $O_i$ : equivale ao intervalo de tempo entre a ativação e a liberação do serviço; o serviço não pode ser liberado antes de  $O_i$ ; o deslocamento pode ser utilizado na modelagem das restrições de precedência em grupos de tarefas que constituem transações; a combinação de  $J_i$  com  $O_i$  estabelecem um mínimo e máximo de flutuação de liberação.

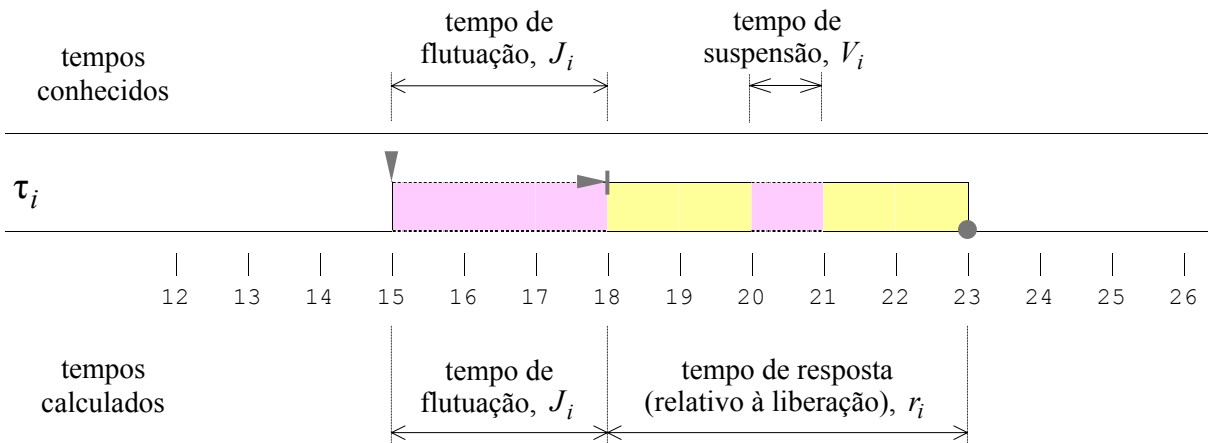


FIGURA 2.8: Tempo de flutuação de liberação em serviços

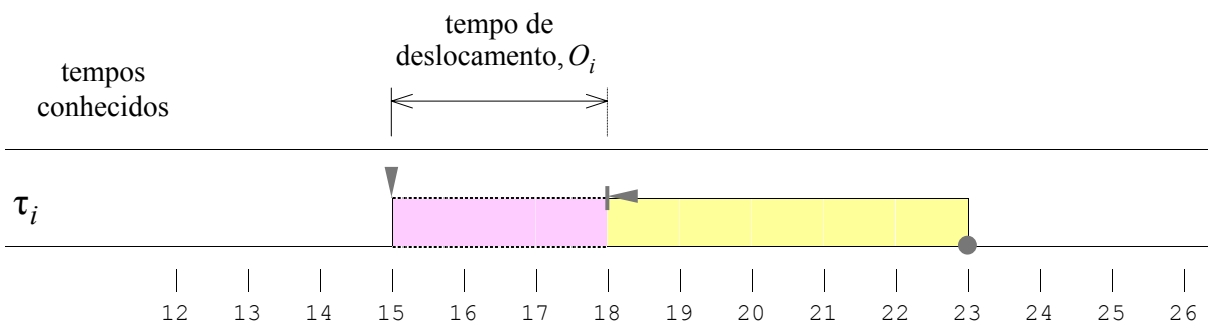


FIGURA 2.9: Tempo de deslocamento de liberação em serviços

A flutuação de término é a variação do tempo de resposta de um serviço para outro. É a diferença entre o pior e o melhor tempo de resposta. A flutuação de término tem importância em transações, permitindo determinar a flutuação de liberação (ou partida) do serviço  $i+1$  com relação ao serviço  $i$  na cadeia de precedência. A FIGURA 2.10 mostra a anotação do tempo da flutuação de término em relação ao tempo de resposta máximo e mínimo.



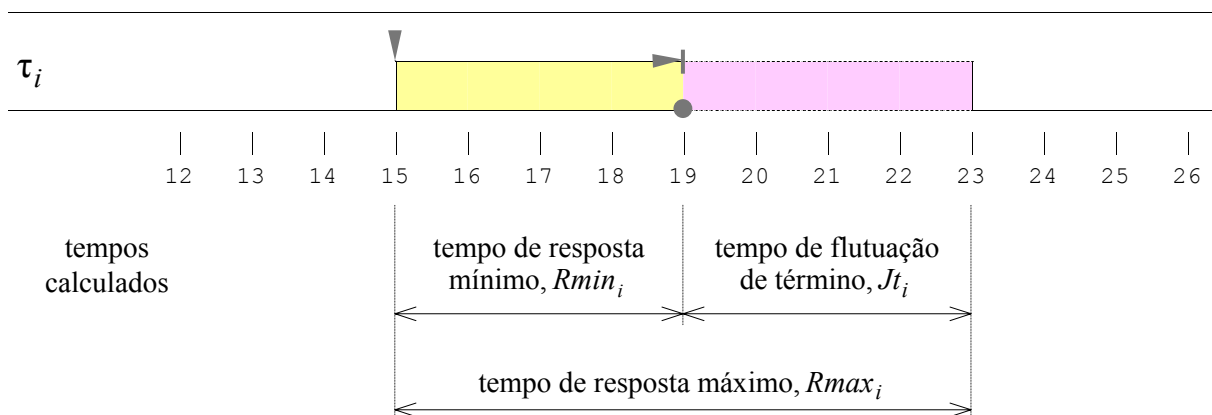


FIGURA 2.10: Tempo de flutuação de término em serviços

#### 2.4.2 Valoração das medidas de tempos

Pelas definições vistas anteriormente, as medidas de tempos são valores fixos bem conhecidos. Porém, aplicadas a sistemas reais o quadro muda completamente. Nos sistemas reais, incluindo os programas computacionais, os tempos variam, e a inter-relação dos componentes propaga estas variações. Como sistemas em tempo real impõem restrições de tempo severas (na maioria de seus componentes), é necessário limitar de alguma forma os valores das medidas de tempos.

A FIGURA 2.11 mostra os pontos mais importantes na linha de tempo com relação ao tempo em execução de programas, a saber:

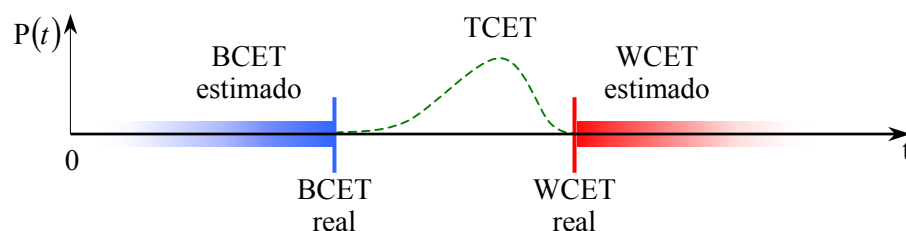


FIGURA 2.11: Valoração de medidas de tempo

- BCET real: é o melhor caso de tempo em execução real e estabelece um limite teórico inferior para a estimativa de tempo;

- WCET real: é o pior caso de tempo em execução real e estabelece um limite teórico superior para a estimativa de tempo;
- BCET estimado ou simplesmente BCET: é o resultado do processo de estimação do BCET real; o BCET estimado deve ser menor ou igual ao BCET real, e quanto menor a diferença entre eles, maior a precisão da estimativa;
- WCET estimado ou simplesmente WCET: é o resultado do processo de estimação do WCET real; o WCET estimado deve ser maior ou igual ao WCET real, e quanto menor a diferença entre eles, maior a precisão da estimativa;
- $P(t)$ : é a probabilidade de ocorrência de casos para os quais o tempo em execução é igual a  $t$  em que  $\overline{BCET \leq t \leq WCET} \rightarrow P(t) = 0$ ; (a distribuição de probabilidade mostrada na figura é meramente ilustrativa.)
- TCET: é o caso típico de tempo em execução; os critérios para estabelecer-se o TCET são diversos, por exemplo, o tempo médio  $(WCET - BCET)/2$  ou o instante  $t$  com a probabilidade  $P(t)$  máxima.

Embora os valores WCET, BCET e TCET estejam relacionados diretamente com o tempo de computação  $C_i$ , a noção de pior caso, melhor caso e caso típico também aplicam-se às medidas de tempos  $B_i$ ,  $C_{i,S}$ ,  $R_i$ ,  $r_i$ ,  $I_i$ ,  $J_i$  e  $V_i$  apresentadas na seção anterior. Para referenciar os casos de todas estas medidas sugere-se o uso de WCxT, BCxT e TCxT.

Para a teoria de escalonamento interessam somente os valores WCxT, uma vez que a escalonabilidade de um aplicativo deve ser garantida para o pior caso e por consequência para todos os casos. Os valores BCxT, TCxT e  $P_i(t)$  são interessantes para animação e simulação.

## 2.5 CLASSIFICAÇÃO DE TAREFAS

As tarefas em tempo real podem ser classificadas quanto ao padrão de ativação e fenômenos de disparo em três tipos: periódicas, aperiódicas e esporádicas. A FIGURA 2.12 mostra os diagramas de Gantt para a distribuição das ativações de cada tipo.

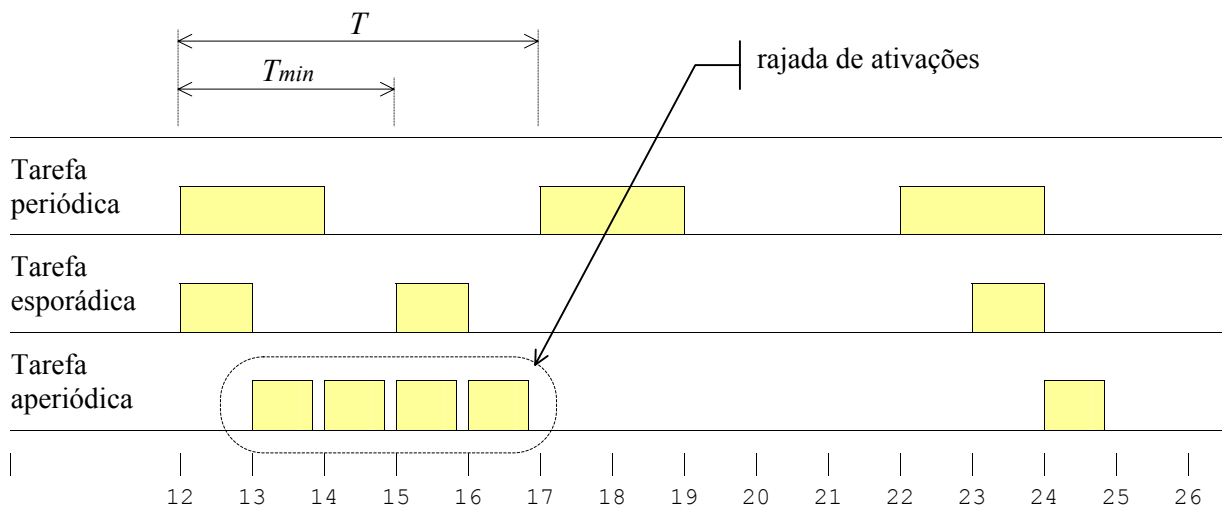


FIGURA 2.12: Distribuição dos serviços de tarefas no tempo

A periodicidade é uma das características mais importantes para a formulação de um modelo analítico que permite prever e garantir os tempos de computação dentro de intervalos aceitáveis. Assim, as tarefas aperiódicas e esporádicas devem ser transformadas em tarefas periódicas, mesmo que isto conduza a situações de utilização do processador muito pessimistas.

Além dos tipos listados acima, um aplicativo em tempo real pode conter tarefas sem limitações de prazo, por exemplo, rotinas em retaguarda para verificação da funcionalidade de dispositivos. As tarefas deste tipo podem ser ignoradas na análise de escalonamento desde que não interfiram nas demais tarefas (BRIAND; ROY, 1999).

### 2.5.1 Tarefas periódicas

As tarefas periódicas consistem em uma seqüência (ilimitada) de serviços idênticos lançados a intervalos fixos de tempo (BUTTAZZO, 1997). O padrão de

ativação destas tarefas é dirigido por tempo. As tarefas possuem dois parâmetros característicos: a) o período  $T$ , que equivale ao tempo entre duas ativações consecutivas; e b) a fase  $\varphi$ , que equivale ao intervalo de tempo entre um referencial externo e a primeira ativação. Um caso típico são as atividades periódicas de coleta de dados, computação de algum resultado e a disponibilização de um resultado. Por exemplo, uma tarefa lê a pressão em um manômetro a cada 200ms.

### 2.5.2 Tarefas aperiódicas

As tarefas aperiódicas consistem em uma seqüência de serviços lançados aleatoriamente, em geral como resposta a algum evento externo (SPRUNT; SHA; LEHOCZKY, 1989). O padrão de ativação destas tarefas é dirigido por eventos. Dada a arbitrariedade das ativações, fenômeno conhecido como "rajada de ativações", é impossível garantir que um conjunto de tarefas possa tratá-las adequadamente. Dada às suas características, de um modo geral as tarefas aperiódicas não são interessantes para aplicativos em tempo real. No entanto, é possível acomodá-las em servidores de diferimento e servidores esporádicos (BRIAND; ROY, 1999). Com o artifício dos servidores as tarefas aperiódicas são transformadas em tarefas periódicas.

### 2.5.3 Tarefas esporádicas

As tarefas esporádicas constituem um subconjunto especial de tarefas aperiódicas para as quais se estabelece a menor taxa de ativações, que equivale a um período fixo mínimo  $T_{min}$  entre ativações consecutivas (SPRUNT; SHA; LEHOCZKY, 1989). Com este artifício as tarefas esporádicas são transformadas em tarefas periódicas. Tipicamente estas tarefas modelam o tratamento de interrupções, sobrecargas de escalonamento e comunicação.

## 2.6 PRIORIDADES E FILAS

A determinação da ordem de atendimento aos serviços, em geral leva em conta dois critérios: sua prioridade e o tempo de espera por processador (BRIAND; ROY, 1999). Os escalonadores interpretam estes critérios com grande diversidade, por exemplo, a prioridade pode estar relacionada com a importância da tarefa ou grupo de tarefas, a espera pode ser resolvida por PRIMEIRO A CHEGAR PRIMEIRO ATENDIDO (PCPA) ou, ainda, a combinação de ambos.

As filas constituem o mecanismo apropriado para apoio à administração da ordem de atendimento aos serviços. Em geral, os escalonadores são construídos ao redor de esquemas de filas próprios para suas estratégias e políticas de escalonamento. Isto é assunto do próximo capítulo.

### 2.6.1 Prioridade estrita

Do ponto de vista da teoria de escalonamento, "prioridade" é um atributo da tarefa que figura como critério suficiente e necessário para administrar a ordem de atendimento aos serviços. Assim, cada tarefa  $\tau_i$  possui uma prioridade única  $P_i$ , que estabelece um conjunto ordenado (total) de tarefas  $\Gamma = (\tau_1, \tau_2, \dots, \tau_i, \dots, \tau_n)$ . As prioridades são mapeadas sobre números naturais em que 1 corresponde à menor prioridade.

As funções listadas abaixo, usadas ao longo do próximo capítulo, definem subconjuntos de tarefas do conjunto  $\Gamma$ :

- $hp(i)$  o conjunto de tarefas com prioridade maior que da tarefa  $\tau_i$ ;
- $hep(i)$  o conjunto de tarefas com prioridade maior ou igual a da tarefa  $\tau_i$ ;
- $lp(i)$  o conjunto de tarefas com prioridade menor que da tarefa  $\tau_i$ .

É importante observar que esta noção de prioridade dispensa qualquer esquema de filas notacional. O conjunto (ordenado) das tarefas no estado PRONTO atende diretamente a seleção da tarefa a receber o controle do processador.

### 2.6.2 Inversão de prioridade

Em ambientes de disputa por recursos compartilhados, serviços com maior prioridade podem ficar bloqueados, quando não conseguem alocar algum recurso que já esteja alocado por outro serviço com menor prioridade. Este fenômeno, conhecido por “inversão de prioridade”, produz um efeito em que o tempo de resposta de serviços com maior prioridade é atrasado por serviços de menor prioridade. O escalonamento deve prover estratégias para contornar este fenômeno.

### 2.6.3 Impasse de bloqueio

Mais fracamente ligada à questão da prioridade, existe outro fenômeno em ambientes de disputa por recursos compartilhados, conhecido por “impasse de bloqueio”, em que serviços envolvidos com recursos bloqueiam-se mutuamente. A FIGURA 2.13 mostra esta situação para as tarefas  $\tau_a$  e  $\tau_b$  e os recursos  $S_1$  e  $S_2$ .

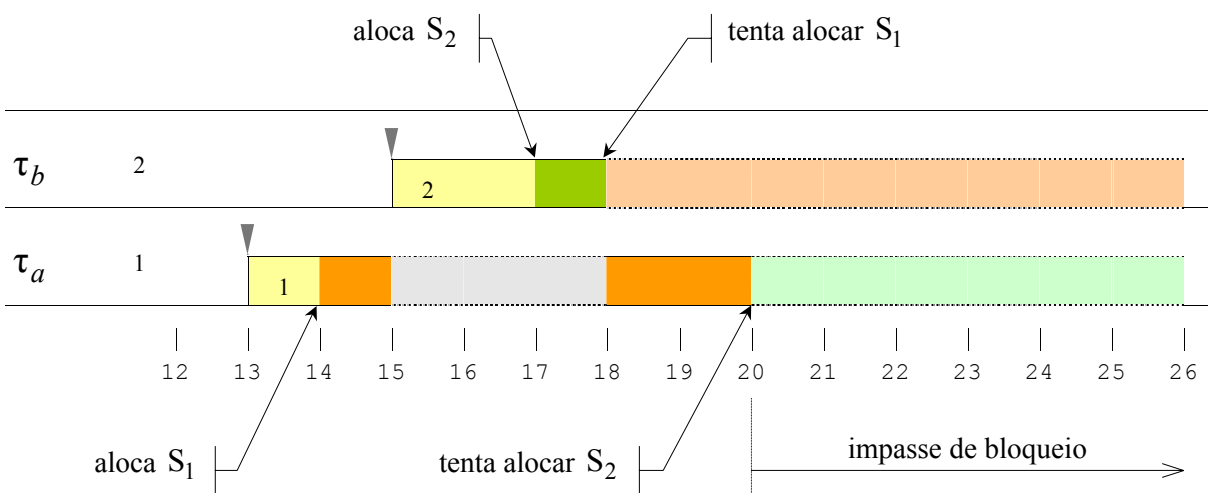


FIGURA 2.13: Exemplo de impasse de bloqueio

A tarefa  $\tau_a$  aloca o recurso  $S_1$  e em seguida é preemptada pela tarefa  $\tau_b$ . Esta por sua vez aloca o recurso  $S_2$  e em seguida tenta alocar o recurso  $S_1$ , mas não consegue porque já está alocado por outra tarefa e entra no estado (SUSPENSO)/BLOQUEADO. A tarefa  $\tau_a$  volta ao estado EXECUTANDO e tenta alocar o

recurso  $S_2$ , mas também não consegue porque já está alocado por outra tarefa. Configura-se assim um impasse de bloqueio em que as tarefas envolvidas não mais podem receber o controle do processador. O escalonamento deve prover estratégias para contornar este fenômeno.

## 2.7 AMBIENTES PREEMPTIVOS E NÃO-PREEMPTIVOS

As tarefas que compõem um aplicativo em tempo real são essencialmente cooperativas e disputam os mesmos recursos, em particular, o processador. Existem duas famílias principais de estratégias de alocação do processador para as tarefas: a) o escalonamento não-preemptivo (FIGURA 2.14); e b) o escalonamento preemptivo (FIGURA 2.15). Entre estes dois extremos existem estratégias alternativas conhecidas por preempção diferida (BURNS; WELLINGS, 2001).

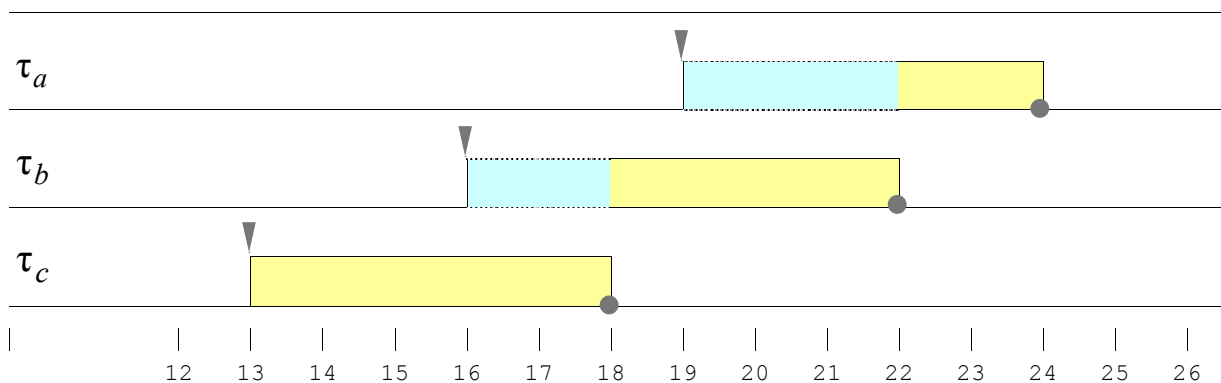


FIGURA 2.14: Exemplo de escalonamento não-preemptivo

Diagrama de Gantt para três tarefas com prioridades  $P_a > P_b > P_c$ . Quando  $\tau_b$  é lançada no instante 16 deveria receber o controle do processador, mas fica bloqueada até o instante 18 quando  $\tau_c$  termina. O mesmo raciocínio aplica-se à tarefa  $\tau_a$ .

### 2.7.1 Escalonamento não-preemptivo

Sob o escalonamento não-preemptivo, o serviço de uma tarefa executa até ceder a vez voluntariamente, quando outra tarefa pode receber o controle do processador. Esta cessão dá-se explícita ou implicitamente.

Nos casos explícitos, a tarefa solicita a revisão do escalonamento, por exemplo, pela execução da chamada `yield()` disponível em diversos ambientes operacionais que suportam a não-preempção. Se alguma tarefa estiver na vez para executar, então a tarefa atual será reescalonada, e a nova tarefa receberá o controle do processador.

Nos casos implícitos, o serviço da tarefa perde o controle do processador quando termina ou quando sofre suspensões ou bloqueios de processamento. Suspensões e bloqueios são devidos à solicitação de entrada/saída, espera por liberação de recursos travados, espera por encontros de sincronização ou comunicação e execução de chamadas, tais como `delay()` e `sleep()`.

Apesar da popularidade do escalonamento não-preemptivo dada a simplicidade de implementação, por exemplo, não necessita de proteção para recursos compartilhados<sup>1</sup>, as vantagens não compensam os pontos negativos. Em particular, determinada tarefa pode executar indefinidamente sem que outras tarefas possam executar. Assim, os tempos de bloqueio tornam-se ilimitados (BRIAND; ROY, 1999).

### 2.7.2 Escalonamento preemptivo

Sob o escalonamento preemptivo, o serviço de uma tarefa pode ser suspenso temporariamente em qualquer ponto e a qualquer instante sempre que outra tarefa esteja na vez para executar. Este estado de suspensão denomina-se "preemptido".

Além disso, o serviço pode ceder a vez voluntariamente nos casos implícitos apresentados na seção anterior. A cessão explícita não faz sentido, pois, o controle voltaria para a própria tarefa que certamente está na vez.

Os serviços de tarefas podem desativar temporariamente a preempção com auxílio de um recurso compartilhado. Este mecanismo permite criar seções críticas que não possam ser preemptidas. O impacto destes bloqueios deve ser considerado na análise de escalonabilidade.

---

<sup>1</sup> O planejamento do uso dos recursos compartilhados está completamente nas mãos do projetista, que sabe exatamente quando a tarefa recebe e perde o controle do processador.



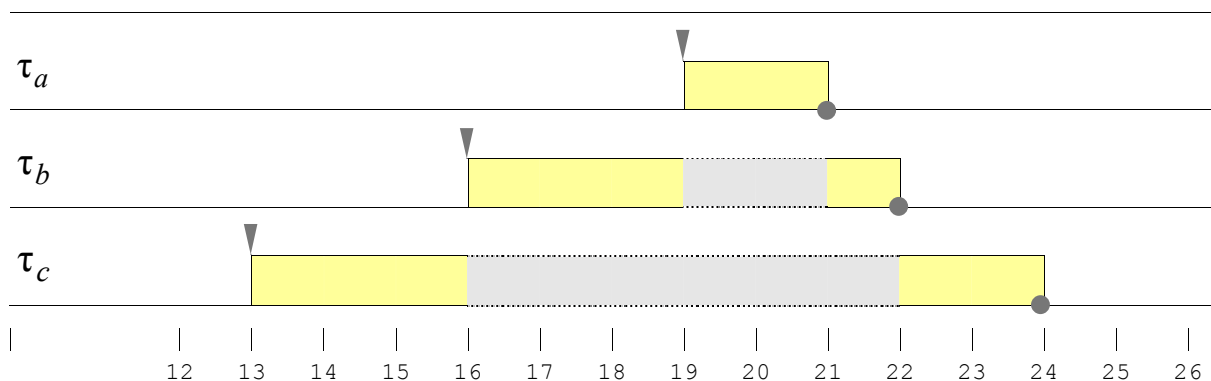


FIGURA 2.15: Exemplo de escalonamento preemptivo

Diagrama de Gantt para três tarefas com prioridades  $P_a > P_b > P_c$ . Quando  $\tau_b$  é lançada no instante 16 a tarefa  $\tau_c$  fica preemptida até o instante 22. Quando a tarefa  $\tau_a$  é lançada no instante 19 a tarefa  $\tau_b$  fica preemptida até o instante 21.

Uma vez que os serviços das tarefas podem ser interrompidos em qualquer ponto, o controle de acesso a recursos compartilhados necessita de travas para garantir a exclusividade de acesso. Além disso requer cuidados especiais com situações, tais como impasses de bloqueio e inversões de prioridade.

### 2.7.3 Escalonamento com preempção diferida

A preempção diferida consiste em dividir o tempo do processador em pequenos blocos não-preemptíveis. Assim, o serviço de uma tarefa não pode ser preemptido imediatamente, mas em determinados cortes de tempo. Os critérios para estabelecimentos destes cortes podem ser diversos, formando blocos de tamanho fixo ou variável.

Neste trabalho interessa, no entanto, um caso particular em que o escalonador é ativado a intervalos de tempo fixos. Em cada ativação o escalonador decide qual tarefa deve receber (ou continuar com) o controle do processador.

Esta estratégia possui duas vantagens: a) aumento da escalonabilidade do aplicativo, reduzindo o tempo de computação da tarefa devido à redução do número de passagens de controle do processador entre as tarefas; e b) melhora da estimativa do pior tempo de execução dos blocos não-preemptíveis, que pode levar em consideração

os tempos ganhos com os recursos dos processadores modernos, tais como *caches*, *prefetch queues* e *pipelines* (BURNS; WELLINGS, 2001).

## 2.8 AMBIENTES N-PROCESSADORES

Os ambientes que dão suporte às tarefas podem ser classificados quanto à arquitetura e configuração de seus processadores de acordo com a QUADRO 2.1 abaixo.

QUADRO 2.1: Classificação dos ambientes n-processadores

<b>ambiente</b>	<b>acoplamento</b>	<b>simetria</b>	<b>heterogêneo</b>	<b>homogêneo</b>
monoprocessado			–	X
multiprocessado	fraco		X	–
	forte	assimétrico	X	–
		simétrico	X	X

A tabela apresenta três colunas principais para classificação dos ambientes: a) em relação à quantidade de processadores que compõem o ambiente; b) em relação ao tipo de acoplamento dos processadores e simetria; e c) uma classificação baseada na variedade de processadores que compõem o ambiente (BURNS; WELLINGS, 2001).

A teoria de escalonamento propõe abordagens distintas para os diversos ambientes, com maior ou menor complexidade para o escalonamento das tarefas. A literatura apresenta pelo menos três grupos de abordagens principais: a) abordagens tradicionais para ambientes monoprocessados; b) abordagens especializadas em ambientes multiprocessados com acoplamento forte e simétricos; e c) abordagens para ambientes distribuídos que englobam as demais configurações.

Uma verificação importante a respeito de ambientes multiprocessados é o reconhecimento da relação restritiva "pode executar" entre processadores e tarefa mostrada na FIGURA 2.16.

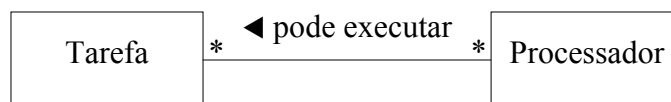


FIGURA 2.16: Relação entre processadores e tarefas

Nesta seção dá-se uma visão geral dos ambientes para aplicativos em tempo real embora o trabalho inicialmente esteja restrito aos ambientes monoprocessados.

### 2.8.1 Acoplamento fraco

O acoplamento fraco caracteriza-se pelo fato dos processadores não terem acesso a uma memória comum. Desta forma a sincronização e a comunicação entre tarefas deve ser realizada através de um esquema de passagem de mensagens. Ambientes deste tipo são conhecidos como sistemas distribuídos. Cada nodo tem seu próprio relógio, e estes relógios estão fracamente sincronizados diferindo entre si por um valor limitado e conhecido (TANENBAUM, 1988).

Os ambientes fracamente acoplados não são necessariamente puros. Algum nodo pode ser um sub-ambiente multiprocessado fortemente acoplado.

### 2.8.2 Acoplamento forte

O acoplamento forte caracteriza-se pelo fato dos processadores terem acesso a uma memória comum. Desta forma a sincronização e a comunicação entre tarefas pode ser realizada através de variáveis compartilhadas.

A arquitetura dos processadores pode ser assimétrica ou simétrica. No primeiro caso, cada processador cumpre uma função específica no sistema. Por exemplo, o processador trata do protocolo de comunicação em uma rede. Em geral, os ambientes não são necessariamente puros. Um sistema de computação pode ser composto por processadores simétricos e assimétricos.

No caso de arquiteturas simétricas, os  $n$  processadores que compõem o ambiente têm acesso comum aos dispositivos de entrada/saída e às fontes de interrupções (INTEL, 1997). Determinada tarefa pode ser executada em qualquer um dos processadores a qualquer tempo sem prejuízo de sua funcionalidade. Esta troca de processadores tem um custo adicional devido aos recursos independentes destes, tais como *cache*, *prefetch queues* e *pipelines*, que podem descartar dados favoráveis à tarefa corrente.

### 2.8.3 Ambientes homogêneos e heterogêneos

Esta classificação restringe-se à diversidade dos processadores que compõem o ambiente. Sistemas homogêneos são compostos por processadores do mesmo tipo. Sistemas heterogêneos por sua vez são compostos por processadores de tipos diferentes e impõem problemas de representação de dados e programas que podem impactar a comunicação entre processadores (HERLIHY; LISKOV, 1982).

## 2.9 CONCLUSÃO

Neste capítulo apresentou-se diversos conceitos e aspectos relativos a sistemas em tempo real necessários para a compreensão dos próximos capítulos. Além disso unificou-se parte da terminologia em torno dos conceitos na língua alvo do trabalho.

Mostrou-se diversos conceitos básicos, tais como: a) pontualidade e suas categorias: severo, rígido e fraco; b) modelo típico de aplicativos em tempo real; c) definição de tarefas, serviços e seus ciclos de vida e estados; d) medidas de tempos: período de ativação de serviços, prazo de término, tempo de resposta, tempo de interferência e tempo de uso de recursos compartilhados; e) classificação das tarefas quanto à ativação em periódicas, esporádicas e aperiódicas; e f) ambientes preemptivos e não-preemptivos, monoprocessados e n-processados.

Apresentou-se um modelo para representação dos tempos através de diagramas de Gantt com anotações de eventos, tais como instante de ativação, prazo e flutuações de liberação e término.

Mostrou-se o conceito de prioridade, o fenômeno de inversão de prioridade e o impasse de bloqueio de recursos compartilhados.



### 3 FUNDAMENTOS DO ESCALONAMENTO

No capítulo anterior apresentou-se os principais conceitos a respeito de aplicativos em tempo real e suas composições, entre outros, os conceitos de tarefa, serviço, medidas de tempos e classificações. Neste capítulo apresenta-se a teoria clássica de escalonamento orientada a aplicativos em tempo real. Trata-se ainda da classificação, vantagens e desvantagens dos escalonadores, e da visão analítica da escalonabilidade.

#### 3.1 INTRODUÇÃO

De um modo geral, os ambientes de processamento lidam com um ou mais aplicativos concorrentes competitivos e/ou cooperativos. Os primeiros são de propósito geral e acomodam diversas tarefas ou serviços interessadas em obter o máximo dos recursos para si. Na maior parte dos casos não é possível determinar *a priori* a carga de serviços que chegam e saem continuamente. A natureza destes ambientes requer políticas de escalonamento capazes de lidar com esta imprevisibilidade.

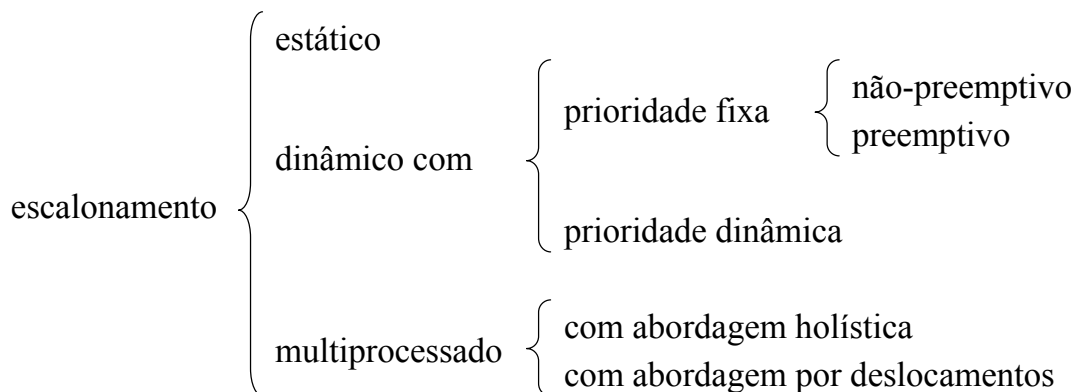
Os aplicativos cooperativos, como sistemas em tempo real, em geral são compostos por um conjunto previamente conhecido de tarefas e serviços, tem um propósito específico e distribuem os recursos de modo a beneficiar o todo. Os sistemas embarcados representam precisamente estes aplicativos. No entanto, aplicativos cooperativos podem concorrer com aplicativos concorrentes no mesmo ambiente de processamento em que aqueles possuem reservas garantidas de recursos.

Por fim, supõe-se a necessidade de um escalonador executivo com o propósito de distribuir o recurso processador e demais recursos entre as tarefas e serviços de acordo com uma política de escalonamento.

### 3.1.1 Classificação dos escalonadores

Existem inúmeras políticas de escalonamento, que podem ser classificadas de acordo com o QUADRO 3.1 (BUTTAZZO, 1997).

QUADRO 3.1: Classificação das políticas de escalonamento



No escalonamento estático, a ordem de execução dos serviços é pré-estabelecida pelo projeto, e o escalonador simplesmente realiza esta ordem. Tem baixo custo de processamento e é de fácil implementação, porém, de características contrárias à extensibilidade e flexibilidade, entre outras. Por exemplo, não suporta criação dinâmica de tarefas.

No escalonamento dinâmico, a ordem de execução dos serviços é determinada pelo escalonador com base em políticas de escalonamento. Tem maior custo de processamento, no entanto tem boas características de extensibilidade e flexibilidade. Por exemplo, suporta criação dinâmica de tarefas e troca de prioridades.

As categorias de escalonamento dinâmico serão apresentadas em detalhes nas próximas seções sob a luz da Análise por Taxas Monotônicas (RMA) e Análise por Prazos Monotônicos (DMA), como aspectos da teoria de escalonamento orientada para aplicativos em tempo real.



### 3.1.2 Exemplos de políticas de escalonamento

Nesta seção apresenta-se um resumo das políticas de escalonamento relevantes, excluindo as políticas que se enquadram em RMA e DMA. Estas últimas serão tratadas nas seções de análises de escalonabilidade.

#### 3.1.2.1 Primeiro a Chegar Primeiro Atendido

A política PRIMEIRO A CHEGAR PRIMEIRO ATENDIDO (FCFS, PCPA) é de escalonamento dinâmico em ambientes não-preemptivos em que a ordem de execução dos serviços é determinada pela ordem em que eles entram no estado PRONTO. Raramente é utilizada em aplicativos em tempo real, a não ser como mecanismo de desempate entre tarefas e serviços de uma mesma fila de prioridade em escalonadores mais complexos.

#### 3.1.2.2 Round Robin

A política *ROUND ROBIN* (RR) é semelhante à PCPA no tratamento da ordem de execução dos serviços. No entanto, cada serviço recebe o controle do processador por um intervalo de tempo máximo  $T_q$ , normalmente entre 10 a 100ms (TANENBAUM; WOODHUL, 2000). Se o serviço não terminar neste intervalo de tempo, será preemptido e deverá aguardar o próximo intervalo para continuar. Caracteriza-se assim um ambiente preemptivo. As tarefas podem ceder voluntariamente o controle do processador antes do término de seu intervalo de tempo.

A FIGURA 3.1 mostra um conjunto de três tarefas escalonadas por RR. No exemplo,  $T_q$  é igual a 20ms, tempo máximo para cada serviço ocupar o processador.

Os serviços são ativados na ordem  $\tau_1$ ,  $\tau_2$  e  $\tau_3$  e são atendidos na mesma ordem até terminarem. As duas últimas tarefas foram ativadas fora de fase, portanto apresentam flutuação de liberação.

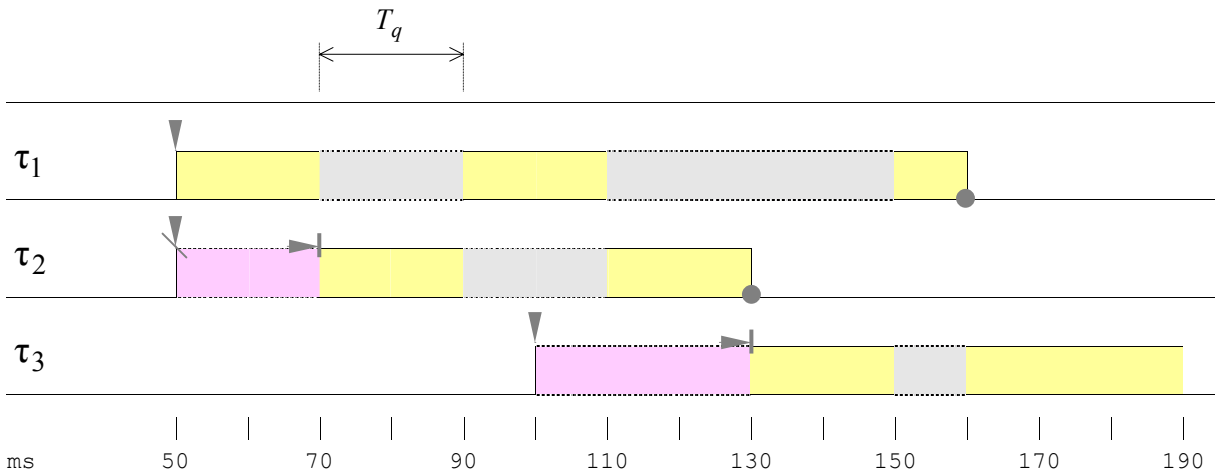


FIGURA 3.1: Exemplo de escalonamento *Round Robin*

O desempenho desta política é fortemente influenciada pelo tempo  $T_q$ , pelo tempo de execução das tarefas e pelo número de tarefas ativadas. O resultado final situa-se entre o comportamento semelhante à PCPA, quando  $T_q$  é maior ou igual ao tempo de execução da tarefa mais demorada, e ineficiência total, quando  $T_q$  é menor ou igual ao tempo gasto pelo mecanismo de preempção.

### 3.1.2.3 Executivo Cíclico

Para entender o EXECUTIVO CÍCLICO (CE), considere-se o conjunto de tarefas periódicas  $\tau_1$ ,  $\tau_2$  e  $\tau_3$ . A tarefa  $\tau_1$  possui um período de 4ms com 1ms de tempo de processamento a cada período, a tarefa  $\tau_2$  possui um período de 6ms com 2ms de tempo de processamento a cada período, e a tarefa  $\tau_3$  possui um período de 12ms com 4ms de tempo de processamento a cada período. A FIGURA 3.2 mostra um diagrama de tempos possível para escalonar o conjunto de tarefas.

Primeiro determina-se um ciclo maior, que corresponde ao intervalo de tempo necessário para atender a todas as tarefas pelo menos uma vez. Mais precisamente, o ciclo maior deve ser igual ao MMC dos períodos das tarefas. Deve ser o menor possível para viabilizar a construção de um algoritmo escalonador, e para tanto os períodos das tarefas devem ser arranjados neste sentido. Cada ciclo maior deve realizar exatamente a mesma seqüência de ativações de serviços.

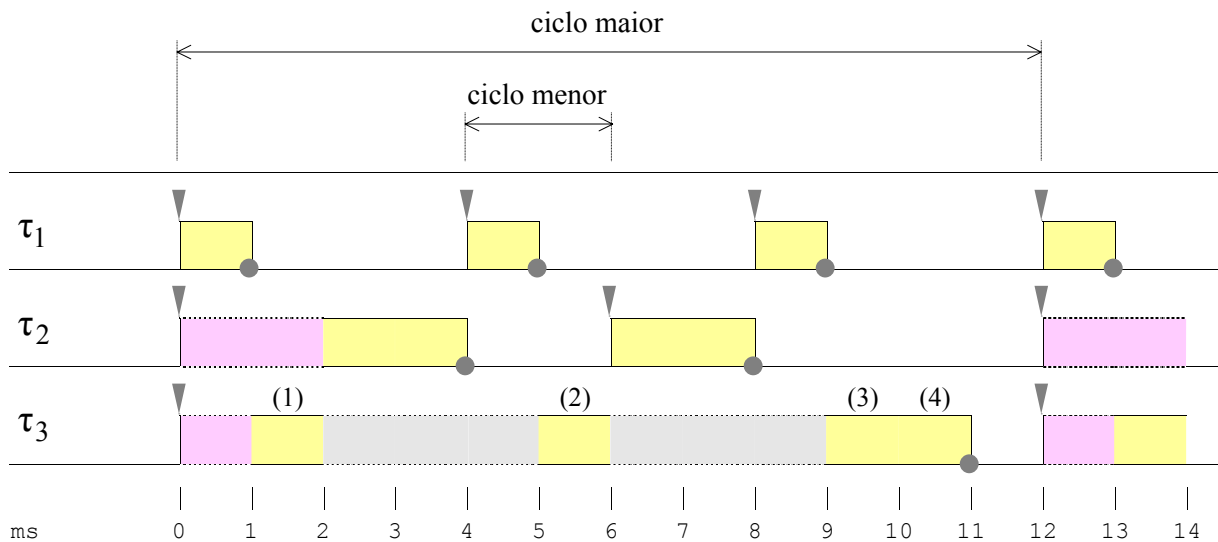


FIGURA 3.2: Exemplo de escalonamento com o Executivo Cíclico

Em seguida determina-se um ciclo menor, que corresponde ao intervalo de tempo no qual os serviços ou parte destes recebem o controle do processador. O ciclo menor deve garantir que um serviço não receba o controle do processador antes de sua ativação, por exemplo, o serviço de  $\tau_2$  ativado no tempo 6 só poderá receber o controle do processador no intervalo de 6 a 12ms. Mais precisamente, o ciclo menor deve ser igual ao MDC dos períodos das tarefas. Os períodos das tarefas devem ser arranjados de modo a maximizar o tempo do ciclo menor para reduzir a fragmentação do código das tarefas.

No exemplo, os tempos foram determinados favoravelmente, assim o ciclo maior é de 12ms e o ciclo menor de 2ms. Pelo diagrama de Gantt da FIGURA 3.2 observa-se que as tarefas  $\tau_1$  e  $\tau_2$  encaixam-se perfeitamente no ciclo menor e que a tarefa  $\tau_3$  está fragmentada em quatro sub-rotinas com 1ms de processamento cada uma. O ALGORITMO 3.1 propõe um escalonador para o exemplo (BURNS; WELLINGS, 2001).

Pelo algoritmo, o ciclo maior está subdividido em 6 ocorrências do ciclo menor, cada uma com o início marcado por um relógio de 2 ms. O processamento das tarefas está distribuído nos intervalos do ciclo menor.

### ALGORITMO 3.1: Exemplo de um escalonador executivo cíclico

---

```
loop
  esperaTiqueDoCicloMenor;
  executaT1;
  executaT3_Part1;
  esperaTiqueDoCicloMenor;
  executaT2;
  esperaTiqueDoCicloMenor;
  executaT1;
  executaT3_Part2;
  esperaTiqueDoCicloMenor;
  executaT2;
  esperaTiqueDoCicloMenor;
  executaT1;
  executaT3_Part3;
  esperaTiqueDoCicloMenor;
  executaT3_Part4;
end loop;
```

---

#### 3.1.2.4 Prazo Mais Próximo Primeiro

A política PRAZO MAIS PRÓXIMO PRIMEIRO (EDF) seleciona o serviço com o prazo mais cedo dentre o conjunto de serviços que estão no estado PRONTO. No ambiente não-preemptivo, o serviço selecionado executa até o final, até entrar no estado SUSPENSO/BLOQUEADO ou até entrar voluntariamente no estado PRONTO. No ambiente preemptivo, o serviço pode ser preemptido por outro serviço com prazo mais próximo que este.

A FIGURA 3.3 mostra um exemplo de escalonamento EDF preemptivo. Os tempos a seguir são considerados absolutos. O serviço  $\tau_3$  está executando a partir do instante 13. Quando o serviço  $\tau_2$  é ativado no instante 16, o serviço  $\tau_3$  é preemptido porque seu prazo 26 é maior que o prazo 23 do serviço  $\tau_2$ . Quando o serviço  $\tau_1$  é ativado no instante 18, já entra preemptido porque seu prazo 25 é maior que o prazo 23 do serviço  $\tau_2$ . No instante 21, o serviço  $\tau_2$  termina, e o controle passa para o serviço  $\tau_1$  com prazo 25, menor que o prazo do serviço  $\tau_3$ .

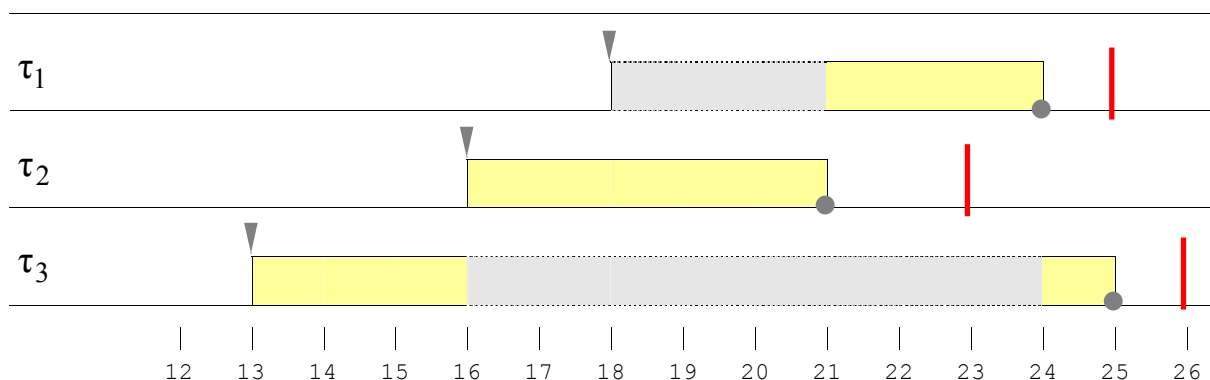


FIGURA 3.3: Exemplo escalonamento EDF preemptivo

O escalonamento EDF não é adequado para ambientes não-preemptivos dada a interferência de serviços demorados com prazos longos em serviços recém-chegados com prazos mais curtos, colocando em risco a garantia teórica de cumprimentos de prazos. Por isso restringe-se as análises ao ambiente preemptivo (FIDGE, 2002).

### 3.1.2.5 Menor Tempo de Relaxamento Primeiro

Define-se como tempo de relaxamento  $L_i$ , a diferença de tempo entre o prazo efetivo e o prazo estimado do serviço (BURNS; WELLINGS, 2001). Assim, a prioridade do serviço é determinada dinamicamente de modo que o serviço com menor tempo de relaxamento tenha a maior prioridade. A política MENOR TEMPO DE RELAXAMENTO PRIMEIRO (LLF) é adequada ao ambiente preemptivo pelas mesmas razões do escalonamento EDF. A FIGURA 3.4 mostra um exemplo com o tempo de relaxamento. Observa-se que este tempo tende a diminuir constantemente.

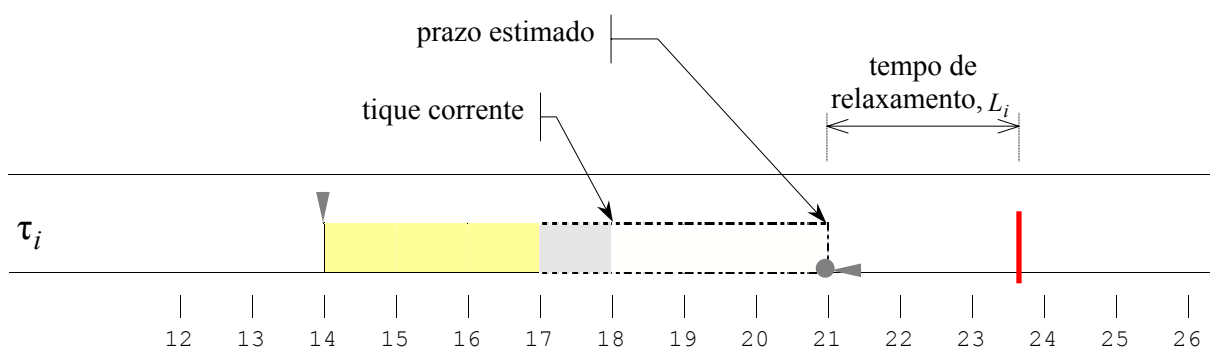


FIGURA 3.4: Definição de tempo de relaxamento

Enquanto um serviço estiver no estado EXECUTANDO, seu prazo permanece inalterado. No entanto, quando estiver nos estados PRONTO e SUSPENSO/BLOQUEADO, seu prazo estimado aumenta e conseqüentemente seu tempo de relaxamento diminui. Em algum momento no futuro será o serviço com maior prioridade.

Quando duas ou mais tarefas possuem tempos de relaxamento muito próximos, entram num "fenômeno pingue-pongue", causando alto número de preempções, afetando negativamente o desempenho do ambiente de processamento.

### 3.1.2.6 Serviço Mais Curto Primeiro

A política SERVIÇO MAIS CURTO PRIMEIRO (SJF) seleciona o serviço com o menor tempo de execução estimado dentre o conjunto de serviços que estão no estado PRONTO. No ambiente não-preemptivo, o serviço selecionado executa até o final, ou até entrar no estado SUSPENSO/BLOQUEADO ou até entrar voluntariamente no estado PRONTO. No ambiente preemptivo, o serviço pode ser preemptido por outro serviço com tempo de execução estimado menor. A FIGURA 3.5 mostra os tempos de execução.

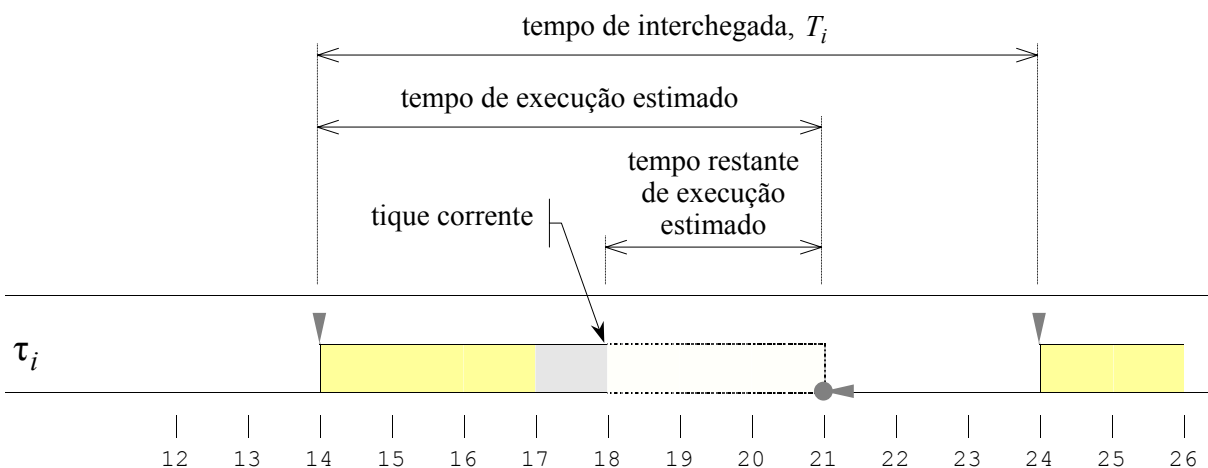


FIGURA 3.5: Tempos de execução estimados

A variante desta política MENOR TEMPO RESTANTE PRIMEIRO (SRTF) considera o tempo restante de execução estimado de cada serviço ao invés do tempo (total) de execução estimado. Os tempos de execução são estimados de duas formas: a) os

tempos são fixos e previamente conhecidos; ou b) pela distribuição de Poisson dos tempos de serviço e de interchegada (período) baseados no histórico de execução das tarefas.

### **3.1.2.7 Maior Prioridade Primeiro**

A política MAIOR PRIORIDADE PRIMEIRO (HPF) determina a ordem de execução das tarefas pela prioridade. As tarefas com maior prioridade são executadas em primeiro lugar. As prioridades são pré-determinadas pelo projeto. Em geral leva-se em conta a importância das tarefas, sua periodicidade e outros atributos conhecidos do aplicativo como um todo. É uma política muito utilizada para aplicativos em tempo real compostos por até 30 tarefas a partir do que a complexidade aumenta e dificulta a determinação das prioridades. Opera tanto em ambientes não-preemptivos como em ambientes preemptivos.

## **3.2 ESCALONAMENTO ESTÁTICO**

No escalonamento estático, a ordem de execução dos serviços é pré-estabelecida pelo projeto, e o escalonador simplesmente realiza esta ordem. O escalonador pode ser genérico ou embutido como parte do código, por exemplo, o Executivo Cíclico.

A abordagem possui vantagens (BURNS; WELLINGS, 2001), tais como:

- a) produz programas determinísticos em que se sabe que tarefa executa a que tempo;
- b) não requer sistemas operacionais para escalonamento;
- c) os serviços apresentam flutuações de liberação e término muito pequenas e bem comportadas que se repetem a cada ciclo maior (LOCKE, 1992);
- d) não possui processos e/ou linhas de controle;
- e) as tarefas residem num mesmo espaço de endereçamento e podem trocar dados entre si diretamente sem necessidade de proteção para acesso concorrente;
- f) tem baixo custo operacional;
- e g) os aplicativos possuem garantia de escalonabilidade provada por construção.

A abordagem do escalonamento estático possui maior número de desvantagens, tais como: a) aplicativos são de difícil projeto em que se deve saber qual serviço executar a que tempo, ao passo em que aplicativos em tempo real apenas necessitam prever sua escalonabilidade (LOCKE, 1992); b) é adequada somente para tarefas periódicas, e no caso de tarefas esporádicas, os serviços devem ser ativados com frequência maior ou igual à expectativa dos eventos a serem tratados, levando a serviços sem trabalho útil; c) acomoda com dificuldade tarefas com período muito longo, situações contornáveis por sub-escalonamentos, por exemplo, a cada  $n$  ocorrências do ciclo maior ativa-se a tarefa com período longo (BURNS; WELLINGS, 2001); d) os serviços devem harmonizar com múltiplos comuns do ciclo maior e menor, e quando isto não acontece aparecem vazios na carga do processador, tornando o aplicativo ineficiente (TINDELL, 2000); e) as tarefas com tempo de processamento maior que o ciclo menor devem ser fragmentadas o que compromete a estrutura do programa e por consequência sua manutenibilidade e confiabilidade (TINDELL, 2000); f) gera implementações monolíticas em que a estrutura resultante do código não reflete a estrutura do aplicativo, dificultando o projeto, a manutenção e a prova de correção do programa; g) as tarefas tornam-se instáveis com a presença de sobrecarga, e caso alguma tarefa exceda seu tempo de processamento por receber algum evento atrasado, as tarefas seguintes poderão sofrer as consequências não obstante sua importância no aplicativo (LOCKE, 1992).

Considerando as vantagens e desvantagens do escalonamento estático, as soluções com múltiplas linhas de controle são preferíveis, uma vez que preservam a estrutura do aplicativo sem introduzir artifícios para harmonizar o código com os ciclos nem arranjar os períodos e outros tempos na tentativa de viabilizar um Executivo Cíclico.

### **3.3 ESCALONAMENTO DINÂMICO**

No escalonamento dinâmico, a ordem de execução dos serviços é determinada pelo escalonador com base em políticas de escalonamento. Requer políticas de



escalonamento mais complexas. Pode ser subdividido em três categorias: a) escalonamento dinâmico não-preemptivo com prioridade fixa; b) escalonamento dinâmico preemptivo com prioridade fixa; e c) escalonamento dinâmico com prioridade dinâmica.

### **3.3.1 Escalonamento dinâmico com prioridade fixa**

No escalonamento dinâmico com prioridade fixa, cada tarefa está associada a uma prioridade estaticamente pré-determinada pelo projeto. Diversos critérios podem ser utilizados para estabelecer as prioridades, por exemplo, o escalonamento HPF. Outros dois critérios são: o escalonamento por RMA em que os serviços com períodos menores recebem prioridades maiores, e o escalonamento por DMA em que os serviços com prazos menores recebem prioridades maiores.

O escalonamento dinâmico com prioridade fixa aplica-se a ambientes preemptivos e não-preemptivos. As figuras da Seção 2.7 exemplificam os dois casos.

#### **3.3.1.1 Escalonamento dinâmico com prioridade fixa não-preemptivo**

O escalonamento com prioridade fixa em ambientes não-preemptivos não permite que serviços sejam preemptidos por outros.

Esta abordagem possui vantagens, tais como: a) oferece um paradigma simples de programação baseado em co-rotinas em que as tarefas cedem o controle do processador voluntariamente para outras tarefas; a implementação lança mão de um número reduzido de construtos para divisão em tarefas, disponíveis em linguagens como Ada (BURNS; WELLINGS, 1996); b) permite que tarefas sejam implementadas separadamente, e que o conjunto de tarefas reflita diretamente a estrutura de um aplicativo em particular, sem possíveis artifícios de escalonamento embutidos como no escalonamento estático; c) o conjunto de tarefas de um aplicativo pode ser analisado formalmente em relação à escalonabilidade, considerando o escalonamento com prioridade fixa; e d) BURNS e WELLINGS (1996) sugerem que o comportamento

simples destes aplicativos é compatível com o padrão DO-178B (RTCA, 1992) nível A.

Além das vantagens, a abordagem do escalonamento com prioridade fixa em ambientes não-preemptivos possui desvantagens, tais como: a) não consegue responder às alterações do conjunto de tarefas no estado PRONTO justamente porque não se pode parar o serviço corrente; desta forma tarefas com maior prioridade podem ser retardadas severamente; e b) mais seriamente, em casos de faltas, tarefas podem monopolizar o processador indefinidamente; isto torna o aplicativo instável em casos de sobrecarga.

### **3.3.1.2 Escalonamento dinâmico com prioridade fixa preemptivo**

O escalonamento com prioridade fixa em ambientes preemptivos permite que serviços sejam preemptidos.

Esta abordagem possui vantagens, tais como: a) garante que tarefas com maior prioridade sejam atendidas mais rapidamente, respeitando a ordem de prioridade estabelecida pelo projeto; as tarefas com menor prioridade são retardadas; b) não há necessidade do projeto preocupar-se com a cessão do controle do processador para outras tarefas; o escalonador executivo determina o momento da passagem do controle para outro serviço por preempção; a implementação e manutenção das tarefas fica mais simples, uma vez que não há necessidade de levar em conta o contexto temporal entre as tarefas; c) em relação ao escalonamento estático, não há necessidade de montar os períodos das tarefas sobre valores harmônicos (TINDELL, 2000), e não se requer que haja relação entre as características de tempo e a prioridade; o projeto pode estabelecer qualquer combinação entre prioridade, período e prazo dos serviços, por exemplo, usando os princípios de escalonamento por RMA e DMA (BURNS; WELLINGS, 2001); d) estão disponíveis diversos sistemas operacionais confiáveis e eficientes que suportam o ambiente preemptivo para tempo real, e) existem provas de escalonabilidade bem estabelecidas, que permitem examinar um dado conjunto de tarefas quanto a seus prazos; e f) em casos de sobrecarga os aplicativos sob controle de

escalonamento preemptivo tornam-se estáveis; as tarefas de menor prioridade podem perder seus prazos em detrimento às tarefas de maior prioridade.

No entanto, a abordagem do escalonamento com prioridade fixa em ambientes preemptivos possui desvantagens para aplicativos embarcados: a) os sistemas operacionais (núcleos) com escalonamento preemptivo são mais complexos, e a preempção demanda mais tempo e memória; b) o escalonamento preemptivo pode gerar flutuações de liberação e término maiores que o escalonamento estático; estas flutuações podem ser minimizadas, estruturando o conjunto de tarefas adequadamente (LOCKE, 1992), ou introduzindo deslocamentos em que se substitui parte do tempo de flutuação de liberação por um tempo de deslocamento; e c) considerando os processos de certificação de aplicativos em tempo real, tais como o padrão DO-178B (RTCA, 1992) que em grande parte requerem testes exaustivos de todas as alternativas de fluxo de controle dos algoritmos, não fica claro como atingir estes requisitos em ambientes preemptivos.

A certificação de aplicativos para ambientes preemptivos através de testes torna-se NP-intensiva devido à diversidade combinatorial do fenômeno da preempção. Torna-se necessário aceitar provas de viabilidade através da teoria de escalonamento.

O fato do escalonamento preemptivo não ser determinístico em que não se sabe qual serviço ou até qual trecho do serviço executa quando, pode ser visto em alguns casos como desvantagem, mas, em geral requer-se apenas que os serviços sejam previsíveis em relação aos prazos (LOCKE, 1992).

### **3.3.2 Escalonamento dinâmico com prioridade dinâmica**

No escalonamento dinâmico com prioridade dinâmica, cada serviço recebe uma prioridade em tempo de execução. Os escalonadores desta categoria são mais complexos comparados aos escalonadores de prioridade fixa. O escalonamento EDF em que os serviços com prazo mais próximo recebem prioridade maior e o escalonamento LLF em que os serviços com menor tempo de relaxamento recebem prioridade maior são dois exemplos bem conhecidos de prioridade dinâmica.

A abordagem do escalonamento dinâmico com prioridade dinâmica possui como principal vantagem a de oferecer em tese a melhor garantia no cumprimento dos prazos de serviços já que aloca o processador à tarefa com maior urgência (LIU; LAYLAND, 1973) (GIERING; BAKER, 1994).

No entanto, esta abordagem possui desvantagens, tais como: a) causa maior carga adicional no tempo de execução comparado ao escalonamento de prioridade fixa, uma vez que as prioridades de todas as tarefas no estado PRONTO devem ser recalculadas a cada ativação ou término de serviço; e b) torna-se instável com sobrecarga transiente, dificultando a previsão de qual tarefa pode perder o prazo. As desvantagens parecem sobrepujar as vantagens teóricas do escalonamento com prioridade dinâmica, dando lugar ao uso preferencial do escalonamento com prioridade fixa.

### **3.4 ANÁLISES DE ESCALONABILIDADE**

Nesta seção trata-se da análise formal e matemática da escalonabilidade. Resume-se diversas abordagens sob diversas restrições. Inicia-se com um modelo simples de tarefas independentes, introduzindo-se gradualmente complicadores como bloqueios, flutuações, deslocamentos, carga adicional, ambientes não-preemptivos e prazos arbitrários de tarefas.

Os testes de escalonabilidade ocupam-se com a possibilidade de provar que determinado conjunto de tarefas com suas restrições é ou não escalonável. Lança-se mão da estratégia de Análise por Taxas Monotônicas (RMA) em que as prioridades são atribuídas sem levar em conta aspectos como importância de tarefas, e sim, uma regra simples: tarefas com períodos menores recebem prioridades maiores. Na Seção 2.6 detalha-se o conceito de prioridade.

Uma variante importante é a estratégia Análise por Prazos Monotônicos (DMA) em que as prioridades das tarefas são atribuídas pela seguinte regra: tarefas com prazos mais próximos recebem prioridades maiores. Pode-se provar, ainda, que se um conjunto de tarefas é escalonável por RMA, também o é por DMA (LEUNG; WHITEHEAD, 1982).

### 3.4.1 Análise com tarefas independentes

A primeira abordagem analítica para estabelecimento de provas de escalonabilidade trata de conjuntos de tarefas simples e básicas com as seguintes restrições:

- tarefas estritamente periódicas;
- tarefas independentes, sem bloqueios ou flutuações;
- prazos iguais aos períodos;
- ambiente preemptivo;
- prioridades fixas atribuídas por RMA.

Na prática é difícil encontrar aplicativos com esta simplicidade, no entanto, as formulações desenvolvidas servem como base e são ampliadas a medida em que algumas restrições são removidas.

Nas próximas seções introduz-se os testes de escalonabilidade para as três abordagens básicas: análise pela utilização do processador, análise pela carga do processador e análise pelo tempo de resposta. Estas abordagens resumem de certa forma as dezenas de testes de escalonabilidade já desenvolvidos (FIDGE, 2002).

#### 3.4.1.1 Análise pela utilização do processador

A taxa de utilização do processador em uma tarefa é seu tempo de computação  $C_i$  dividido pelo seu período  $T_i$ . Se a taxa de utilização for menor ou igual a um, a tarefa é escalonável. Assim, poder-se-ia concluir pela equação 3.1 que um conjunto de tarefas é escalonável se a soma das taxas de utilização de todas as tarefas for menor ou igual a um.

$$\sum_{i=1}^n \frac{C_i}{T_i} \leq 1 \quad (3.1)$$

Isto só é verdade sob certas condições impostas ao conjunto de tarefas: seus serviços devem encaixar-se perfeitamente em todas as combinações de fase entre as

ativações. A TABELA 3.1 e a FIGURA 3.6 mostram um exemplo de tarefas com períodos harmônicos sem considerar prioridades.

TABELA 3.1: Dados para as tarefas da FIGURA 3.6

	Período, $T_i$	Tempo de computação, $C_i$	Prioridade, $P_i$
tarefa 1	6	2	-
tarefa 2	12	3	-
tarefa 3	12	5	-

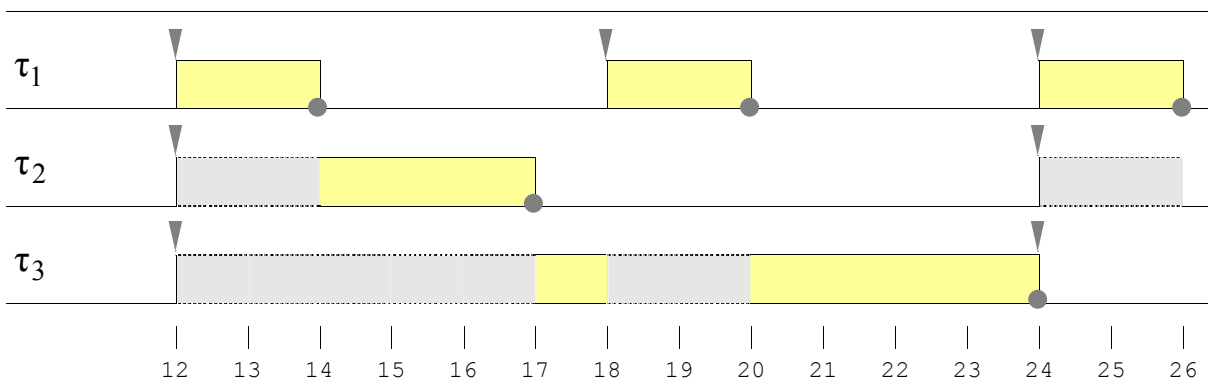


FIGURA 3.6: Exemplo de tarefas escalonáveis por encaixe perfeito

Pela equação 3.1 as  $\tau_1$ ,  $\tau_2$  e  $\tau_3$  seriam escalonáveis, e dado aos períodos harmônicos pode-se observar que de fato o são.

$$\frac{2}{6} + \frac{3}{12} + \frac{5}{12} = 1 \quad (\text{pela equação 3.1})$$

Por exemplo, deslocando-se o instante de ativação da tarefa  $\tau_3$  para à direita, o intervalo de tempo entre as ativações passa a ser utilizado pelas outras tarefas sem modificar a carga do processador no intervalo de tempo do maior períodos. O mesmo raciocínio aplica-se às demais tarefas, constituindo-se uma prova de escalonabilidade.

A TABELA 3.2 e a FIGURA 3.7 mostram um exemplo de tarefas com períodos não harmônicos com prioridades por RMA.

TABELA 3.2: Dados para o exemplo da FIGURA 3.7

	Período, $T_i$	Tempo de computação, $C_i$	Prioridade, $P_i$
tarefa 1	3	1	3
tarefa 2	5	1	2
tarefa 3	11	5	1

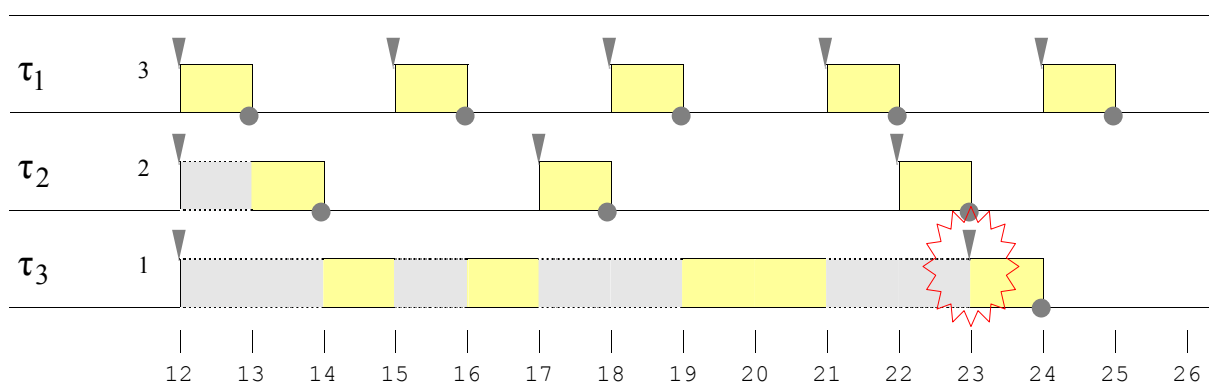


FIGURA 3.7: Exemplo de tarefas não escalonáveis

Pela equação 3.1 as  $\tau_1$ ,  $\tau_2$  e  $\tau_3$  seriam escalonáveis, porém, o próprio exemplo mostra um caso em que uma tarefa ultrapassa seu prazo.

$$\frac{1}{3} + \frac{1}{5} + \frac{5}{11} \leq 1 \quad (\text{pela equação 3.1})$$

Pelos exemplos da TABELA 3.2 e FIGURA 3.6 percebe-se que a falta de harmonia dos períodos degrada a escalonabilidade de tarefas que pela equação 3.1 seriam escalonáveis.

Seja um conjunto de  $n$  tarefas periódicas com prazos iguais aos períodos, prioridades por RMA, em ambiente preemptivo. A equação 3.2 (LUI; LAYLAND,

1973) estabelece um limite máximo  $U(n)$  para a taxa de utilização, que garante a escalonabilidade deste conjunto de tarefas.

$$\sum_{i=1}^n \frac{C_i}{T_i} \leq U(n) = n(2^{1/n} - 1) \quad (3.2)$$

A FIGURA 3.8 mostra o gráfico da função  $U(n)$  com dois pontos interessantes: a) para  $n = 1$  a tarefa tem disponível 100% do tempo do processador; e b) para  $n \rightarrow \infty$  a disponibilidade do processador é de 69,3%. Assim, um aplicativo composto por mais de 20 tarefas com períodos (até perfeitamente) não harmônicos é escalonável se a soma das taxas de utilização for menor que 0,693.

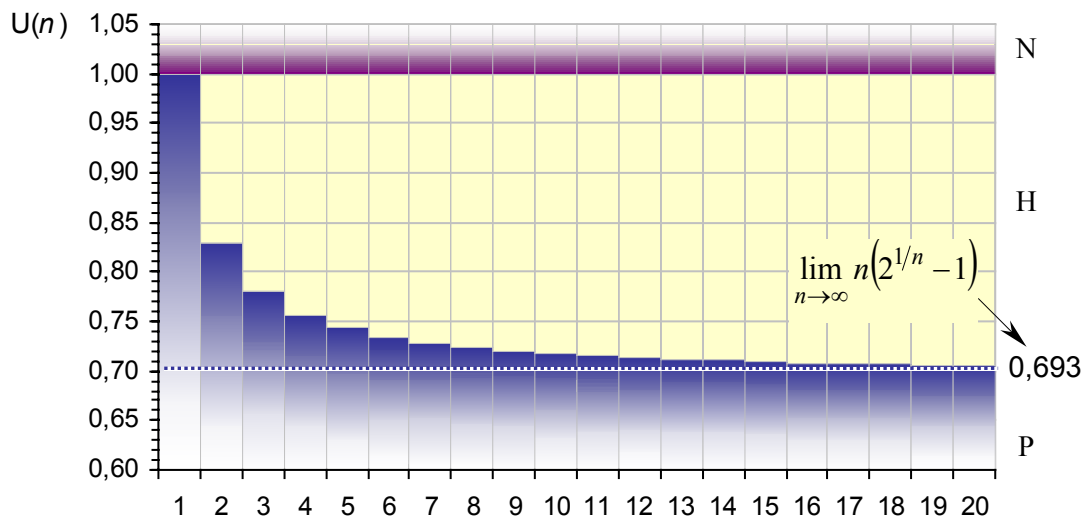


FIGURA 3.8: Variação de  $U(n)$  em função de  $n$

Na FIGURA 3.8 observam-se três regiões em relação ao resultado da equação 3.1: N) a taxa de utilização é maior que 1, e as tarefas não são escalonáveis; P) a taxa de utilização é menor ou igual a  $U(n)$ , e as tarefas sempre são escalonáveis; e H) a taxa de utilização fica entre os limites acima, e nada pode-se afirmar sobre a escalonabilidade das tarefas.

Em geral, as tarefas de um aplicativo apresentam alguma harmonia entre períodos, especialmente devido ao sincronismo entre eventos. Assim, a equação 3.2 torna-se muito pessimista.



O teorema<sup>2</sup> (LEHOCZKY; SHA; DING, 1987) expresso pela equação 3.3 (BRIAND; ROY, 1999) permite verificar a escalonabilidade de um conjunto de tarefas escalonadas por RMA.

$$\left\{ \begin{array}{l} \forall i \in 1..n, \exists (k,l) \in M_i \bullet \sum_{j=1}^i C_j \cdot \left\lceil \frac{l \cdot T_k}{T_j} \right\rceil \leq l \cdot T_k \\ M_i = \left\{ (k,l) \mid k \in 1..i, l \in 1.. \left\lfloor \frac{T_i}{T_k} \right\rfloor \right\} \end{array} \right. \quad (3.3)$$

Na equação acima:

- $n$  é o número de tarefas do conjunto analisado;
- para todo  $m < n$ , as tarefas  $\tau_1$  a  $\tau_m$  têm prioridade maior que a tarefa  $\tau_{m+1}$ ;
- $i$  determina o subconjunto de tarefas  $\tau_1$  a  $\tau_i$  em análise;
- $k$  identifica a tarefa  $\tau_k$  cujos  $l$  períodos são analisados;
- $M_i$  é o conjunto de duplas  $(k,l)$  referentes à tarefa  $\tau_i$  em que  $k$  varia de 1 a  $i$ , e  $l$  de 1 ao número de períodos  $T_k$  ocorridos no período  $T_i$ , sempre maior ou igual a 1;
- o termo da soma é o tempo gasto pela tarefa  $\tau_j$  em  $l$  períodos da tarefa  $\tau_k$ ;
- se existe alguma dupla  $(k,l)$  para a qual o total da soma seja menor ou igual a  $l$  vezes o período da tarefa  $\tau_k$ , o subconjunto de tarefas em análise é escalonável;
- se o teste for bem sucedido para os  $n$  subconjuntos de tarefas para análise, o conjunto todo de tarefas é escalonável.

A equação 3.3 oferece uma prova matemática da escalonabilidade de um conjunto de tarefas com maior grau de otimismo que a equação 3.2.

---

<sup>2</sup> Baseado no teorema da REGIÃO CRÍTICA DE TEMPO de Liu e Layland (LIU; LAYLAND, 1973): “SE UM CONJUNTO DE TAREFAS PERIÓDICAS E INDEPENDENTES É ATIVADO NO MESMO INSTANTE, E CADA TAREFA ATENDE A SEU PRIMEIRO PRAZO, ENTÃO TODOS OS FUTUROS PRAZOS SERÃO ATENDIDOS”. O teorema não condiciona o conjunto de tarefas à RMA, sendo aplicável a qualquer esquema de prioridades estáticas.

### 3.4.1.2 Análise pela carga do processador

A análise pela carga do processador calcula a carga em determinado instante de tempo, considerando alguma tarefa como referência (LEHOCZKY; SHA; DING, 1987).

Pela equação 3.4, a carga do processador  $W_i(t)$  é dada pela soma dos tempos de computação do instante 0 ao instante  $t$  das diversas ativações das tarefas com prioridade maior ou igual à tarefa  $\tau_i$ ,  $j \in \text{hep}(i)$ . O fator  $\lceil t/T_j \rceil$  determina o número de vezes que o período  $T_j$  se sobrepõe ao intervalo de tempo  $t$  que multiplicado por  $C_j$  equivale ao tempo gasto pela tarefa  $\tau_j$ .

$$W_i(t) = \sum_{j \in \text{hep}(i)} \left\lceil \frac{t}{T_j} \right\rceil \cdot C_j \quad (3.4)$$

A equação 3.5 estabelece que a tarefa  $\tau_i$  é escalonável sempre que em algum instante de tempo  $t$  no período  $T_i$  a carga do processador seja menor ou igual a 1.

$$\min_{0 < t \leq T_i} \frac{W_i(t)}{t} \leq 1 \quad (3.5)$$

Fundamentado no teorema da equação 3.3, se dada tarefa passa pelo teste da equação 3.5, esta é escalonável. Do ponto de vista computacional, a aplicação das equações 3.4 e 3.5 no intervalo  $0 < t \leq T_i$  tem custo alto, no entanto é necessário apenas examinar os instantes  $t$  de ativação das tarefas do conjunto  $\text{hep}(i)$  (LEHOCZKY; SHA; DING, 1987).

Para verificar se a tarefa  $\tau_3$  que tem a menor prioridade da TABELA 3.2 acima é escalonável, examina-se os instantes 3, 5, 6, 9, 10 e 11 das ativações de todas as tarefas. Assim,

$$\left( \left\lceil \frac{3}{3} \right\rceil \cdot 1 + \left\lceil \frac{3}{5} \right\rceil \cdot 1 + \left\lceil \frac{3}{11} \right\rceil \cdot 5 \right) \cdot \frac{1}{3} = \frac{7}{3} \quad (\text{instante 3})$$

$$\left( \left\lceil \frac{5}{3} \right\rceil \cdot 1 + \left\lceil \frac{5}{5} \right\rceil \cdot 1 + \left\lceil \frac{5}{11} \right\rceil \cdot 5 \right) \cdot \frac{1}{5} = \frac{8}{5} \quad (\text{instante 5})$$

$$\left( \left\lceil \frac{6}{3} \right\rceil \cdot 1 + \left\lceil \frac{6}{5} \right\rceil \cdot 1 + \left\lceil \frac{6}{11} \right\rceil \cdot 5 \right) \cdot \frac{1}{6} = \frac{9}{6} \quad (\text{instante 6})$$

$$\left( \left\lceil \frac{9}{3} \right\rceil \cdot 1 + \left\lceil \frac{9}{5} \right\rceil \cdot 1 + \left\lceil \frac{9}{11} \right\rceil \cdot 5 \right) \cdot \frac{1}{9} = \frac{10}{9} \quad (\text{instante 9})$$

$$\left( \left\lceil \frac{10}{3} \right\rceil \cdot 1 + \left\lceil \frac{10}{5} \right\rceil \cdot 1 + \left\lceil \frac{10}{11} \right\rceil \cdot 5 \right) \cdot \frac{1}{10} = \frac{11}{10} \quad (\text{instante 10})$$

$$\left( \left\lceil \frac{11}{3} \right\rceil \cdot 1 + \left\lceil \frac{11}{5} \right\rceil \cdot 1 + \left\lceil \frac{11}{11} \right\rceil \cdot 5 \right) \cdot \frac{1}{11} = \frac{12}{11} \quad (\text{instante 11})$$

O menor dos valores obtidos é 12/11 que é maior que 1 e confirma a conclusão a que se chegou pela FIGURA 3.7: a tarefa não é escalonável.

### 3.4.1.3 Análise pelo tempo de resposta

A análise pelo tempo de resposta consiste em determinar a interferência a que uma tarefa está sujeita devida às tarefas com prioridade maior que a sua. Obviamente, a maior interferência entre tarefas ocorre na região crítica de tempo.

Pela equação 3.6 o tempo de resposta é igual ao tempo de computação da tarefa  $\tau_i$  somado à interferência sofrida.

$$R_i = C_i + I_i \quad (3.6)$$

A equação 3.7 determina a interferência  $I_i$  pela soma dos tempos de computação das tarefas com prioridade maior que a tarefa  $i$ ,  $j \in \text{hp}(i)$ , ao longo do tempo de resposta da tarefa em questão  $R_i$ . O fator  $\lceil R_i/T_j \rceil$  determina o número de vezes que o período  $T_j$  se sobrepõe ao período  $R_i$ , e que multiplicado por  $C_j$  equivale ao tempo de interferência da tarefa  $\tau_j$ .

$$I_i = \sum_{j \in \text{hp}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j \quad (3.7)$$

Combinando as equações 3.6 e 3.7, obtém-se a equação 3.8, que determina o pior tempo de resposta para um conjunto de tarefas independentes com prioridade fixa (não necessariamente atribuídas por RMA) num ambiente preemptivo.

$$R_i = C_i + \sum_{j \in \text{hp}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j \quad (3.8)$$

Embora seja exata, a equação 3.8 não pode ser resolvida algebricamente. É necessário lançar mão do algoritmo recorrente definido pela equação 3.9 (AUDSLEY *et alii*, 1993).

$$R_i^{(v+1)} = C_i + \sum_{j \in \text{hp}(i)} \left\lceil \frac{R_i^{(v)}}{T_j} \right\rceil \cdot C_j \quad (3.9)$$

As sucessivas versões de  $R_i^{(v)}$ , para  $v \in 1..k$ , iniciando com  $R_i^{(0)} = 0$ , crescem de forma monotônica eventualmente convergindo para  $R_i^{(n)} = R_i^{(n+1)}$ . Se em alguma versão o tempo de resposta for maior que o prazo  $R_i^{(v)} > D_i = T_i$ , conclui-se que a tarefa não é escalonável.

### 3.4.2 Análise com bloqueios

Bloqueio é um tipo de interferência a que uma tarefa está sujeita devido às tarefas com prioridade menor que a sua. Por exemplo, uma tarefa de prioridade menor detém a trava de um recurso compartilhado, em seguida outra tarefa de prioridade maior requer a trava do mesmo recurso que por consequência deve esperar até a liberação deste recurso. Esta espera é o tempo de bloqueio  $B_i$ . Bloqueios podem, ainda, ser causados pela sincronização e comunicação entre tarefas. Remove-se assim a restrição de independência das tarefas.

Para tratar adequadamente dos tempos de bloqueio, as equações da seção anterior devem ser modificadas. A seguir apresenta-se as equações da análise pela utilização do processador e pelo tempo de resposta das tarefas.

A equação 3.10 modifica a equação 3.2, permitindo verificar a escalonabilidade de cada subconjunto de tarefas  $\{\tau_1 \dots \tau_i\}$ , para  $i$  de 1 a  $n$  isoladamente. Consideram-se as mesmas restrições sobre o conjunto de tarefas.

$$\forall i \in 1..n \cdot \sum_{j=1}^i \frac{C_j}{T_j} + \frac{B_i}{T_i} \leq U(i) = i(2^{1/i} - 1) \quad (3.10)$$

A equação 3.11 resume a equação 3.10, possibilitando o cálculo de escalonabilidade do conjunto de tarefas num só lance. A equação 3.11 é mais pessimista que a equação 3.10 (BRIAND; ROY, 1999).

$$\sum_{i=1}^n \frac{C_i}{T_i} + \max_{i=1}^{n-1} \frac{B_i}{T_i} \leq U(n) = n(2^{1/n} - 1) \quad (3.11)$$

A equação 3.12 modifica a equação 3.4, considerando as mesmas restrições sobre o conjunto de tarefas (BRIAND; ROY, 1999).

$$\left\{ \begin{array}{l} \forall i \in 1..n, \exists (k, l) \in M_i \cdot \sum_{j=1}^{i-1} C_j \cdot \left\lceil \frac{l \cdot T_k}{T_j} \right\rceil + C_i + B_i \leq l \cdot T_k \\ M_i = \left\{ (k, l) \mid k \in 1..i, l \in 1.. \left\lceil \frac{T_i}{T_k} \right\rceil \right\} \end{array} \right. \quad (3.12)$$

A equação 3.13 modifica a equação 3.8, considerando as mesmas restrições sobre o conjunto de tarefas (AUDSLEY *et alii*, 1993). A solução segue raciocínio semelhante ao da equação 3.9.

$$R_i = C_i + B_i + \sum_{j \in \text{hp}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil \cdot C_j \quad (3.13)$$

### 3.4.2.1 Valoração do tempo de bloqueio

Tomando o quadro de tarefas da FIGURA 3.9 como exemplo, observa-se que a tarefa  $\tau_4$  obtém a trava do recurso  $S_1$  no instante 13 e em seguida é preemptada por diversas vezes pelas tarefas de prioridade maior. No instante 17, a tarefa  $\tau_1$  solicita a trava de  $S_1$ , e não conseguindo entra no estado (SUSPENSO)/BLOQUEADO até que o recurso seja liberado no instante 23. No caso, o tempo de bloqueio  $B_i$  é igual a 6 unidades. Contudo, este tempo é determinado basicamente pela interferência das tarefas  $\tau_2$  e  $\tau_3$  no tempo de resposta da tarefa  $\tau_4$ . Tal tratamento de bloqueios leva a impasses de bloqueio e inversões de prioridade, além do que torna difícil ou mesmo impossível determinar o tempo de bloqueio limite para o pior caso (BUTTAZZO, 1997).

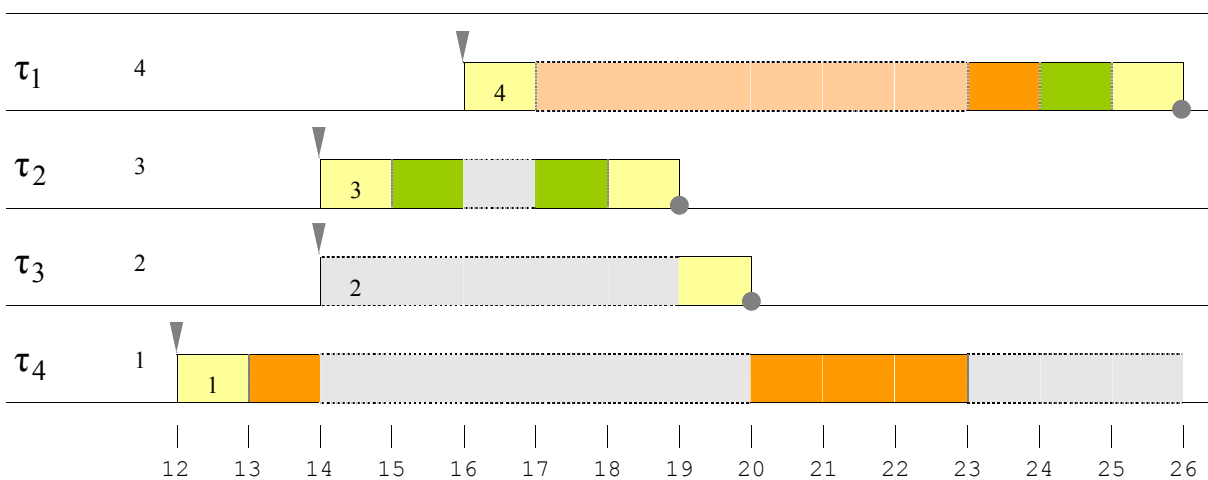


FIGURA 3.9: Exemplo de tarefas sem herança de prioridade

Diversos protocolos de travamento de recursos foram propostos no sentido de evitar a inversão de prioridade e possibilitar a determinação do tempo de bloqueio. Uma solução simples mas ineficiente é inibir a preempção durante as seções críticas (BUTTAZZO 1997).

A seguir serão apresentados os três protocolos mais comuns: Protocolo de Herança de Prioridade (PIP), Protocolo de Teto de Prioridade (PCP) e Protocolo de

Trava Máxima (HL)<sup>3</sup> (BRIAND; ROY, 1999). A valoração do tempo de bloqueio  $B_i$  depende do protocolo em questão.

### 3.4.2.2 Protocolo de Herança de Prioridade

Sob o PROTOCOLO DE HERANÇA DE PRIORIDADE<sup>4</sup> (PIP), a tarefa possui uma prioridade básica atribuída estaticamente e uma prioridade ativa que: a) fora de seções críticas é igual à sua prioridade básica; e b) durante seções críticas é adquirida dinamicamente, evitando inversão de prioridade.

A prioridade ativa de uma tarefa durante a seção crítica que detém a trava de um recurso é igual ao máximo entre sua prioridade básica e a maior prioridade ativa do conjunto de tarefas bloqueadas que esperam pela liberação deste recurso. Num dado intervalo de tempo a tarefa pode estar em mais de uma seção crítica simultaneamente, e neste caso a prioridade ativa é igual à maior das prioridades ativas destas seções críticas. A tarefa obtém a trava de um recurso sempre que esta estiver disponível.

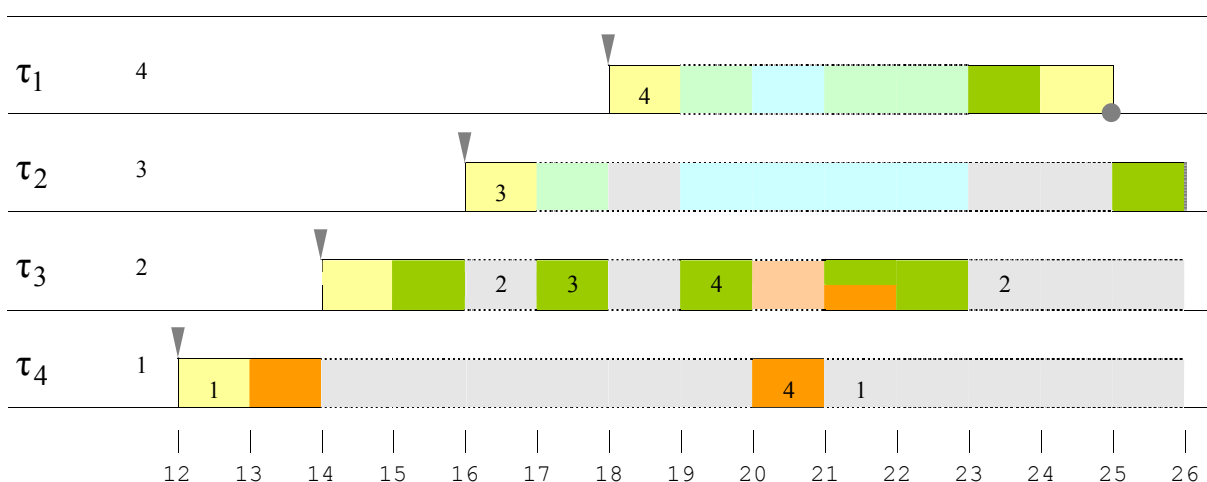


FIGURA 3.10: Situações sob o Protocolo de Herança de Prioridade

<sup>3</sup> Ambientes como Ada e Java implementam diversas políticas de escalonamento, ativadas através de artifícios de linguagem de programação, por exemplo, `pragma Priority (...)`. O padrão de indústria POSIX 1003.1a,1b,1c,1d regulamenta um conjunto mínimo de políticas disponibilizadas em sistemas operacionais (BRIAND; ROY, 1999).

<sup>4</sup> Definido pelo protocolo POSIX: `PRIO_INHERIT`.

A FIGURA 3.10 exemplifica o comportamento de um conjunto de tarefas sob o Protocolo de Herança de Prioridade. A seguir discute-se algumas situações importantes para compreender o Protocolo de Herança de Prioridade:

- no intervalo  $[13,20)$  a tarefa  $\tau_4$  detém a trava do recurso  $S_1$  sem que esta seja requisitada por outra tarefa; sua prioridade ativa é igual à sua prioridade básica;
- no intervalo  $[15,17)$  a tarefa  $\tau_3$  detém a trava do recurso  $S_2$  sem que esta seja requisitada por outra tarefa; sua prioridade ativa é igual à sua prioridade básica;
- no instante 17 a tarefa  $\tau_2$  requisita a trava do recurso  $S_2$ , ficando bloqueada; a prioridade ativa da tarefa  $\tau_3$ , que detém esta trava, passa para o máximo das prioridades ativas das tarefas  $\tau_2$  e  $\tau_3$ , envolvidas com o recurso  $S_2$ ;
- no instante 19 a tarefa  $\tau_1$  requisita a trava do recurso  $S_2$ , ficando bloqueada; a prioridade ativa da tarefa  $\tau_3$ , que detém esta trava, passa para o máximo das prioridades ativas das tarefas  $\tau_1$ ,  $\tau_2$  e  $\tau_3$ , envolvidas com o recurso  $S_2$ ;
- no instante 20 a tarefa  $\tau_3$  requisita a trava do recurso  $S_1$ , ficando bloqueada; a prioridade ativa da tarefa  $\tau_4$ , que detém esta trava, passa para o máximo das prioridades ativas das tarefas  $\tau_3$  e  $\tau_4$ , envolvidas com o recurso  $S_1$ ;
- no instante 21 a tarefa  $\tau_4$  libera o recurso  $S_1$ ; sua prioridade ativa volta para sua prioridade básica; a tarefa  $\tau_3$ , que está bloqueada pelo recurso  $S_1$ , obtém a trava deste recurso e prossegue, agora com a trava de dois recursos;
- no instante 23 a tarefa  $\tau_3$  libera o recurso  $S_2$ ; sua prioridade ativa volta para sua prioridade básica; a tarefa  $\tau_1$ , que está esperando pelo recurso  $S_2$ , obtém a trava deste recurso e prossegue.

No intervalo  $[20,21)$  observa-se o fenômeno da cadeia de bloqueios onde a tarefa  $\tau_4$  bloqueia a tarefa  $\tau_3$ , que bloqueia a tarefa  $\tau_1$  e indiretamente a tarefa  $\tau_2$ . Esta cadeia de bloqueios pode ser circular, gerando um impasse de bloqueio.

A FIGURA 3.11 reedita a situação das tarefas da FIGURA 3.9 sob o Protocolo de Herança de Prioridade. Percebe-se claramente a mudança do comportamento da tarefa



$\tau_1$  em relação às tarefas de prioridades intermediárias. A tarefa  $\tau_3$  também sofre de bloqueio apesar de não usar os recursos.

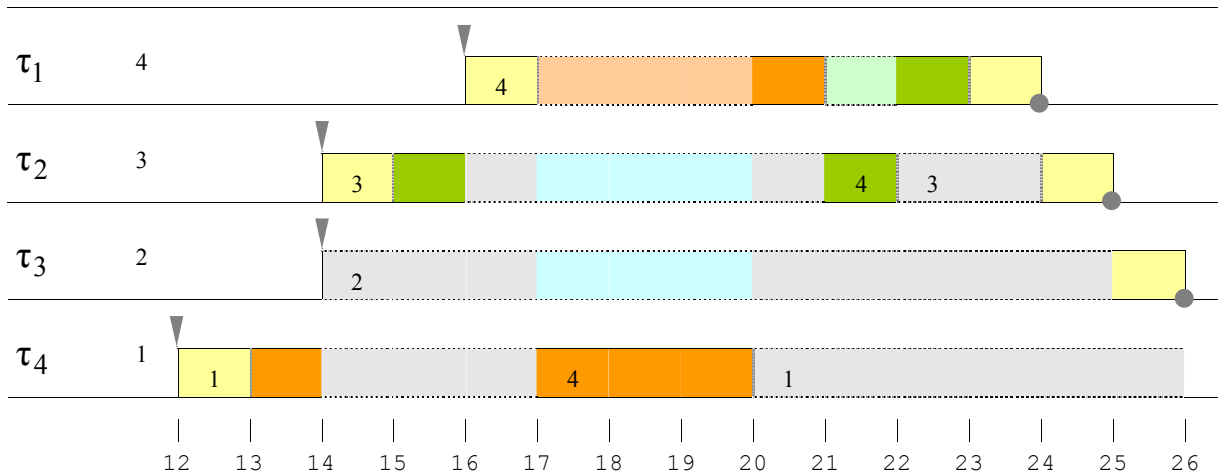


FIGURA 3.11: Exemplo de Protocolo de Herança de Prioridade

A equação 3.14 determina de forma não otimizada o pior caso de tempo de bloqueio que uma tarefa pode sofrer sob o Protocolo de Herança de Prioridade, baseada na equação 3.15 e na equação apresentada em (BURNS; WELLINGS, 2001).

$$B_i = \sum_{\substack{k \in lp(i) \\ S \in locks(k,i) \\ m \in csc(k,S)}} (C_{k,S})_m \quad (3.14)$$

A função  $locks(k,i)$  retorna o conjunto de recursos utilizados pela tarefa  $k$  e também pelas tarefas com prioridade básica maior ou igual à prioridade básica da tarefa  $i$ . A função  $csc(k,S)$  retorna o conjunto de índices  $m$  para cada seção crítica do recurso  $S$  contido na tarefa  $k$ . A função  $lp(i)$  retorna o conjunto de tarefas com prioridade básica menor que a prioridade da tarefa  $i$ . Pela FIGURA 3.12, para  $k = 4$ ,  $S = S_1$  e  $m = 3$ ,  $(C_{k,S})_m$  é igual a 1.

Por exemplo, aplicando a equação 3.14 sobre os dados da FIGURA 3.12 obtém-se a TABELA 3.3, em que para cada combinação  $(i,k)$  acessível  $k \in lp(i)$  têm-se no lado esquerdo das células os conjuntos  $locks(k,i)$  e no lado direito o conjunto de tempos  $(C_{k,S})_m$  das seções críticas. A coluna  $B_i$  é a soma dos tempos da linha correspondente.

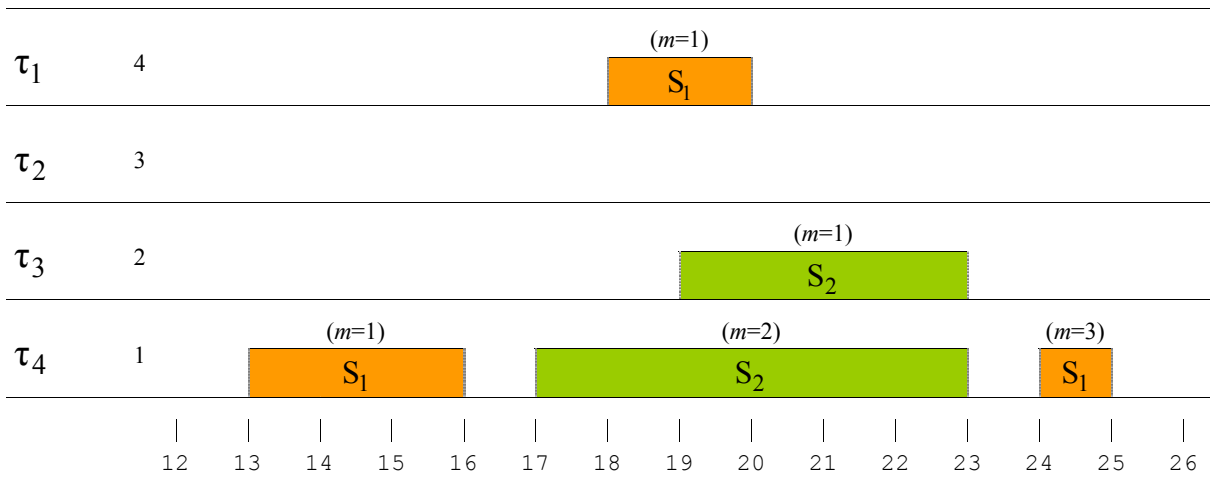


FIGURA 3.12: Distribuição de seções críticas por tarefa

TABELA 3.3: Tempo de bloqueio sob o Protocolo de Herança de Prioridade

$i \ k$	1	2	3	4	$B_i$
1		$\emptyset$	$\emptyset$	$\{S_1\}$   $\{3,1\}$	4
2			$\emptyset$	$\{S_1\}$   $\{3,1\}$	4
3				$\{S_1, S_2\}$   $\{3,1,6\}$	10
4					0

A FIGURA 3.13 mostra dois casos em que seções críticas são combinadas e dificultam a atribuição de valores: no caso da FIGURA 3.13A as seções críticas estão aninhadas, e as tarefas dependentes de  $S_1$  devem enxergar apenas o tempo referente a  $S_1$ ; no caso da FIGURA 3.13B as seções críticas estão sobrepostas, e as tarefas dependentes de  $S_1$  e  $S_2$  simultaneamente devem enxergar a soma dos tempos subtraída do tempo de sobreposição.

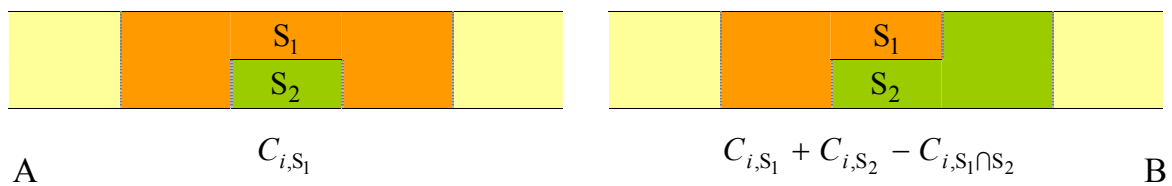


FIGURA 3.13: Combinações de seções críticas

### 3.4.2.3 Protocolo de Teto de Prioridade

O Protocolo de Herança de Prioridade não previne impasses de bloqueio, além disso pode levar a tempos de bloqueio pessimistas demais devidos à formação de cadeias de bloqueio como mostra o instante 20 da FIGURA 3.10 (BURNS; WELLINGS, 2001). O PROTOCOLO DE TETO DE PRIORIDADE (PCP, OCPP) resolve estes problemas pelas seguintes características:

- tarefas podem ser bloqueadas uma vez por tarefas de prioridade mais baixa;
- previne impasses de bloqueio e cadeias de bloqueios;
- garante acesso mútuo exclusivo a recursos intrinsecamente.

Sob o Protocolo de Teto de Prioridade, o recurso tem seu teto de prioridade igual à maior prioridade básica das tarefas em que é utilizado. O aplicativo possui a variável teto geral, que guarda a maior prioridade dos recursos travados. A prioridade das tarefas é definida da mesma forma como no PIP. A tarefa obtém a trava de um recurso livre se sua prioridade ativa for maior que o teto geral ou se ela detém a trava do recurso que fixou o teto geral.

A FIGURA 3.14 reedita a situação das tarefas da FIGURA 3.11 sob o Protocolo de Teto de Prioridade. Novamente a tarefa  $\tau_1$  termina mais cedo.

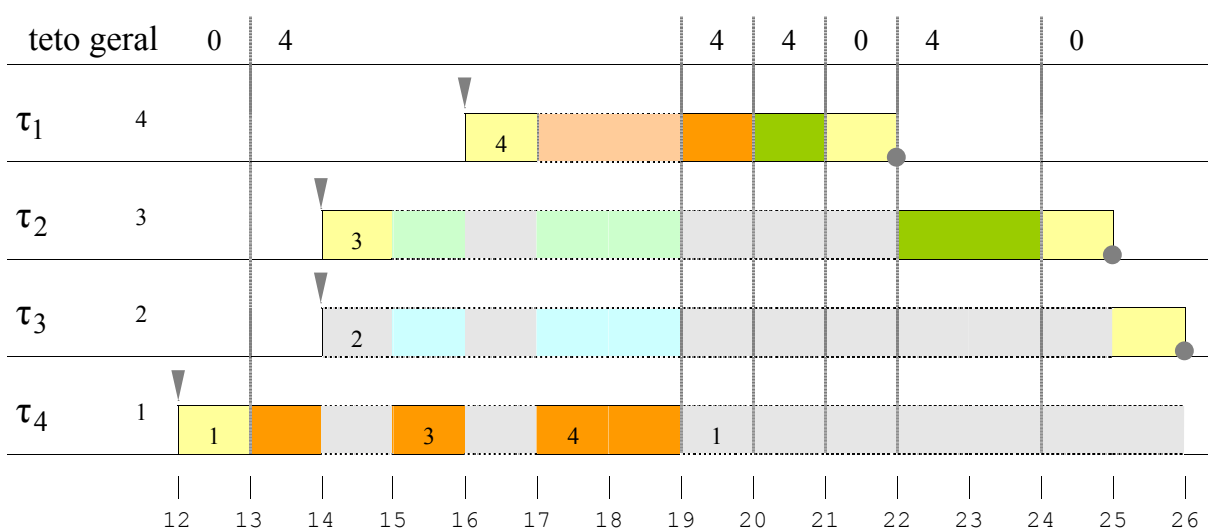


FIGURA 3.14: Exemplo de Protocolo de Teto de Prioridade

A FIGURA 3.14 exemplifica o comportamento de um conjunto de tarefas. A seguir discute-se algumas situações importantes para compreender o Protocolo de Teto de Prioridade:

- o recurso  $S_1$  tem teto de prioridade igual a 4 (utilizada nas tarefas  $\tau_1$  e  $\tau_4$ );
- o recurso  $S_2$  tem teto de prioridade igual a 4 (utilizada nas tarefas  $\tau_1$  e  $\tau_2$ );
- no instante 13 a tarefa  $\tau_4$  obtém a trava do recurso  $S_1$ ; o teto geral passa para o teto de prioridade do recurso  $S_1$ ;
- no instante 15 a tarefa  $\tau_2$  requisita a trava do recurso  $S_2$ ; não consegue, pois, sua prioridade ativa não é maior que o teto geral; a prioridade ativa da tarefa  $\tau_4$  passa para o máximo das prioridades ativas das tarefas  $\tau_2$  e  $\tau_4$ , envolvidas com o recurso  $S_2$ ;
- no instante 17 a tarefa  $\tau_1$  requisita a trava do recurso  $S_1$ ; não consegue, pois, sua prioridade ativa não é maior que o teto geral; a prioridade ativa da tarefa  $\tau_4$  passa para o máximo das prioridades ativas das tarefas  $\tau_1$  e  $\tau_4$ , envolvidas com o recurso  $S_1$ ;
- no instante 19 a tarefa  $\tau_1$  libera o recurso  $S_1$ ; sua prioridade ativa volta para sua prioridade básica; o teto geral passa para 0;
- no instante 19 a tarefa  $\tau_1$  obtém a trava do recurso  $S_1$ ; o teto geral passa para o teto de prioridade do recurso  $S_1$ ;
- no instante 20 a tarefa  $\tau_1$  libera o recurso  $S_1$ ; sua prioridade ativa volta para sua prioridade básica; o teto geral passa para 0;
- no instante 20 a tarefa  $\tau_1$  obtém a trava do recurso  $S_2$ ; o teto geral passa para o teto de prioridade do recurso  $S_2$ ;
- no instante 21 a tarefa  $\tau_1$  libera o recurso  $S_2$ ; o teto geral passa para 0.

O que acontece nos instantes 15 e 17 previne impasses de bloqueio, simplesmente não permitindo que outros recursos com teto de prioridade menor ou igual ao teto geral sejam travados.

A equação 3.15 determina o pior caso de tempo de bloqueio que uma tarefa pode sofrer sob o Protocolo de Teto de Prioridade (TINDELL, 2000).

$$B_i = \max_{\substack{k \in lp(i) \\ S \in locks(k,i) \\ m \in csc(k,S)}} (C_{k,S})_m \quad (3.15)$$

Por exemplo, aplicando a equação 3.15 sobre os dados da FIGURA 3.12 obtém-se a TABELA 3.4, em que para cada combinação  $(i,k)$  acessível  $k \in lp(i)$  tem-se no lado esquerdo das células os conjuntos  $locks(k,i)$  e no lado direito o conjunto de tempos  $(C_{k,S})_m$  das seções críticas. A coluna  $B_i$  é o máximo dos tempos da linha correspondente.

TABELA 3.4: Tempos de bloqueio sob o Protocolo de Teto de Prioridade

$i \ k$	1	2	3	4	$B_i$
1		$\emptyset$	$\emptyset$	$\{S_1\}$ $\{3,1\}$	3
2			$\emptyset$	$\{S_1\}$ $\{3,1\}$	3
3				$\{S_1, S_2\}$ $\{3,1,6\}$	6
4					0

#### 3.4.2.4 Protocolo de Trava Máxima

O PROTOCOLO DE TRAVA MÁXIMA<sup>5</sup> (HL, ICPP) é uma pequena variação do PCP. O recurso tem seu teto de prioridade igual à maior prioridade básica das tarefas em que é utilizado. A tarefa tem prioridade básica e ativa como no PIP, porém, sua prioridade ativa é igual ao maior teto de prioridade do conjunto de recursos que detém, mais um. A tarefa sempre obtém a trava de um recurso livre.

A FIGURA 3.15 reedita a situação das tarefas da FIGURA 3.14 sob o Protocolo de Trava Máxima. Novamente a tarefa  $\tau_1$  termina mais cedo.

<sup>5</sup> Definido pelo protocolo POSIX: PRIO\_PROTECT (BRIAND; ROY, 1999).

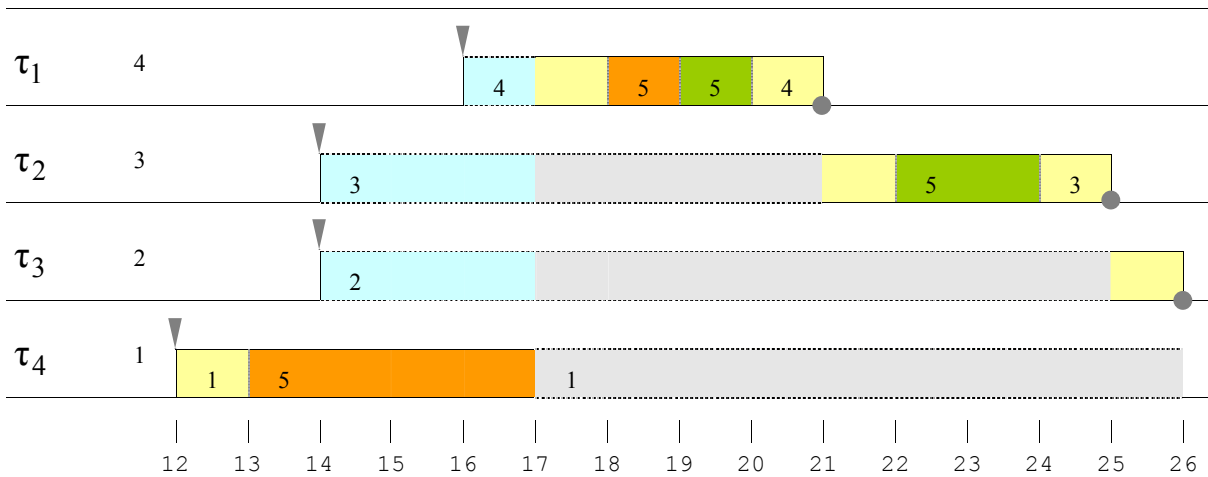


FIGURA 3.15: Exemplo de Protocolo de Trava Máxima

A FIGURA 3.15 exemplifica o comportamento de um conjunto de tarefas. A seguir discute-se algumas situações importantes para compreender o Protocolo de Trava Máxima:

- o recurso  $S_1$  tem teto de prioridade igual a 4 (utilizada nas tarefas  $\tau_1$  e  $\tau_4$ );
- o recurso  $S_2$  tem teto de prioridade igual a 4 (utilizada nas tarefas  $\tau_3$  e  $\tau_2$ );
- no instante 13 a tarefa  $\tau_4$  obtém a trava do recurso  $S_1$ ; imediatamente sua prioridade ativa passa para o teto de prioridade do recurso  $S_1$  mais um;
- no intervalo [14,16] as tarefas  $\tau_1$ ,  $\tau_2$  e  $\tau_3$  são ativadas; não conseguem o controle do processador, uma vez que suas prioridades ativas são menores que a prioridade ativa da tarefa  $\tau_4$ ; desta forma não haverá tentativa de utilizar os recursos  $S_1$  e  $S_2$ ;
- no instante 17 a tarefa  $\tau_4$  libera o recurso  $S_1$ ; sua prioridade ativa volta para sua prioridade básica.

A equação 3.15 também determina o pior caso de tempo de bloqueio que uma tarefa pode sofrer sob o Protocolo de Trava Máxima (TINDELL, 2000).

### 3.4.3 Análise não-preemptiva

A análise de escalonabilidade em ambiente não-preemptivo toma como base a equação 3.13, considerando os tempos de interferência e os tempos de bloqueio em torno do único recurso compartilhado, o processador.

A equação 3.16 define o tempo de resposta como o tempo de computação acrescido do tempo de atraso na liberação.

$$R_i = a_i + C_i \quad (3.16)$$

A equação 3.18 define o atraso de liberação da tarefa composto pela soma de dois termos: a) o tempo de bloqueio  $B_i$  (equação 3.17) igual ao maior tempo de computação das tarefas com prioridade menor; e b) a interferência causada pelas tarefas com prioridade maior (BURNS; WELLINGS, 2001). A solução segue raciocínio semelhante ao da equação 3.9.

$$B_i = \max_{k \in lp(i)} C_k \quad (3.17)$$

$$a_i = B_i + \sum_{j \in hp(i)} \left( \left\lfloor \frac{a_i}{T_j} \right\rfloor + 1 \right) \cdot C_j \quad (3.18)$$

A TABELA 3.5 e a FIGURA 3.16 exemplificam o escalonamento de tarefas no ambiente não-preemptivo. A seguir discute-se algumas situações importantes para compreender a análise em ambiente não-preemptivo:

- no instante 13 a tarefa  $\tau_3$  é ativada e liberada; em seguida as tarefas  $\tau_1$  e  $\tau_2$  são ativadas, mas mesmo com prioridades maiores não podem ser liberadas antes do término da tarefa  $\tau_3$ ;
- no instante 15 obviamente a tarefa  $\tau_1$  é liberada antes da tarefa  $\tau_2$  pela ordem de prioridade;
- no instante 17 a tarefa  $\tau_1$  é ativada e, mesmo com prioridade maior que a tarefa  $\tau_2$ , deve aguardar o término desta.

TABELA 3.5: Dados para as tarefas da FIGURA 3.16

	Período $T_i$	Tempo de computação $C_i$	Prioridade $P_i$	Tempo de bloqueio $B_i$	Tempo de resposta $a_i + C_i = R_i$
tarefa 1	4	1	3	2	$2+1 = 3$
tarefa 2	6	2	2	2	$3+2 = 5$
tarefa 3	7	2	1	0	$3+2 = 5$

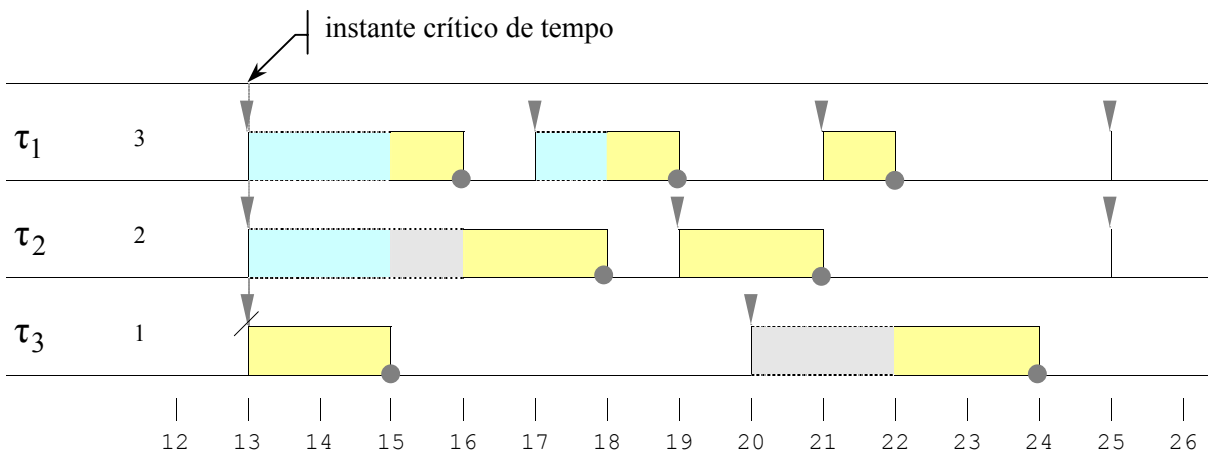


FIGURA 3.16: Exemplo de tarefas em ambiente não-preemptivo

### 3.4.4 Flutuação de liberação

As tarefas que apresentam flutuação de liberação não se encaixam perfeitamente no esquema periódico. Considerando os dados da TABELA 3.6 sem a flutuação de liberação, a tarefa  $\tau_2$  seria escalonável com tempo de resposta  $R_2 = 9$ .

TABELA 3.6: Dados para as tarefas da FIGURA 3.17

	Período $T_i$	Tempo de computação $C_i$	Flutuação $J_i$	Prioridade $P_i$	Tempo de resposta $r_i + J_i = R_i$
tarefa 1	9	2	2	2	$2+2 = 4$
tarefa 2	11	7	1	1	$11+1 = 12$



O flagrante da FIGURA 3.17 com flutuação de liberação mostra que a tarefa  $\tau_2$  não é escalonável, uma vez que perde seu prazo no instante 25.

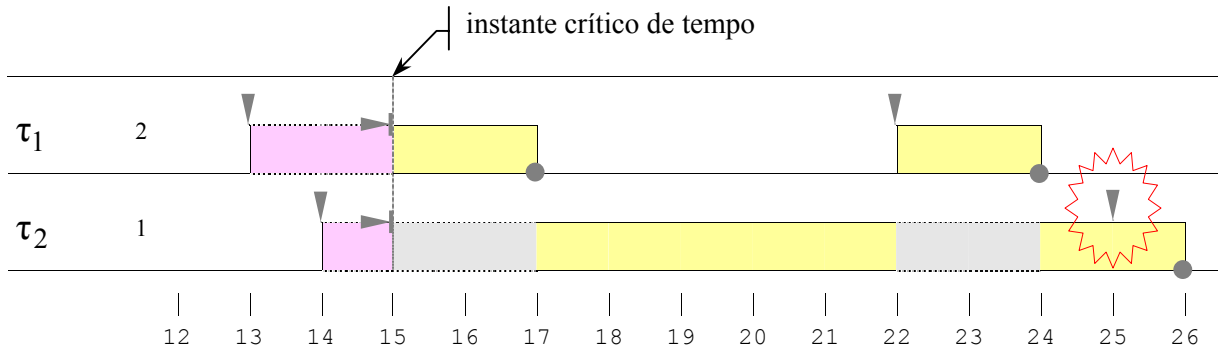


FIGURA 3.17: Exemplo de tarefas com flutuação de liberação

A equação 3.19 modifica a equação 3.13, considerando a interferência da flutuação de liberação. Primeiro determina-se o tempo de resposta  $r_i$  em relação ao instante de liberação da tarefa. O termo  $r_i + J_j$  acresce a flutuação da tarefa  $\tau_j$  para capturar as possíveis interferências no intervalo  $J_j$ . A solução segue raciocínio semelhante ao da equação 3.9.

$$r_i = C_i + B_i + \sum_{j \in \text{hp}(i)} \left\lceil \frac{r_i + J_j}{T_j} \right\rceil \cdot C_j \quad (3.19)$$

A equação 3.20 acrescenta a flutuação de liberação da própria tarefa ao seu tempo de resposta em relação à liberação e determina o tempo de resposta em relação à ativação.

$$R_i = r_i + J_i \quad (3.20)$$

A região crítica de tempo para tarefas com flutuação de liberação inicia no instante da liberação das tarefas e não no instante de ativação.

### 3.4.5 Deslocamento

O deslocamento representa um intervalo de tempo fixo entre a ativação e a liberação de uma tarefa. Portanto, a determinação do tempo de resposta na presença de deslocamentos pode utilizar o resultado da equação 3.20 para  $R_i^{(J)}$  acrescido do deslocamento como mostra a equação 3.21.

$$R_i = R_i^{(J)} + O_i \quad (3.21)$$

A princípio o tempo de resposta com deslocamento torna-se muito pessimista, principalmente porque as equações 3.8, 3.13 e 3.20 levam em consideração a região crítica de tempo. O deslocamento objetiva justamente introduzir uma diferença de fase entre as ativações de duas ou mais tarefas.

A TABELA 3.7 e a FIGURA 3.18 mostram um exemplo de duas tarefas que se comunicam. A tarefa  $\tau_2$  deve ser liberada somente quando terminar o prazo da tarefa  $\tau_1$  como garantia da disponibilidade dos dados comunicados. O deslocamento é responsável pela diferença de fase na liberação das duas tarefas.

TABELA 3.7: Dados para as tarefas da FIGURA 3.18

	Período $T_i$	Tempo de computação $C_i$	Desloca- mento $O_i$	Prioridade $P_i$	Prazo $D_i$	Tempo de resposta $R_i^{(J)} + O_i = R_i$
tarefa 1	6	2	0	2	3	$2+0 = 2$
tarefa 2	6	1	3	1	5	$3+3 = 6$

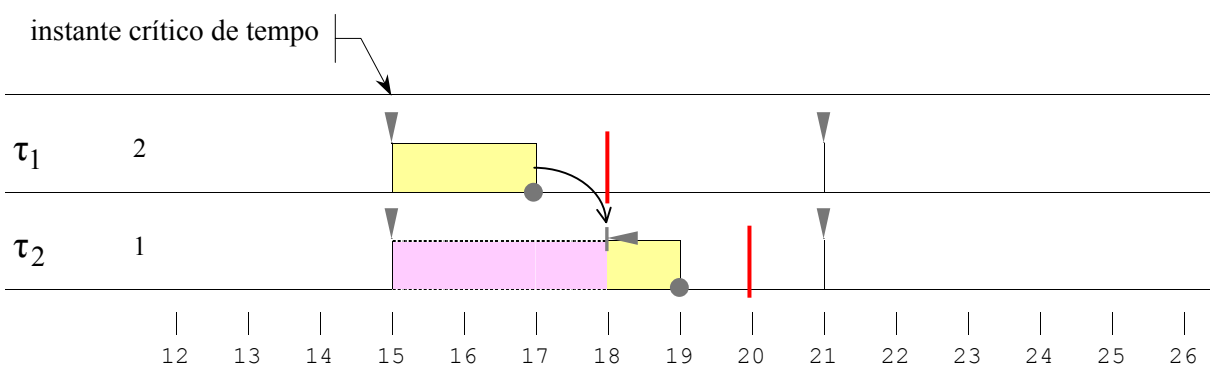


FIGURA 3.18: Exemplo de tarefas com deslocamento de fase de liberação

A última coluna da TABELA 3.7 mostra que com a aplicação da equação 3.8 a tarefa  $\tau_2$  perde seu prazo, porque considera-se que as duas tarefas sejam liberadas no mesmo instante (crítico de tempo). Pela especificação isto jamais acontece, como mostra a FIGURA 3.18. Este assunto está fortemente relacionado com as transações apresentadas no capítulo anterior.

### 3.4.6 Prazos no período

Nas seções anteriores considerou-se os prazos coincidindo com os períodos das tarefas. No entanto há casos: a) em que as tarefas devem disponibilizar seus resultados antes de final do período; e b) tratar dados disponíveis em janelas de tempo menores que o período. Para verificar a escalonabilidade de tarefas com  $D_i \leq T_i$  basta observar a equação 3.22 em que o tempo de resposta deve ser menor ou igual ao prazo.

$$R_i \leq D_i \quad (3.22)$$

As abordagens de análise pela utilização do processador e pela carga do processador não são adequadas.

### 3.4.7 Prazos arbitrários

Para a situação em que  $D_i > T_i$  as equações 3.19 e 3.20 devem ser modificadas de modo a contemplar o seguinte fato: uma tarefa pode ter mais de um serviço ativado ao mesmo tempo em que os serviços mais recentes podem sofrer interferência dos serviços anteriores.

A equação 3.23 define um cenário de tempo de resposta  $r_{i,q}$  para cada possível sobreposição  $q = 0, 1, 2, \dots$  (TINDELL; BURNS; WELLINGS, 1994). A solução segue raciocínio semelhante ao da equação 3.9.

$$r_{i,q} = (q+1) \cdot C_i + B_i + \sum_{j \in \text{hp}(i)} \left[ \frac{r_{i,q} + J_j}{T_j} \right] \cdot C_j \quad (3.23)$$

O termo  $q \cdot C_i$  estabelece o tempo de computação no cenário  $q$ . A equação 3.24 determina o tempo de resposta do serviço no cenário  $q$ . O termo  $qT_i$  é o instante de ativação do serviço relativo ao início do cenário.

$$R_{i,q} = r_{i,q} - qT_i + J_i \quad (3.24)$$

A equação 3.25 determina o pior tempo de resposta a partir dos  $q+1$  cenários. O número de cenários a considerar é o menor valor de  $q$  tal que  $R_{i,q} \leq T_i$ .

$$\begin{cases} k = \min \{q \in \mathbf{N} \mid R_{i,q} \leq T_i\} \\ R_i = \max_{q=0}^k R_{i,q} \end{cases} \quad (3.25)$$

A abordagem anterior aplica-se a prazos arbitrários menores, iguais, e maiores que o período. Considerando-se, por exemplo,  $D_i \leq T_i$ , então pela equação 3.25  $k = 0$ , e conseqüentemente a equação 3.23 reduz-se à equação 3.19.

A TABELA 3.8 e a FIGURA 3.19 exemplificam o escalonamento de tarefas com prazos arbitrários com especial atenção para a tarefa  $\tau_3$ .

TABELA 3.8: Dados das tarefas para a FIGURA 3.19

	Período $T_i$	Tempo de computação $C_i$	Prioridade $P_i$	Tempo de bloqueio $B_i$	Flutuação $J_i$	Prazo $D_i$
tarefa 1	60	10	3	0	0	60
tarefa 2	70	20	2	0	0	50
tarefa 3	140	70	1	20	0	210

As tarefas  $\tau_1$  e  $\tau_2$  possuem prazos menores ou iguais aos períodos, portanto têm apenas um cenário de tempo de resposta  $q = 0$  com tempos de resposta  $R_1 = 10$  e  $R_2 = 30$ . O tempo de bloqueio na tarefa  $\tau_3$  é ilustrativo apesar de inconsistente.

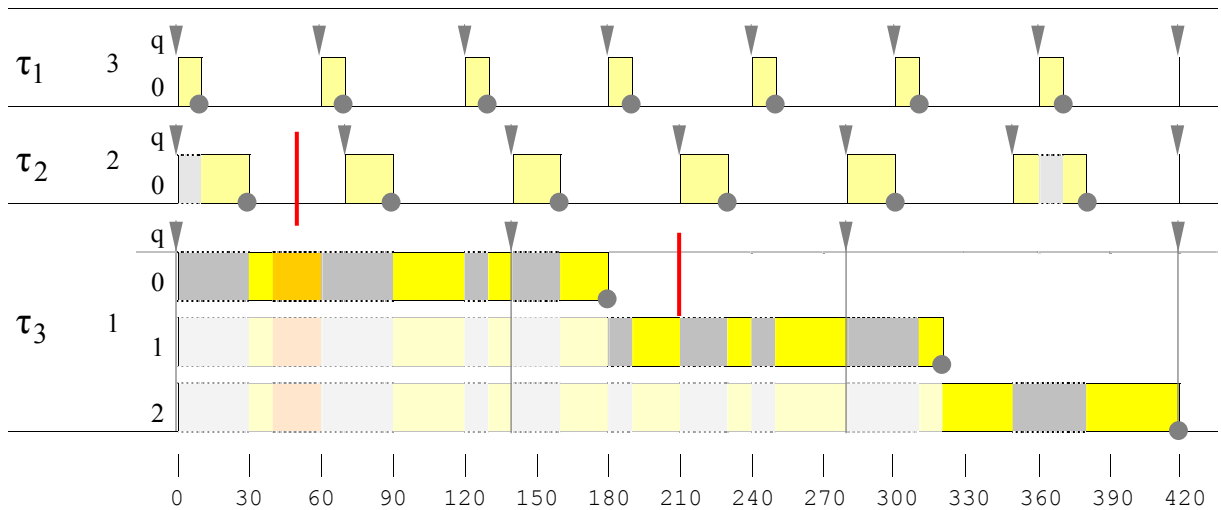


FIGURA 3.19: Exemplo de tarefas com prazos arbitrários

A tarefa  $\tau_3$  possui prazo maior que o período e pode ter mais de um cenário de tempo de resposta. Iniciando em  $q = 0$ , obtém-se o tempo de resposta para o primeiro cenário como mostra a seqüência de cálculos abaixo:

$$r_{3,0}^{(1)} = 70 + 20 + 10 + 20 = 120$$

$$r_{3,0}^{(2)} = 70 + 20 + \lceil 120/60 \rceil \cdot 10 + \lceil 120/70 \rceil \cdot 20 = 150$$

$$r_{3,0}^{(3)} = 70 + 20 + \lceil 150/60 \rceil \cdot 10 + \lceil 150/70 \rceil \cdot 20 = 180$$

$$r_{3,0}^{(4)} = 70 + 20 + \lceil 180/60 \rceil \cdot 10 + \lceil 180/70 \rceil \cdot 20 = 180$$

Na versão (3) o tempo de resposta do cenário torna-se estável. A TABELA 3.9 acompanha os cálculos. Enquanto a coluna 4 for igual a não, prossegue-se com o próximo cenário. Ao final seleciona-se o maior valor da coluna  $R_3$ , que equivale ao pior tempo de resposta da tarefa  $\tau_3$ .

TABELA 3.9: Acompanhamento dos cálculos de tempo de resposta

$q$	$r_{3,q}$	$R_{3,q}$	$R_{3,q} \leq T_i = 140$	$R_3$
0	180	$180 - 0 \cdot 140 = 180$	não	180
1	320	$320 - 1 \cdot 140 = 180$	não	180
2	420	$420 - 2 \cdot 180 = 140$	sim	<del>140</del>

No caso da equação 3.23 gerar um valor  $r_{i,q}^{(v+1)}$  maior que  $D_i + qT_i$ , o tempo de resposta do cenário não converge, e a tarefa não é escalonável.

### 3.5 ATRIBUIÇÃO DE PRIORIDADES

As equações 3.19 e 3.20 consideram o prazo igual ao período e a atribuição de prioridades por RMA. Com prazos arbitrários a abordagem por DMA parece mais adequada (BRIAND; ROY, 1999).

Na abordagem por DMA tarefas com prazo mais curto recebem maior prioridade, e em caso de empate, prevalece o critério RMA. BURNS e WELLINGS (2001) provam que qualquer conjunto de tarefas escalonável por RMA também é escalonável por DMA. A ordem por DMA é a mais abrangente.

#### 3.5.1 Algoritmo de atribuição de prioridades

Para prazos arbitrários não há um algoritmo simples como taxas ou prazos monotônicos para determinar a ordem de prioridades mais satisfatória. A prova em BURNS e WELLINGS (2001) já parte de um conjunto de tarefas escalonável ! O teorema a seguir fundamenta um algoritmo de atribuição de prioridades genérico: “*Se a tarefa  $\tau$  é exeqüível com a menor prioridade, e, se existe uma ordem de prioridades em que todo o conjunto de tarefas é exeqüível, então existe uma ordem em que  $\tau$  possui a menor prioridade*” (AUDSLEY *et alii*, 1993).

Uma tarefa com a menor prioridade sofre interferência das demais tarefas (com prioridade maior) independente da ordem destas. Logo, se a tarefa for escalonável na menor prioridade, pode receber a menor prioridade. O mesmo raciocínio aplica-se de forma recorrente ao conjunto das tarefas restantes até a última tarefa com a maior prioridade, ou até concluir que o conjunto de tarefas não é escalonável. Desta forma prova-se o teorema acima por indução.

O ALGORITMO 3.2 implementa a especificação acima. A variável  $t$  é o conjunto ordenado de tarefas na entrada e o conjunto de tarefas ordenado por

prioridade na saída;  $n$  é a cardinalidade do conjunto  $t$ ; e  $ok$  responde se o conjunto de tarefas é ou não escalonável. O procedimento *trocaTarefas()* troca a posição da tarefa  $i$  com a posição da tarefa  $p$ . O procedimento *testaEscalonabilidade()* aplica a equação 3.25 à tarefa  $i$ . Se alguma tarefa  $i$  não for escalonável, o procedimento *atribuiPrioridade()* pára e retorna  $ok = \text{falso}$ .

---

### ALGORITMO 3.2: Algoritmo para atribuir prioridades (sem bloqueios)

---

```

procedure atribuiPrioridade
  (t: in out Tarefas; n: natural; ok: out boolean) is
begin
  for i in 1..n loop
    for p in i..n loop
      -- troca tarefas i e p de posição
      if p>i then trocaTarefas (t, i, p); end if;
      -- verifica se a tarefa i é escalonável
      testaEscalonabilidade (t, i, ok);
      exit when ok;
    end loop;
    exit when not ok;
  end loop;
end atribuiPrioridade;

```

---

O algoritmo funciona de forma trivial para tarefas sem bloqueio. Como o bloqueio equivale a uma interferência de tarefas com prioridade menor, deve-se calcular o tempo de bloqueio  $B_i$  da tarefa na posição em que será testada sua escalonabilidade, levando em conta o protocolo de travamento de recursos em uso. Sob os protocolos PCP e HL, por exemplo, os recursos devem receber suas prioridades de acordo com as posições das tarefas em que são usados.

## 3.6 LEVANTAMENTO DE FERRAMENTAS

Esta seção apresenta a revisão de diversas ferramentas para análise e simulação de escalonamento em sistemas em tempo real. Os dados foram compilados a partir de artigos publicados em revistas, dissertações de mestrado e teses de doutorado,

propaganda encontrada em páginas de empresas comerciais e páginas de instituições de ensino.

O QUADRO 3.2 mostra a lista de ferramentas examinadas juntamente com a identificação de algumas de suas características.

QUADRO 3.2: Características das ferramentas pesquisadas

Ferramentas	Analítico	Simulação	Real	Abstrato	Livre	Acessível	SO	Ano	Acadêmico Comercial Outros
							Linux, Unix Windows		
1 AFTER	X	-	X	-	-	-	W, U	2003	-
2 ARTISST	-	X	-	X	X	X	L	2003	O
3 ASSERTS	X	X	-	X	-	-	X	1999	C
4 CAISARTS	X	-	-	X	-	-	-	1996	A
5 CarbonKernel	-	X	X	-	X	X	L	2002	C
6 Cheddar	X	-	X	X	X	X	W, L	2006	A
7 DET/SAT/SIM	X	X	-	X	-	-	-	1998	O
8 DRTSS / PERTSSim	-	X	-	X	-	-	U	1996	A
9 MatLab TrueTime	-	X	-	-	X	X	U, L, W	2006	C
10 Perf	X	X	X	-	X	X	W	2002	A
11 PerfoRMAx	X	X	-	X	-	X	W	2006	C
12 RapidRMA	X	X	-	X	-	X	U, L, W	2006	C
13 Scheduler 1-2-3	X	X	-	-	-	-	U	1988	A
14 ScheduLite	X	-	-	X	-	-	W	1996	A
15 SEW	X	-	X	-	X	X	L, W	1999	A
16 STRESS	X	X	-	X	-	-	U	1994	A
17 TEV	X	X	X	-	X	X	W	2006	A
18 TimeWiz	X	X	X	-	-	X	W	2006	C
19 UTSA	-	X	X	-	X	X	U, L, W (Java)	2005	A
20 YASA	-	X	-	X	X	X	L, U, W	2006	A

A coluna **Analítico** indica se a ferramenta analisa os aplicativos e determina suas propriedades de escalabilidade baseado na teoria de escalonamento. A coluna **Simulação** indica se a ferramenta experimenta os aplicativos com cargas artificiais para estabelecer valores estatísticos para suas propriedades de escalabilidade sem garantias de exeqüibilidade. Por um lado, a simulação entra em campo para suprir a falta de cobertura teórica, por outro lado pode ser utilizada para visualizar a evolução temporal dos aplicativos. As colunas **Real** e **Abstrato** indicam se os aplicativos são



representados através de modelos reais ou abstratos, respectivamente, por exemplo, em linguagem de programação tipo C, Java, ou através grafos de tarefas, recursos, etc. As colunas **Livre**, **Acessível**, **SO** e **Ano** indicam se a ferramenta é de livre acesso, se está de fato acessível no momento, sob que sistema operacional pode ser utilizada e até que ano foi possível detectar sua presença. A ferramenta 19 da lista está disponível através de *browsers*. A última coluna indica a origem das ferramentas. Em geral resultam de trabalhos acadêmicos como monografias, dissertações de mestrado e teses de doutorado. Algumas evoluíram a partir de outras e passaram a ser comercializadas associadas a ambientes de desenvolvimento proprietários, tais como RapidRMA e perfoRMAx.

No Apêndice F mostra-se alguns detalhes a respeito de cada ferramenta. A grande parte das ferramentas de análise está baseada nas teorias clássicas de escalonamento. As ferramentas de simulação apresentam maior evolução, uma vez que a cada geração incorporam novas políticas de escalonamento, especialmente relativas ao escalonamento dinâmico, tais como EDF e LLF. O suporte gráfico das ferramentas aos modelos e resultados também evoluiu, tornando suas interfaces mais agradáveis e mais expressivas.

Qualquer uma das ferramentas pode ser utilizada no ensino com maior ou menor grau de adequação. Quanto mais reais forem os modelos, mais trabalhosos seriam os exercícios, talvez deslocando o foco. No entanto, algumas ferramentas foram elaboradas para fins educativos, tais como Cheddar e TEV.

### 3.7 CONCLUSÃO

Neste capítulo apresentou-se um resumo da teoria de escalonamento tradicional, abrangendo: a) a classificação dos escalonadores em estáticos (CE), dinâmicos com prioridade fixa (HPF, RMA, DMA) e dinâmicos com prioridade dinâmica (EDF, LLF); b) a análise de escalonabilidade sob a ótica da utilização do processador, da carga do processador, e do tempo de resposta; c) conjuntos de tarefas independentes; d) conjuntos de tarefas com bloqueios na disputa por recursos compartilhados; e) a

introdução dos protocolos PIP, PCP e HL; e f) extensões para consideração de flutuações, deslocamento, prazos no período e prazos arbitrários.

Apresentou-se, ainda, um quadro resumido com o resultado da pesquisa por ferramentas para análise e simulação de escalonamento já desenvolvidas e eventualmente disponíveis.

## **4 DESCRIÇÃO DA FERRAMENTA**

A maior parte das ferramentas apresentadas na Seção 3.6 (e detalhadas no Apêndice F) não são adequadas ao ensino e aprendizado. Algumas ferramentas objetivam atender a ambientes de desenvolvimento específicos no tratamento de problemas reais, envolvendo linguagens e sistemas operacionais comerciais. Outras foram desenvolvidas para testar e validar certas teorias de escalonamento sem preocupações com a interface pessoa-máquina. Algumas ainda podem ser utilizadas no ensino de conteúdos específicos, por exemplo, o comportamento do escalonamento LLF.

No sentido de abranger um espectro mais amplo do ensino e aprendizado do escalonamento, propõe-se uma ferramenta que integra os diversos conteúdos relacionados, tais como análise do comportamento temporal e casual de algoritmos, medição de tempos de espera e execução e análise e simulação de escalonamento.

Para simplificar este propósito propõe-se uma abordagem abstrata não baseada em problemas concretos nem ambientes específicos como linguagens e sistemas operacionais em particular. Para tanto pode-se descrever o comportamento de um sistema através de imperativos de intenções baseadas em probabilidades. Mesmo assim, as descrições resultantes guardam semelhança com linguagens comuns de programação.

### **4.1 REQUISITOS PARA A FERRAMENTA**

Os principais requisitos para o desenvolvimento da ferramenta de apoio ao ensino e aprendizado do escalonamento de tarefas em sistemas tempo real compreendem:

- suporte aos fundamentos da teoria de escalonamento apresentados no Capítulo 3, seguindo os conceitos de tarefas e serviços;
- análise e simulação do escalonamento de sistemas a partir de tabelas de propriedades de tarefas, processadores e recursos;

- construção de algoritmos para capturar o comportamento temporal e casual de tarefas através de uma linguagem de programação;
- simulação de tarefas expressas na linguagem de programação através de uma máquina virtual associada aos escalonadores;
- construção de tabelas de propriedades de tarefas, processadores e recursos a partir dos algoritmos das tarefas através da segmentação dos algoritmos e medição de tempos de pior e melhor caso;
- especificação da linguagem de programação abstrata, homomorfa a linguagens de uso corrente, tais como C++, Java e Pascal;
- especificação dos elementos léxicos e sintáticos da linguagem programação baseados na língua portuguesa, minimizando a utilização de símbolos, e com construtos próximos à linguagem natural;
- cobertura de diversos paradigmas de programação, tais como fluxo por saltos praticado em montadores, programação estruturada e modular e programação através de métodos em objetos ou componentes;
- suporte a regiões críticas em programas, implementados através de recursos com múltiplas instâncias;
- suporte à comunicação e sincronização de tarefas através de encontros e construtos, tais como `accept` e `select...end` da linguagem Ada;
- suporte à comunicação assíncrona através de encontros, simulando filas de mensagens com tamanhos variáveis;
- suporte ao tratamento de faltas semelhante ao tratamento de exceções na linguagem Ada ou ao construto `try...catch` na linguagem C++;
- suporte à diversidade temporal e casual através de árbitros de tempo, contagem, extensão e probabilidade, com diversas distribuições de probabilidade, tais como distribuição uniforme, polinomial e exponencial;
- implementação de um gerador de números aleatórios com distribuição uniforme, grande abrangência e capacidade de repetição de séries de amostras;

- suporte à análise e simulação com multiprocessamento simétrico e assimétrico, e recursos de múltiplas instâncias;
- manutenção de experimentos em arquivos para facilitar sua repetição e o intercâmbio estudante-estudante-professor;
- apresentação de interface gráfica para acompanhamento de todos os passos na simulação;
- suporte para comparação visual de gráficos e resultados entre diversos experimentos;
- ambiente operacional autocontido e autônomo de simples instalação e sem dependência de outros aplicativos.

O conjunto de requisitos abrange as principais atividades relacionadas à escalonabilidade em sistemas em tempo real desde a medição de tempos, análise e simulação do escalonamento e da simulação de algoritmos.

## **4.2 DESENVOLVIMENTO DA FERRAMENTA**

Para atender aos requisitos listados na Seção 4.1 a ferramenta subdivide-se em 5 módulos conforme as principais funcionalidades: módulo 1) edição de arquivos fonte textuais e arquivos fonte XML; módulo 2) compilação e carga de arquivos fonte com conversão de arquivos fonte em árvores de sintaxe e geração reversa de arquivos fonte; módulo 3) tradução de árvores de sintaxe com geração de grafos de segmentos e planilhas de tempos; módulo 4) análise e simulação de escalonamento com edição de planilhas de tempos; e módulo 5) simulação de tarefas e rotinas pela interpretação de grafos de segmentos.

A FIGURA 4.1 mostra o modelo da ferramenta em que as caixas com computador representam os pontos de interação do usuário com a ferramenta, as caixas arredondadas representam os principais processos, as caixas abertas à direita representam depósitos de objetos e as setas representam o fluxo de informações entre usuários, processos e depósitos. Os grupos de elementos destacados pelas cores relacionam-se com os módulos da seguinte forma: módulo 1 em cor cinza, módulo 2

em cor azul, módulo 3 em cor verde, módulo 4 em cor cáqui e módulo 5 em cor magenta.

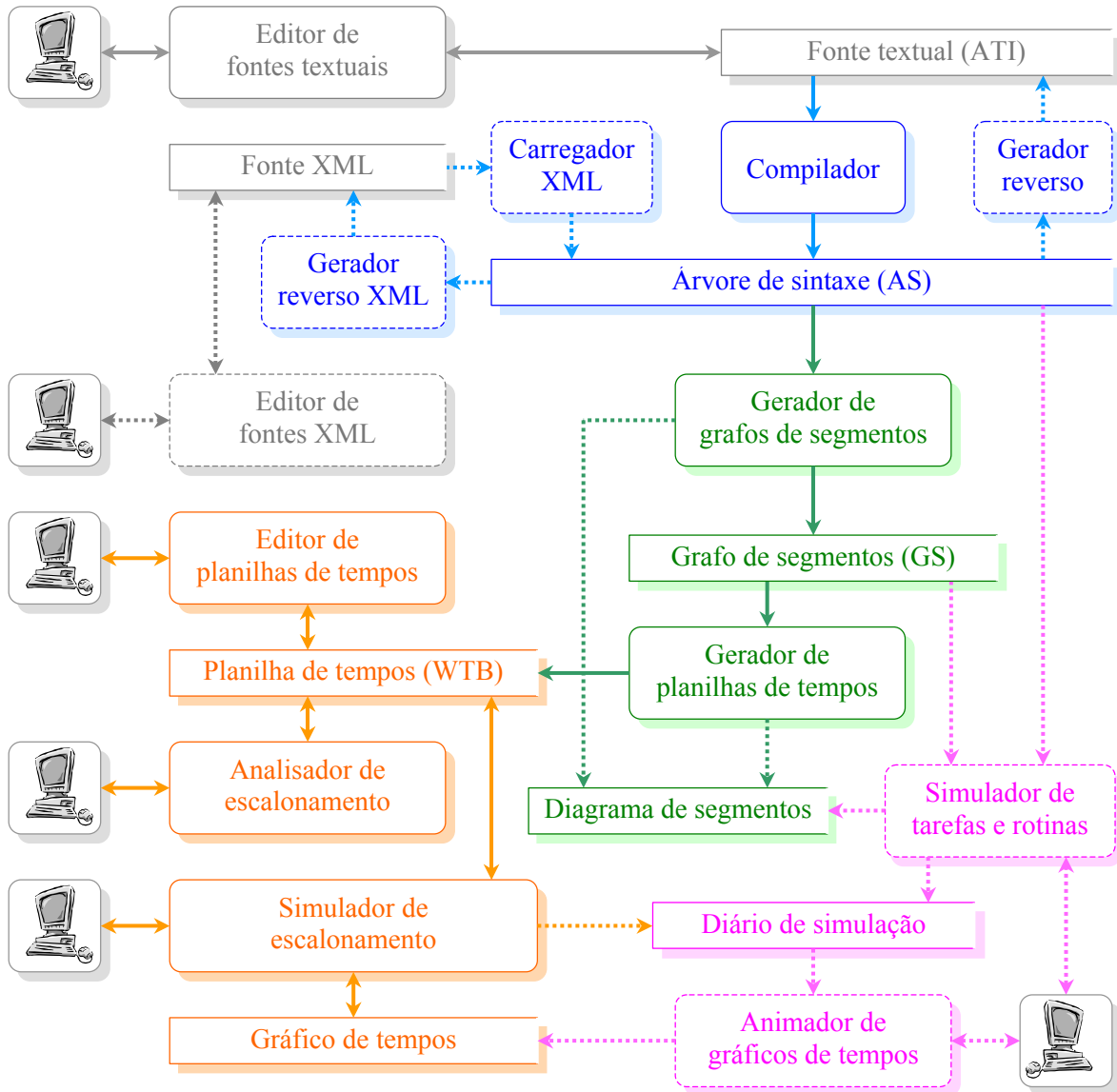


FIGURA 4.1: Modelo funcional da ferramenta *AnimaTi*

A ferramenta dispõe, ainda, de um programa principal que aciona os módulos e coordena os seus estados. Além disso oferece serviços de configuração do perfil dos usuários (cores, valores default) e oferece um visor de fatos de análise (erros, alertas e avisos) (Seção 4.2.2.5).

## 4.2.1 Módulo de criação e edição de fontes

O objetivo do módulo de criação e edição de fontes é criar e editar arquivos fonte contendo algoritmos expressos na linguagem definida para a ferramenta. Define-se dois tipos de arquivos fonte: a) arquivo fonte textual, construído a partir da gramática da linguagem *AnimaTi* (Apêndice A); e b) arquivo fonte XML, construído a partir da descrição formal dos construtos da linguagem *AnimaTi* para XML (Apêndice B).

A unidade básica de trabalho com a ferramenta é o *experimento*, sempre associado a um arquivo fonte principal. Este arquivo fonte pode agregar diversos outros arquivos fonte por um mecanismo de inclusão recorrente. Assim o experimento é caracterizado por uma árvore de arquivos fonte. Experimentos podem ser construídos a partir de arquivos fonte textuais e XML intercalados.

O módulo possui dois componentes de acordo com a FIGURA 4.1: a) o editor de fontes textuais; e b) o editor de fontes XML. Nas próximas seções descreve-se estes componentes.

### 4.2.1.1 Editor de fontes textuais

O editor de fontes textuais é um editor básico para textos, operando com caracteres de tamanho fixo acrescido de funcionalidades orientadas a construção de blocos, tais como indentação de linhas e transformação de linhas em comentários. Ressalta-se que o editor não conhece a sintaxe da linguagem *AnimaTi*.

O editor pode ser instanciado mais de uma vez. Cada instância do editor pode operar com múltiplos arquivos fonte textuais distintos. Determinado arquivo fonte textual por sua vez pode ser editado simultaneamente em mais de um editor com versões distintas. O usuário deve administrar esta situação apoiado por mensagens de alerta. A extensão default dos arquivos fonte textuais é *ati*.

A FIGURA 4.2 mostra o visor para o editor de arquivos fonte textuais e seus visores auxiliares.

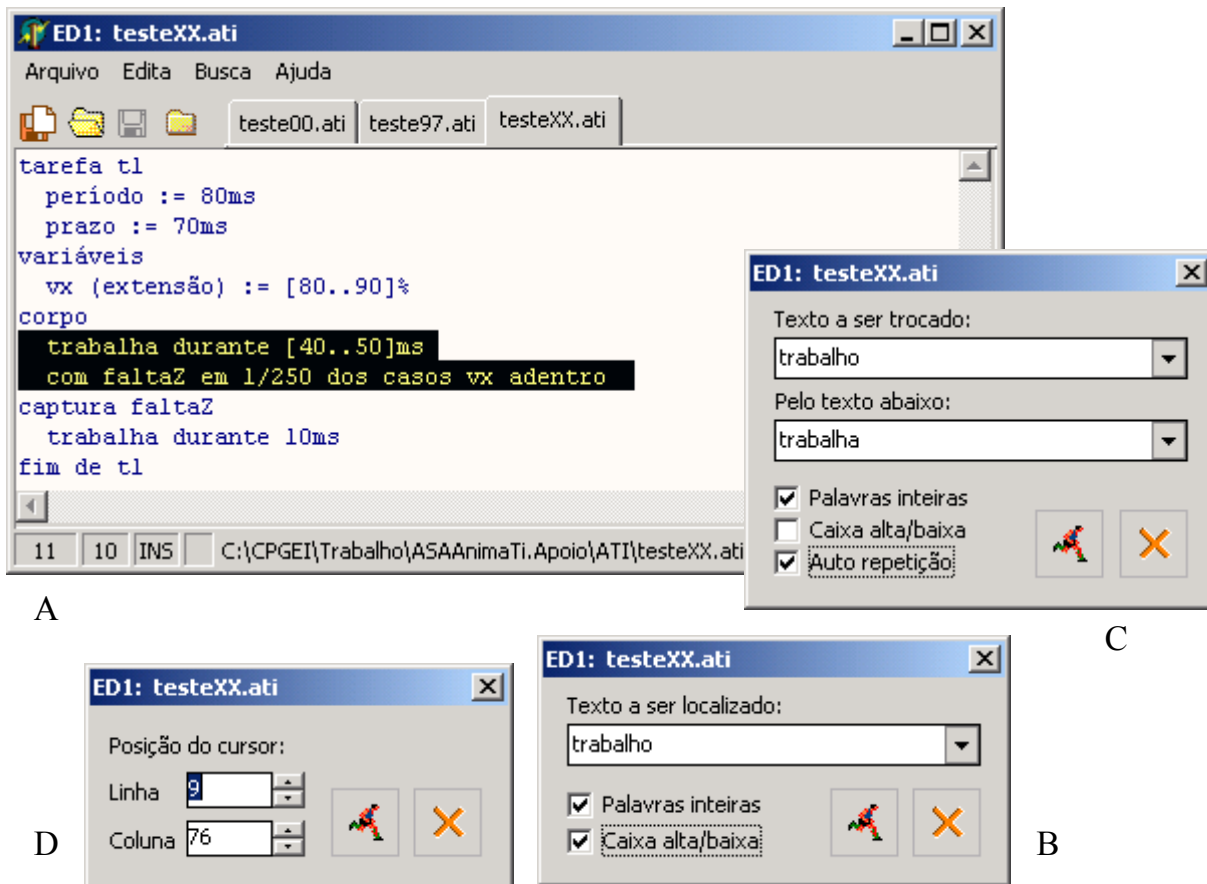


FIGURA 4.2: Visores do editor de fontes textuais

As funcionalidades disponibilizadas pelo editor de arquivos fonte textuais incluem: a) criação e manutenção de arquivos textuais; b) impressão; c) deslocamento em blocos de linhas; d) transformação de blocos de linhas em comentários; e) localização de palavras e frases (FIGURA 4.2B); f) localização e troca de palavras e frases por outras (FIGURA 4.2C); e g) posicionamento do cursor por linha e coluna (FIGURA 4.2D).

O editor de arquivos fonte textuais pode ser acionado e/ou instanciado por iniciativa do visor de fatos de análise (erros, alertas e avisos) para mostrar determinado arquivo fonte textual e ressaltar a parte do texto envolvida em determinado fato de análise.



#### 4.2.1.2 Editor de fontes XML

O editor de fontes XML opera diretamente sobre a estrutura hierárquica de arquivos XML. A FIGURA 4.3 exemplifica os aspectos visuais em que os construtos da linguagem são representados por caixas recorrentes. Por exemplo, o ator `repete...fim` mais externo é representado por uma caixa de repetição cujo interior está ocupado por duas caixas justapostas representando dois segmentos. A caixa do ator `caso...fim` está dividida em três casos, e o interior de cada caso constitui um grupo de segmentos. Ressalta-se que o editor conhece a sintaxe da linguagem *AnimaTi*.

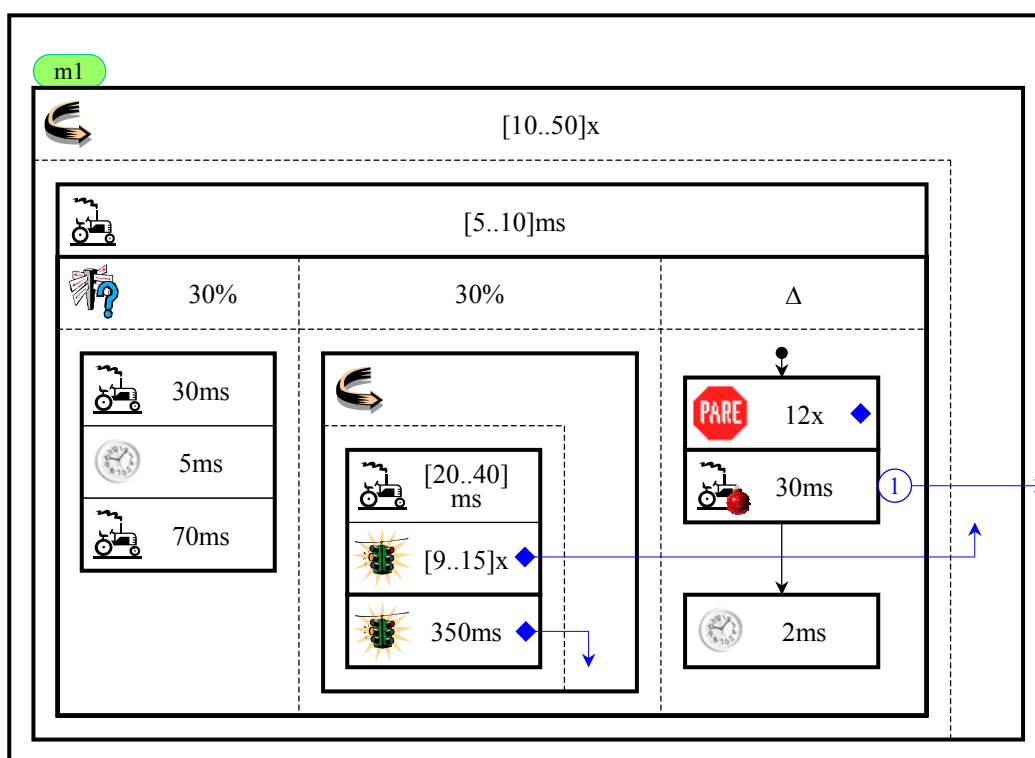


FIGURA 4.3: Exemplo de edição do ALGORITMO 4.3

No Apêndice B descreve-se a sintaxe completa da linguagem *AnimaTi* para arquivos XML. Para preservação dos aspectos visuais introduzidos pelo editor, esta descrição é acrescida das propriedades gráficas, por exemplo, geometria. A extensão default dos arquivos fonte XML é *xml*.

No Apêndice D mostra-se os elementos visuais para as definições XML. As figuras guardam semelhança com as figuras para grafos de segmentos (Apêndice C) e são baseadas na notação de Nassi-Shneiderman.

#### 4.2.1.3 Linguagem *AnimaTi*

Parte importante da ferramenta é sua capacidade de realizar experimentos abstratos completamente dissociados de problemas reais. A princípio isto não é uma vantagem. Contudo, em ambientes de sistemas embarcados e tempo real, os experimentos envolveriam diversas áreas de conhecimentos, tais como eletrônica, sistemas operacionais e linguagens de programação, além da disponibilidade de um laboratório com protótipos de *hardware* e outros equipamentos e da disponibilidade de *software* adequado. Este cenário impõe severas limitações à diversidade e ao tempo de realização de experimentos.

A ferramenta propõe a linguagem de programação imperativa *AnimaTi*, com características sintáticas semelhantes às de linguagens de programação existentes, tais como C e Ada, porém, fundamentalmente diferente em sua semântica. A linguagem não manipula objetos que representam propriedades em modelos computacionais, tais como valores, caracteres, contadores, condições lógicas, e sim, manipula elementos *mutantes* que determinam o comportamento casual e temporal em algoritmos.

Outra característica da linguagem é sua semelhança com linguagens naturais, como mostram as frases `trabalha 10ms e por 20 vezes repete...fim`. A sintaxe é baseada em blocos, e todos os construtos complexos possuem cláusulas divisoras e terminadoras, que envolvem os blocos internos como mostram as palavras em negrito no ALGORITMO 4.1.

### ALGORITMO 4.1: Exemplo com o construto `bloco...fim`

---

```

bloco
  trava r1 com falta1 após 10ms
  caso em 20% dos casos
    -- bloco interno do primeiro caso
  ou em 50% dos casos
    -- bloco interno do segundo caso
  fim de caso
captura falta1
  -- tratamento da falta
captura falta2
  -- tratamento da falta
fim de bloco

```

---

Além disso, a linguagem utiliza poucos símbolos e geralmente em posições naturais ao senso comum, por exemplo, ( ) para delimitar listas de argumentos, [ .. ] para construir intervalos fechados de mutantes intervalares com distribuição de probabilidade uniforme. A sintaxe não requer a utilização de símbolos separadores e/ou delimitadores de frases, tais como vírgula entre argumentos e ponto e vírgula ao final de declarações, contudo podem ser utilizados a gosto do usuário. A linguagem não define palavras reservadas, e somente a posição sintática das palavras confere a estas seus papéis nos construtos da linguagem.

O analisador léxico do compilador reconhece automaticamente arquivos fonte textuais escritos em português ou inglês, preservando isomorfismo de construtos da linguagem, por exemplo, `por 20 vezes repete...fim` equivale sintaticamente a `for 20 times repeat...end`.

Um experimento é descrito por uma árvore de arquivos fonte. Cada arquivo contém uma seqüência de zero ou mais elementos. A linguagem define os 6 tipos de elementos listados abaixo:

- `arquivo`: define o ponto de inclusão de um arquivo fonte textual (escritos em português ou inglês) ou um arquivo fonte XML; qualquer elemento desta lista não pode ser subdividido e distribuído entre diversos arquivos fonte;

- `encontro...fim`: define um ponto de encontro que instrumenta a sincronização de tarefas e comunicação entre tarefas através dos atores `aceita encontro` e `encontra`; opera nos modos síncrono e assíncrono e troca mutantes através da passagem bidirecional de argumentos;
- `processador...fim`: define um processador e suas características, tais como base de tempo (relativo) e multiplicidade; cada tarefa está associada a um processador;
- `recurso...fim`: define um recurso e suas características, tais como política de filas e multiplicidade; os recursos determinam as regiões críticas em tarefas e controlam a mútua exclusão em recursos compartilhados através dos atores `trava`, `libera` e `região...protege...fim`;
- `rotina...fim`: define um algoritmo a ser executado por tarefas e outras rotinas; suporta o conceito de módulo da programação modular e o conceito de método da programação orientada a objetos; o comportamento casual e temporal da rotina e do chamador pode ser modificado pela troca de mutantes através da passagem bidirecional de argumentos; rotinas são acionadas através do ator `executa`;
- `tarefa...fim`: define um algoritmo a ser acionado por um escalonador de acordo com a política de filas em questão; cada tarefa define um conjunto de parâmetros, tais como período de ativação, prazo, flutuação e prioridade.

Tarefas constituem os elementos fundamentais para o módulo de análise e simulação de escalonamento (FIGURA 4.1, cáqui). Para tal requerem características restritivas, tais como não podem conter ciclos de saltos e todos os atores devem ter seus tempos de execução limitados. As tarefas implementam os algoritmos associados ao ciclo de vida dos serviços (Seção 2.3.2), deixando os ciclo de vida das tarefas por conta dos escalonadores e suas políticas de escalonamento.

#### 4.2.1.4 Construtos para algoritmos

Os elementos do tipo tarefa e rotina descrevem algoritmos através dos construtos de atores e suas cláusulas. Os atores são divididos em dois grupos: a) atores de ação, que geram tempos de execução e de espera; e b) atores de controle, que organizam o comportamento casual e temporal dos algoritmos. No Apêndice A descreve-se a sintaxe completa da linguagem *AnimaTi*.

O grupo de atores de ação é formado pelos atores `trabalha` e `retarda`. O ator `trabalha` descreve um tempo de execução `Ci`, que pode ser interrompido pela ocorrência de faltas em instantes especificados pela cláusula `com`. Estas interrupções podem simular eventos, tais como divisão por zero e exceção de endereçamento.

O grupo de atores de controle implementa diversos aspectos relacionados a algoritmos, recursos, encontros, faltas e relógios, tais como:

- modelos de programação: a) programação por marcadores e saltos através dos atores `salta para` e `bifurca...fim`, b) programação estruturada, pela implementação de seleção, repetição e escapes através dos atores `caso ...fim`, `se...fim`, `repete...fim`, `abandona` e `itera`, e c) programação modular e orientada a objetos através do ator `executa`;
- tratamento de faltas: interceptação de faltas pelas cláusulas `captura` embutidas no ator `bloco...fim` e no corpo de tarefas e rotinas; repasse de faltas ocorridas em rotinas aos chamadores; e provocação de faltas através do ator `provoca`;
- operações com recursos: simulação de semáforos e regiões críticas através dos atores `libera`, `trava` e `protege`;
- operações com encontros: simulação de sincronização e comunicação entre tarefas através dos atores `aceita encontro` e `encontra`, além dos construtos tipo `seleciona`.

Diversos atores suportam guardas, que determinam quando os atores devem atuar. Por exemplo, o ator `abandona` em 10% dos casos interrompe a repetição envolvente.

#### 4.2.1.5 Mutantes

Mutantes são elementos que geram números aleatórios de acordo com determinada distribuição de probabilidade. Mudam de estado dinamicamente e simulam condições de acaso, contagens e intervalos de tempo. A linguagem provê quatro tipos de mutantes:

- extensão (relativa): expressa por um número racional no intervalo  $[0,1]$  (ou na forma percentual), por exemplo, 90% dentro do tempo de execução de um trabalho;
- intervalo de tempo: expresso por um número racional positivo em unidades de tempo, por exemplo, 3.5ms de espera;
- número de vezes: expresso por um número natural, por exemplo, repetição por 25 vezes parte de um algoritmo;
- probabilidade: expressa por um número racional no intervalo  $[0,1]$  (ou na forma percentual), por exemplo, em 0.3 dos casos.

Mutantes podem ser declarados literalmente nos construtos da linguagem, ou associados a variáveis e parâmetros. Na forma de variáveis e parâmetros é possível reconfigurar os mutantes ao longo dos algoritmos. Ainda, na forma de parâmetros (bidirecionais), é possível influenciar o comportamento de rotinas e seus chamadores.

Os valores atuais de mutantes são gerados em três etapas: a) obtenção de um número aleatório uniformemente distribuído; b) adaptação da distribuição de probabilidade; e c) adequação às características de um mutante em particular.

Um gerador de números aleatórios produz séries de números aleatórios uniformemente distribuídos (KNUTH, 1998) no intervalo de inteiros  $[0,n]$  sem repetição. Estes números são reagrupados em classes de modo que os índices das classes sejam números aleatórios com distribuições de probabilidade arbitrárias. A FIGURA 4.4 mostra um exemplo do modelo básico deste mecanismo que perfazem as etapas a) e b).

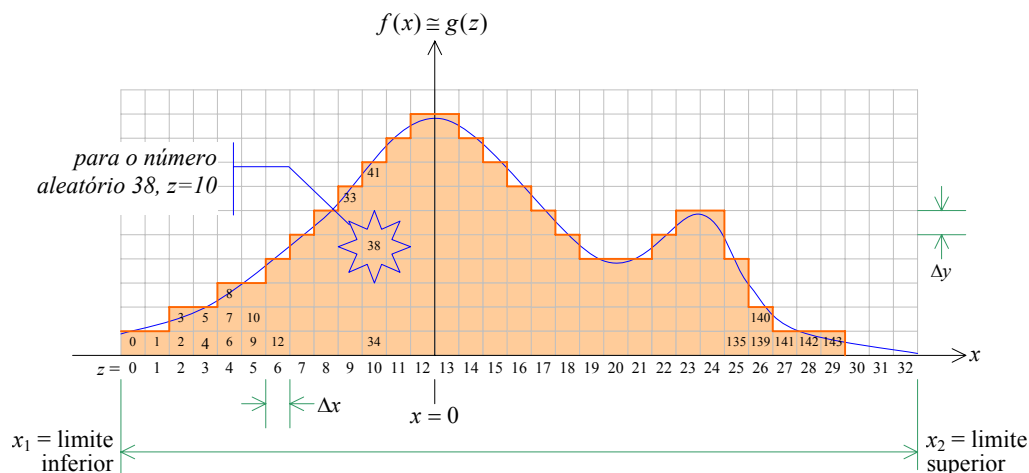


FIGURA 4.4: Exemplo de modelo de valoração dos mutantes

Para uma função de densidade de probabilidade  $f(x)$ ,  $x$  no intervalo  $[x_1, x_2]$ , determina-se uma função discreta semelhante  $g(z)$  em que  $z$  é o índice de classes no intervalo de inteiros  $[0..k]$ . A área sob a curva desta função nos limites de  $x$  é sobreposta por uma grade ortogonal com no máximo  $n$  células. A cada célula da grade associa-se um número inteiro distinto no intervalo  $[0, n]$ . As colunas da grade formam as classes. Os valores  $\Delta x$  e  $\Delta y$  definem as características da grade e devem ser adequados para que a função  $g(z)$  atenda aos requisitos.

No exemplo da FIGURA 4.4, a numeração das células é crescente no sentido do aumento da área, da esquerda para a direita e de baixo para cima. Portanto, para o número aleatório  $na$  vale  $g(z-1) < na \leq g(z)$ . Por conveniência,  $g(-1) = 0$ . Pelo exemplo, o número 38 produzido pelo gerador de números aleatórios pertence à classe 10.

A etapa c) leva em consideração as características de mutantes em particular e transforma linearmente índices de classe  $z$  em valores atuais de mutantes  $v$  pela equação 4.1.

$$v = \frac{b-a}{x_2-x_1} \left( \Delta x \cdot \left\lfloor \frac{2z+1}{2} \right\rfloor - x_1 \right) + a \quad (4.1)$$

Os valores  $a$  e  $b$  são os limites impostos aos mutantes. Para mutantes do tipo probabilidade e extensão os limites são  $a=0$  e  $b=1$ . Para mutantes do tipo intervalo de

tempo e número de vezes, os limites são definidos na declaração de mutantes, por exemplo, `trabalha poligonal(3,4;7,9;11,1)[4..10]ms` em que  $a=4$  e  $b=10$ .

O ambiente de simulação oferece diversas funções de densidade de probabilidade para definir o comportamento de mutantes, relacionadas na lista a seguir:

- $[x_1..x_2]$ : distribuição uniforme no intervalo limitado pelos valores  $x_1$  e  $x_2$ ;
- $\text{exponencial}(\lambda; x_1, x_2)$ : distribuição exponencial com o parâmetro  $\lambda$ , no intervalo limitado pelos valores  $x_1$  e  $x_2$ ;
- $\text{normal}(x, \sigma; x_1, x_2)$ : distribuição normal em torno da média  $x$  com desvio padrão  $\sigma$ , no intervalo limitado pelos valores  $x_1$  e  $x_2$ ;
- $\text{poligonal}(x_1, y_1; \dots; x_n, y_n)$ : distribuição definida pelo polígono aberto que passa pelos pontos indicados, no intervalo limitado pelos valores  $x_1$  e  $x_n$ ;
- $\text{polinomial}(a_0, p_0; \dots; a_n, p_n; x_1, x_2)$ : distribuição definida pelo polinômio de ordem arbitrária  $a_0x^{p_0} + \dots + a_nx^{p_n} = 0$ , no intervalo limitado pelos valores  $x_1$  e  $x_2$ ;
- $\text{pontual}(x_1, \dots, x_n)$ : distribuição uniforme nos pontos indicados, por exemplo, `pontual(5,8,12)ms` declara que o mutante deve oferecer chances iguais para 5ms, 8ms e 12ms;
- $\text{spline}(x_1, y_1; \dots; x_n, y_n)$ : distribuição definida pela curva *spline* que passa pelos pontos indicados, no intervalo limitado pelos valores  $x_1$  e  $x_n$ .

Do ponto de vista da linguagem *AnimaTi*, nomes de funções e listas de argumentos são variáveis. Novas funções podem ser acrescentadas independente da sintaxe da linguagem e utilizadas pelo simulador de tarefas e rotinas.

#### 4.2.2 Módulo de compilação e carga de arquivos fonte

O objetivo principal do módulo de compilação e carga é gerar a árvore de sintaxe a partir de arquivos fonte textuais e XML. A geração reversa permite reproduzir arquivos fonte textuais e XML a partir da árvore de sintaxe. Assim arquivos fonte



textuais podem ser convertidos em arquivos fonte XML e vice-versa, bem como arquivos fonte textuais podem ser convertidos de português para inglês e vice-versa.

Arquivos XML são adequados para transporte de dados entre estações e plataformas com maior controle de integridade. Experimentos no ambiente *AnimaTi* podem ser apresentados em arquivos XML para facilitar o intercâmbio estudante-estudante-professor.

O módulo possui quatro componentes de acordo com a FIGURA 4.1: a) compilador; b) gerador reverso; c) gerador reverso XML; e d) carregador XML. Nas próximas seções descreve-se estes componentes.

#### 4.2.2.1 Compilador

O compilador gera uma árvore de sintaxe, ou parte dela, a partir de arquivos fonte textuais. A técnica utilizada para compilação pode ser caracterizada como compilação distribuída orientada a objetos. Para tanto a gramática que descreve a linguagem deve pertencer ao tipo 2 – linguagens livres de contexto – e ser uma gramática  $LL(k)$ . Estas gramáticas permitem compilação de-cima-a-baixo por recorrência (AHO; SETHI; ULLMAN, 1986). A linguagem *AnimaTi* é descrita por este tipo de gramática.

A árvore de sintaxe é definida por um grande número de classes de objetos. Parte destas classes representa construtos da linguagem *AnimaTi* (Apêndice A). Outra parte representa classes que suportam coleções de objetos, tais como listas de atores, casos e segmentos. O fragmento de ALGORITMO 4.2 mostra o construto do ator `se...fim` com três casos em que cada caso contém um bloco de atores equivalente a um grupo de segmentos.

ALGORITMO 4.2: Exemplo com o construto do ator `se...fim`


---

```

se em 10% dos casos
  trabalha [10..20]ms
  fixa var1 := [10..12]
  salta para <m1> em 1% dos casos
  trabalha 2ms
  <<m1>>
  trabalha [10..20]ms
senão em 40% dos casos
  trabalha 50ms
senão
  trabalha 5ms
fim de se

```

---

Na árvore de sintaxe o construto do ator `se...fim` é um objeto composto por um ou mais casos em que cada caso forma um grupo de segmentos por sua vez composto por um ou mais segmentos. Cada segmento é composto por um grupo fechado de atores formado por um ator complexo (ator que contém blocos) ou por um ou mais atores simples em que os atores internos só recebem e repassam o controle para seus atores adjacentes.

A compilação distribuída orientada a objetos pode ser entendida pelo exemplo acima. O compilador do ator `se...fim` é acionado pelo compilador de segmentos para reconhecer a parte do fonte que descreve o ator em questão. Após reconhecer o primeiro caso, cria um grupo de segmentos e aciona seu compilador. Em seguida reconhece os demais casos repetindo o procedimento acima. Ao reconhecer a cláusula `fim de se` pára. Em resumo, a frase do construto foi reconhecida e convertida em árvore (parcial) de sintaxe.

Voltando ao exemplo, o primeiro caso é composto por três grupos fechados de atores: a) formado pelas linhas 2, 3 e 4, que termina com o ator `salta`; b) formado pela linha 5; e c) formado pelas linhas 6 e 7, que inicia com o marcador `<<m1>>`. O compilador do grupo de segmentos reconhece o primeiro grupo de atores, cria um segmento e aciona seu compilador. Em seguida reconhece os demais grupos, repetindo o procedimento acima. Ao reconhecer a cláusula `senão` pára. O compilador do

segmento por sua vez reconhece o primeiro ator, cria o ator e aciona seu compilador. Em seguida reconhece os demais atores, repetindo o procedimento acima. Ao reconhecer a cláusula `senão pára`. Supondo que um dos atores seja o ator `se...fim`, verifica-se o acionamento recorrente dos compiladores, construindo a árvore de sintaxe.

Em síntese, a compilação distribuída orientada a objetos parte do princípio de que cada construto conhece sua sintaxe e implementa um compilador que trata sua parte do arquivo fonte. O analisador léxico associado ao arquivo fonte desmonta este em lexemas (nome, valores, símbolos) e fornece estes lexemas aos compiladores sob demanda. Em alguns casos os compiladores necessitam "olhar para frente", tomar decisões e eventualmente devolver lexemas ao analisador léxico. A gramática é do tipo LL(4) em que "olhar para frente" está limitado em 4 lexemas. Além disso, o analisador léxico realiza a tradução dos lexemas de inglês para português.

Como o experimento pode ser descrito por mais de um arquivo fonte, e alguns arquivos podem ser fontes XML, torna-se necessário ativar o carregador XML para estes arquivos.

#### **4.2.2.2 Gerador reverso**

O gerador reverso textual percorre a árvore de sintaxe e produz arquivos fonte textuais de acordo com a gramática da linguagem *AnimaTi* (Apêndice A). Parâmetros orientam o gerador e são definidos através de um painel de configuração.

#### **4.2.2.3 Gerador reverso XML**

O gerador reverso XML percorre a árvore de sintaxe e produz arquivos fonte de acordo com a descrição formal dos construtos da linguagem *AnimaTi* para XML (Apêndice B). Parâmetros orientam o gerador e são definidos através de um painel de configuração.

#### 4.2.2.4 Carregador XML

Arquivos fonte XML são considerados sintaticamente corretos e não necessitam de compilação para convertê-los em árvore de sintaxe. O carregador de arquivos fonte XML monta uma árvore de sintaxe, ou parte dela, e analisa sua consistência.

Como o experimento pode ser descrito por mais de um arquivo fonte, e alguns arquivos podem ser fontes textuais, torna-se necessário ativar o compilador para estes arquivos.

#### 4.2.2.5 Visor de fatos de análise

O visor de fatos de análise serve como interface com o usuário para reportar erros, alertas e avisos, que podem ocorrer nos diversos módulos da ferramenta.

A FIGURA 4.1 mostra duas versões do visor de fatos de análise. Apresenta 5 colunas: a) G, que indica através de ícones o grau de severidade do fato; b) Arquivo, que indica o nome do arquivo fonte textual ou XML ou da planilha de tempos a que o fato se refere; c,d) Linha e Col(una) opcionais indicam a posição relativa ao arquivo em que o fato ocorreu; e e) Mensagem, que descreve o fato. Cada tipo de mensagens possui um código que relaciona a mesma com seu gerador.



FIGURA 4.5: Visores de fatos de análise

Cada fato de análise é um objeto com dados suficientes para acionar os editores de arquivos fonte e planilhas de tempos, para abrir o arquivo relacionado com o fato e posicionar o cursor no ponto onde o fato ocorreu.

#### 4.2.2.6 Árvore de sintaxe

A árvore de sintaxe (AS) é formada pela compilação de arquivos fonte textuais e pela carga de arquivos fonte XML. A FIGURA 4.6 mostra a estrutura básica das classes de objetos que compõem a árvore de sintaxe. A classe UnidadeAS reúne os elementos descritos na Seção 4.2.1.3 através das coleções ArquivosFonte, Encontros, Processadores, Recursos, Rotinas e Tarefas. As classes da hierarquia Elemento, ElementoTRE e ElementoTR organizam as características comuns aos elementos e em especial as características dos elementos Rotina e Tarefa, que descrevem os algoritmos.

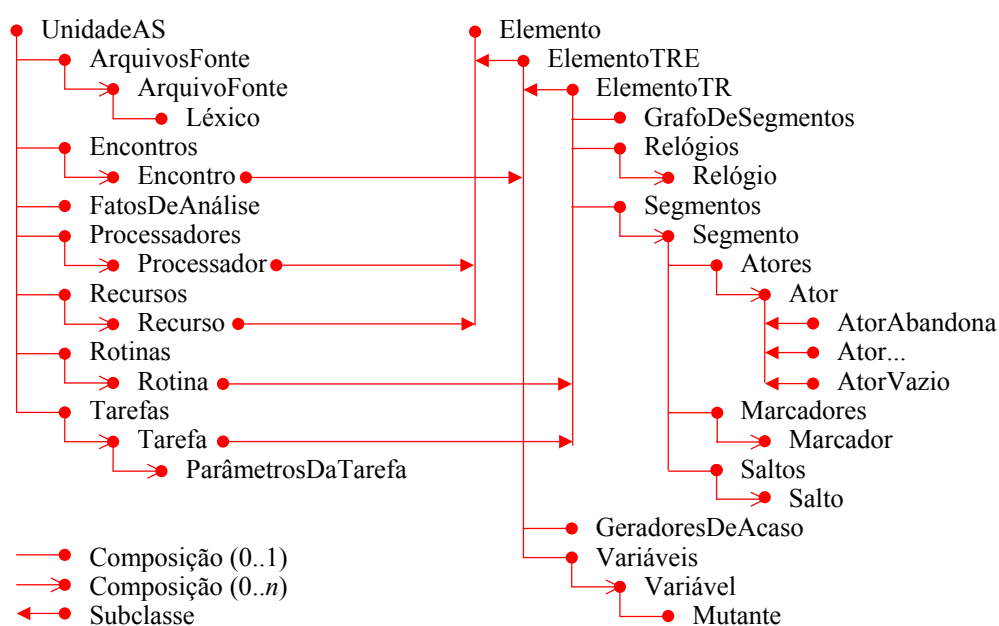


FIGURA 4.6: Classes básicas da árvore de sintaxe

A classe Segmentos reúne um ou mais segmentos pertencentes a um mesmo bloco do algoritmo. A passagem do controle entre os diversos segmentos é regida pela semântica dos atores complexos. A classe Segmento reúne três coleções: Atores, Marcadores e Saltos. O ator `salta` não figura na lista de atores do segmento, e sim, na lista de saltos.

A Seção 4.2.2.1 define o conceito de grupo fechado de atores associado a um segmento. Portanto, o segmento recebe o controle de três formas: a) por ser o primeiro segmento do grupo; b) diretamente do segmento anterior do mesmo grupo; e c) por salto para um de seus marcadores. Além disso, o segmento repassa o controle de quatro formas: a) para outro grupo de segmentos; b) para o próximo segmento do mesmo grupo; c) por salto para um marcador; e d) para outro grupo de segmentos por eventos ocorridos no último ator do segmento, tais como ocorrências de faltas e escapes em repetições.

A FIGURA 4.7 mostra a estrutura de classes do ator complexo `se...fim`, classe `AtorSe` e da classe `Mutante` com as subclasses. Vale observar que o ator é composto por um ou mais casos (classe `CasosSe`) e que cada caso (classe `CasoSe`) possui um objeto da classe `Segmentos`. A classe `Segmentos` forma a recorrência básica na árvore de sintaxe (ou árvore de segmentos).

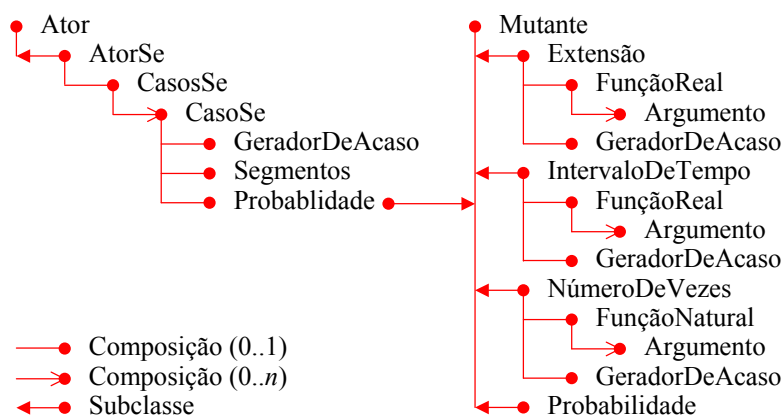


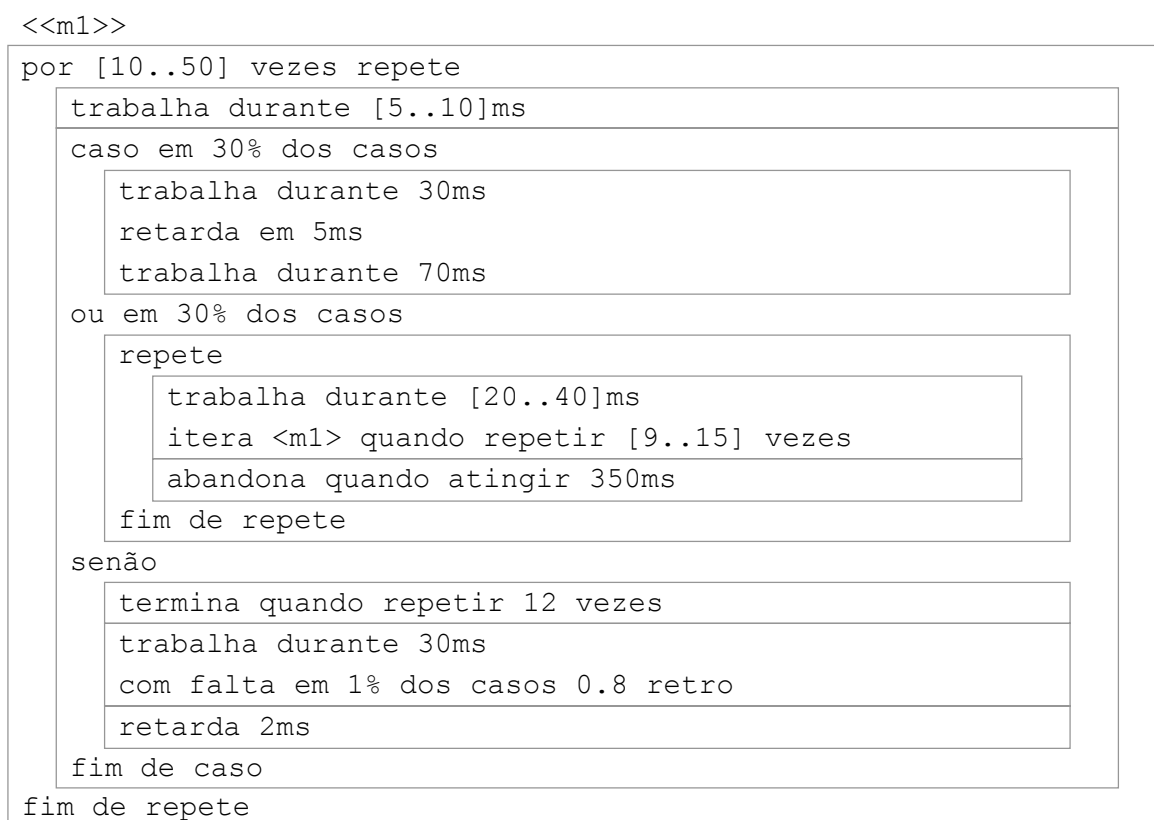
FIGURA 4.7: Classes do ator `se...fim` e dos mutantes

Todos os detalhes da árvore de sintaxe perfazem em torno de 100 classes de objetos e mais de 25 tipos de dados (enumerações e conjuntos).

#### 4.2.2.7 Exemplo de segmentação

O ALGORITMO 4.3 mostra um fragmento de fonte, ressaltando sua divisão em segmentos. O ator `repete...fim` mais externo é um segmento formado por um ator complexo com um marcador associado. O bloco interno deste ator é dividido em dois segmentos, sendo o primeiro formado pelo ator `trabalha` e o segundo pelo ator complexo `caso` (motivador da divisão), que por sua vez é composto por três casos.

ALGORITMO 4.3: Exemplo de fonte com divisão em segmentos



A FIGURA 4.8 mostra uma versão simplificada da árvore de sintaxe para o ALGORITMO 4.3 em que os nomes das classes figuram como objetos. A figura não mostra os objetos das classes Segmentos, CasosCaso, Atores e Ator.

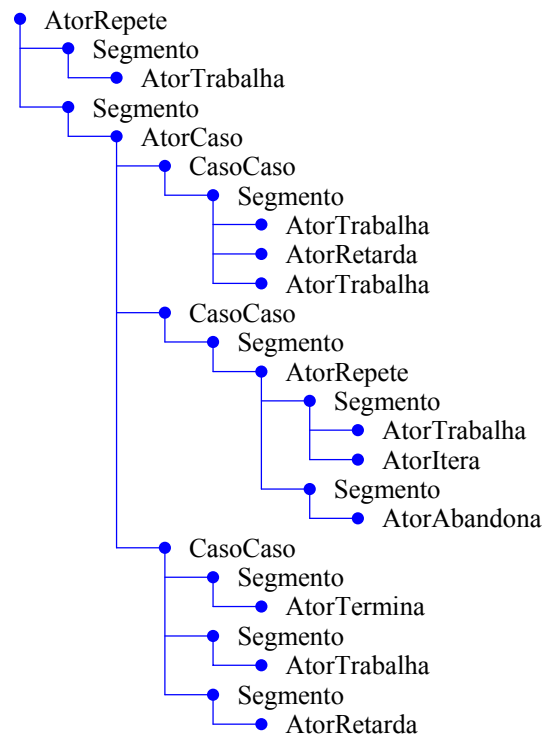


FIGURA 4.8: Árvore de sintaxe simplificada para o ALGORITMO 4.3

Vale observar o bloco do terceiro caso do ator `caso`, que está dividido em três segmentos. No primeiro segmento, o ator `termina` pode repassar o controle (dependendo da guarda) para o final do corpo em que está contido, portanto deve ser o último ator deste segmento. No segundo segmento, o ator `trabalha` pode gerar uma falta (pela cláusula `com`) e repassar o controle para um ponto de captura de faltas em algum bloco envolvente, portanto deve ser o último ator deste segmento. Finalmente, o terceiro segmento é formado pelo último ator do bloco.



### 4.2.3 Módulo de tradução de árvores de sintaxe

O objetivo principal do módulo de tradução de árvores de sintaxe é gerar grafos de segmentos a partir de árvores de sintaxe e a partir destes grafos de segmentos gerar planilhas de tempos e diagramas visuais de grafos de segmentos.

O módulo possui dois componentes de acordo com a FIGURA 4.1: a) o gerador de grafos de segmentos; e b) o gerador de planilhas de tempos. Nas próximas seções descreve-se cada componente.

#### 4.2.3.1 Gerador de grafos de segmentos

O gerador de grafos de segmentos gera um grafo (dirigido) de segmentos para cada elemento das classes Tarefa e Rotina (FIGURA 4.6) a partir de suas árvores (parciais) de sintaxe. A árvore de sintaxe representa os algoritmos de tarefa e rotinas segmentados de acordo com a descrição da Seção 4.2.2.6. O grafo de segmentos é formado por nodos que representam os segmentos e arcos (de saída) que representam o fluxo de controle entre segmentos (GÓES, 2001).

Os nodos preservam a estrutura dos construtos da linguagem *AnimaTi*, portanto estão divididos em dois grupos: a) nodos associados a segmentos de atores complexos; e b) nodos associados a segmentos de atores simples. No primeiro grupo de nodos cada ator complexo é representado por um par de nodos (esquema de nodos) com um nodo de entrada e outro de nodo de saída. O nodo de entrada direciona o fluxo de controle para os segmentos internos ao ator. O nodo de saída reúne o fluxo de controle dos segmentos internos ao ator.

No segundo grupo de nodos cada segmento é representado por um único nodo (esquema de nodos) simples que recebe o controle e repassa o controle para um ou mais nodos de acordo com os saltos associados ao segmento e de fenômenos gerados no último ator do segmento, tais como faltas geradas no ator `trabalha` e escape gerado pelo ator `abandona`. A FIGURA 4.9 mostra exemplos de esquemas de nodos.

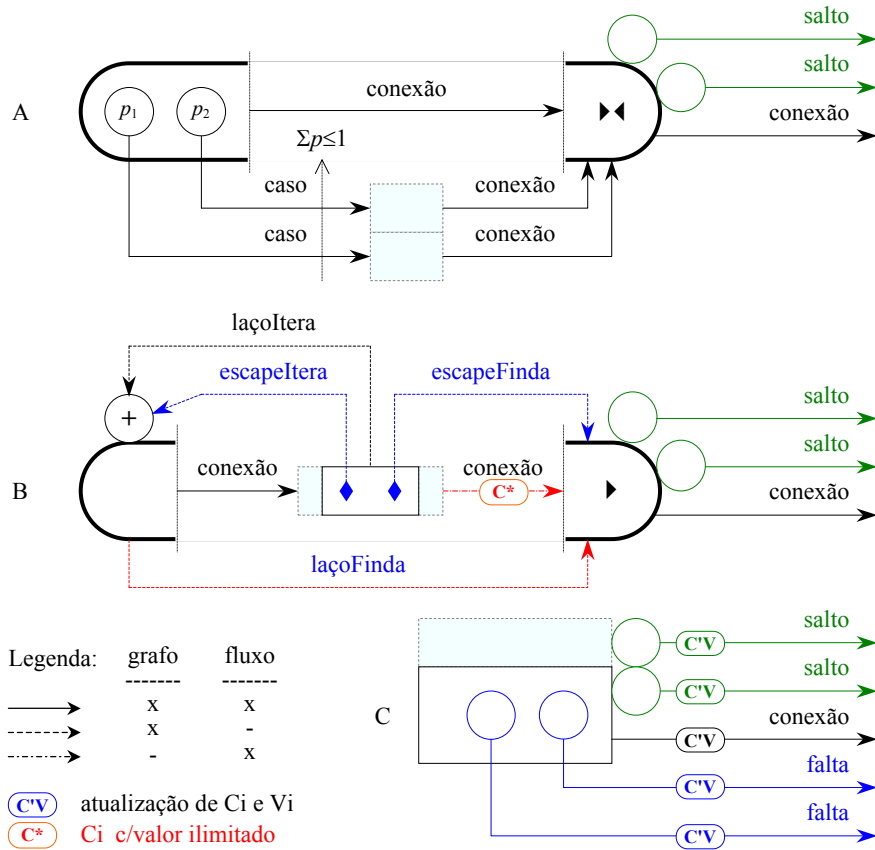


FIGURA 4.9: Exemplos de esquemas de nodos

O esquema de nodos A representa a estrutura do ator complexo *caso...fim*. O nodo de entrada (a esquerda) divide o fluxo de controle em três casos: dois casos de acordo com as probabilidades  $p_1$  e  $p_2$  e um terceiro caso (vazio) devido a ausência da cláusula *senão*. O nodo de saída (à direita) reúne os fluxos de controle dos casos. Os casos não vazios contêm por sua vez subgrafos (área azulada).

De modo semelhante, o esquema de nodos B representa a estrutura do ator complexo *repete...fim*. O nodo de entrada (à esquerda) decide sobre as iterações do laço. O nodo de saída (à direita) reúne os fluxos de controle ao término do laço. O subgrafo sujeito ao laço de repetições pode ser abandonado implicitamente, ou explicitamente pelo ator *abandona*, ou reiterado implicitamente ou reiterado pelo ator *itera*.

O esquema de nodos C representa um segmento que termina com ator *trabalha*. Este ator pode gerar faltas, e em função destas o grafo direciona o fluxo

de controle para os nodos que capturam estas faltas (ou para o nodo final do algoritmo).

O gerador de grafos de segmentos analisa os componentes da árvore de sintaxe, tais como segmentos, marcadores, saltos, faltas, atores de escape e atores em geral, para determinar nodos e arcos, com seus tipos. Na Seção 4.2.3.3 entra-se em mais detalhes a respeito do grafo de segmentos, tipos de nodos e tipos de arcos.

#### 4.2.3.2 Gerador de planilhas de tempos

O gerador de planilhas de tempos percorre os grafos de segmentos pelos arcos de fluxo a fim de coletar valores para o prior caso, caso típico e melhor caso de tempos, WCxT, TCxT e BCxT, respectivamente. Coleta tempos para três valores: tempo de execução  $C_i$ , tempo de suspensão voluntária  $V_i$  e tempo de resposta  $R_i$ . Além disso determina as seções críticas de recursos e coleta seus tempos  $C_i$  e  $V_i$ . Os tempos coletados são anotados na planilha de tempos (WTB) juntamente com valores extraídos dos parâmetros das tarefas, tais como período, prioridade e prazo. Na Seção 4.2.4.5 descreve-se as planilhas de tempos.

A FIGURA 4.10 mostra um fragmento de grafo de segmentos formado pelos nodos N1 a N6. A partir deste elucida-se o processo de coleta de tempos através da passagem de fichas entre os nodos, percorrendo todos os caminhos possíveis.

Cada nodo está associado a uma ficha, contendo as listas das variáveis e dos relógios usados pela tarefa e seus valores, bem como os valores de  $C_i$ ,  $V_i$  e  $R_i$  naquele instante. Cada arco está associado a uma ficha contendo as parcelas dos valores  $C_i$  e  $V_i$  determinadas pela execução dos algoritmos associados aos nodos. Nodos e arcos podem ser inviáveis (ou inatingíveis), por exemplo, o nodo N3 e o arco N3-N4.

Supondo, por simplicidade, que os nodos N1, N2 e N3 tenham sido visitados por todos os seus arcos aferentes, então cada um destes nodos possui sua ficha e as fichas em seus arcos eferentes atualizadas. Portanto, podem visitar seus nodos adjacentes, no caso o nodo N4, percorrendo os arcos eferentes, levando suas fichas e as fichas dos respectivos arcos.

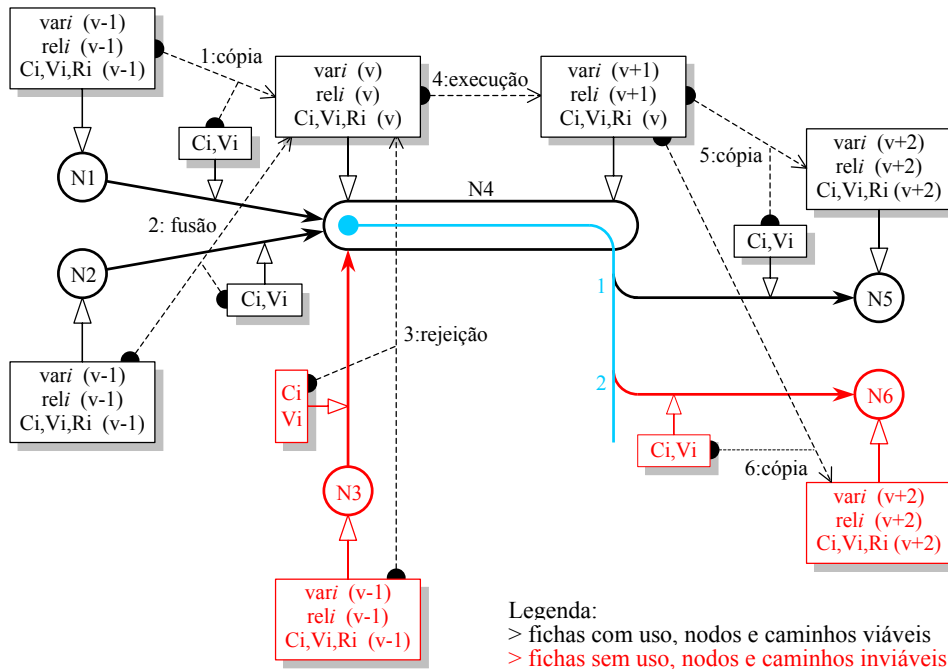


FIGURA 4.10: Processo de passagem de fichas

O nodo N4 monta sua ficha inicial versão (v) através de três operações: a) cópia, em que os valores da ficha do visitante são atualizados com os valores da ficha do arco de visita e copiados para a ficha vazia própria; b) fusão, em que os valores da ficha do visitante são atualizados com os valores da ficha do arco de visita e confrontados com os valores da ficha própria para preservação do pior e melhor caso, com cálculo do caso típico; e c) rejeição, em que a fichas do visitante e do arco de visita não são utilizadas.

Supondo que o nodo N4 ainda não tenha sido visitado, então a visita a partir do nodo N1 através do arco N1–N4 implica na operação de cópia. Em seguida, a visita a partir do nodo N2 através do arco N2–N4 implica na operação de fusão. Finalmente, a visita a partir do nodo N3 através do arco N3–N4 implica na operação de rejeição.

Agora o nodo N4 pode visitar os nodos N5 e N6. Primeiramente deve executar o algoritmo associado ao segmento do nodo a fim de atualizar sua ficha e gerar as fichas de seus arcos eferentes e julgar a viabilidade dos arcos. No caso, o arco N4–N6 tornou-se inviável por suposição. Os valores das fichas dos arcos podem diferir, por exemplo, pelo tratamento de faltas no ator `trabalha` em que o tempo  $C_i$  até a ocorrência da falta é menor que o tempo  $C_i$  normal previsto. Em seguida visita o nodo

N5 através do arco N4–N5 e o nodo N6 através do arco N4–N6 (inviável). As fichas iniciais dos nodos N5 e N6 são geradas por cópia. O nodo N6 torna-se inviável devido à inviabilidade do arco N4–N6.

As seções críticas e seus tempos  $C_i$  e  $V_i$  são determinadas a partir dos valores correspondentes das fichas dos arcos. As seções críticas (FIGURA 3.13) podem gerar configurações: a) em que a seção crítica de um recurso engloba seções críticas de outros recursos; ou b) em que a seção crítica de um recurso se sobrepõe a seções críticas de outros recursos. O segundo caso constitui-se numa restrição para o gerador de planilhas de tempos.

O gerador de planilhas de tempos requer: a) grafos acíclicos do ponto de vista dos arcos de fluxo; b) grafos que não contenham arcos de fluxo com valores  $C_i$  e/ou  $V_i$  ilimitados, como mostra o arco de conexão sobreposto pelo símbolo  $C^*$  na FIGURA 4.9B; e c) grafos isentos de atores que tratam encontros.

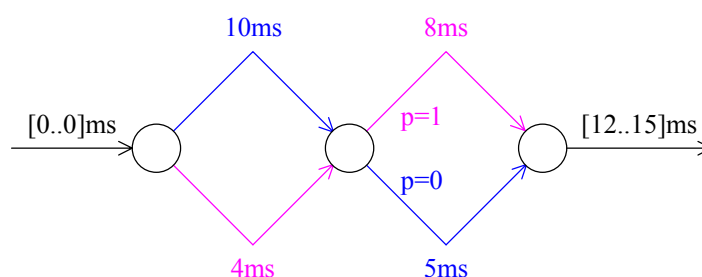
O gerador de planilhas de tempos detecta e elimina pessimismos e otimismo da forma apresentada pelo ALGORITMO 4.4 em que os piores e melhores casos de tempos não refletem a natureza temporal do algoritmo.

#### ALGORITMO 4.4: Exemplo de tempos pessimistas/otimistas

```

se em 50% dos casos
  trabalha 10ms
  fixa p := 0
senão
  trabalha 4ms
  fixa p := 1
fim
se em p dos casos
  trabalha 8ms
senão
  trabalha 5ms
fim

```



caminhos mutuamente exclusivos

No exemplo, o pior caso de tempo de execução é igual a 18ms. No entanto, considerando o efeito de probabilidade da variável  $p$ , que divide o fluxo em dois

caminhos mutuamente exclusivos, o pior caso é igual a 15ms. O mesmo raciocínio aplica-se ao melhor caso de tempo de execução, que passa de 9ms para 12ms.

#### 4.2.3.3 Grafo de segmentos

Grafos de segmentos (GS) são representados por objetos das classes apresentadas na FIGURA 4.11. Cada objeto da classe ElementoTR, Tarefas e Rotinas, está associado a um grafo de segmentos da classe GrafoDeSegmentos formado por nodos de segmentos da classe NodoSegmento, que representam os segmentos. Cada nodo possui arcos de saída da classe ArcoDeSaída, que representam o fluxo de controle entre nodos de segmentos. Além disso, cada nodo possui uma ficha da classe FichaWTB para anotar os valores de variáveis (classe VariávelWTB), relógios (classe RelógioWTB) e tempos  $C_i$ ,  $V_i$  e  $R_i$  pelo gerador de planilhas de tempos (Seção 4.2.3.2).

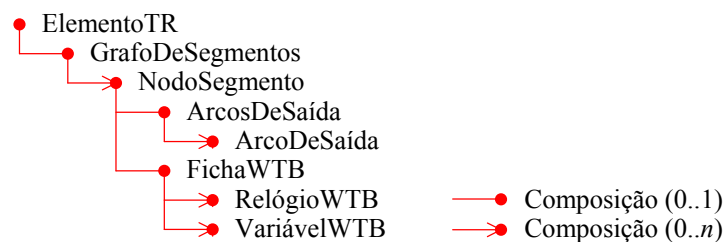


FIGURA 4.11: Classes do grafo de segmentos

Os arcos de saída são classificados de acordo com seus papéis nos nodos em arcos do tipo: caso, casoSalta, conexão, escapeFinda, escapelitera, falta, laçoFinda, laçoItera, salto e término. Além disso possuem 4 indicações opcionais: a) fluxo, para arcos pertencentes ao fluxo de controle para o gerador de planilhas de tempos; b) grafo, para arcos pertencentes ao fluxo de controle para a simulação dos algoritmos; c) infinito, para arcos que levam a situações de tempos ilimitados ou para o tempo de execução  $C_i$  ou para o tempo de suspensão  $V_i$ ; e d) viável, para arcos que participam

do fluxo de controle efetivamente. Arco infinito para o tempo de execução  $C_i$  ocorre, por exemplo, quando o número de iterações do ator `repete...fim` não está limitado e seu nodo de saída não possui arcos aferentes do tipo `escapeFinda` ou estes arcos são inviáveis. A espera por encontros ou recursos sem cláusula `com` torna o arco infinito para o tempo de suspensão  $V_i$ .

A indicação de fluxo e grafo nos arcos permite interpretar o grafo de segmentos sob dois pontos de vista: a) geração da planilha de tempos, que percorre os arcos de fluxo; e b) simulação dos algoritmos, que percorre os arcos de grafo. Exemplificando pela FIGURA 4.9B, o percurso dos arcos marcados com conexão e salto forma uma linha reta, e os tempos  $C_i$ ,  $V_i$  dos segmentos internos são multiplicados por um fator de repetição do ator. Todos os arcos, exceto o arco entre os segmentos internos e o nodo de saída, são percorridos pela simulação dos algoritmos. No Apêndice E relaciona-se os diversos tipos de nodos associados a grupos de atores. Cada tipo de nodo relaciona os tipos de arcos de saída associados, indicando a situação em que estes ocorrem.

#### 4.2.3.4 Diagrama de segmentos

O gerador de grafos de segmentos gera diagramas visuais, mostrando os nodos e os arcos dos grafos de segmentos através de figuras. Segmentos, atores, marcadores, saltos e demais adornos estão associados a figuras distintas para facilitar a sua identificação visual. O diagrama guarda semelhança com a estrutura representada pelos esquemas de nodos exemplificados na FIGURA 4.9.

A FIGURA 4.12 mostra o diagrama de segmentos referente ao ALGORITMO 4.3 inserido no corpo de uma tarefa.

No Apêndice C mostra-se a coleção de figuras desenhadas para cada elemento visual da linguagem *AnimaTi*. A utilização de cores é arbitrária e serve para ressaltar certas propriedades como tipo do arco e situação do arco. Arcos em vermelho, por exemplo, são arcos inviáveis.

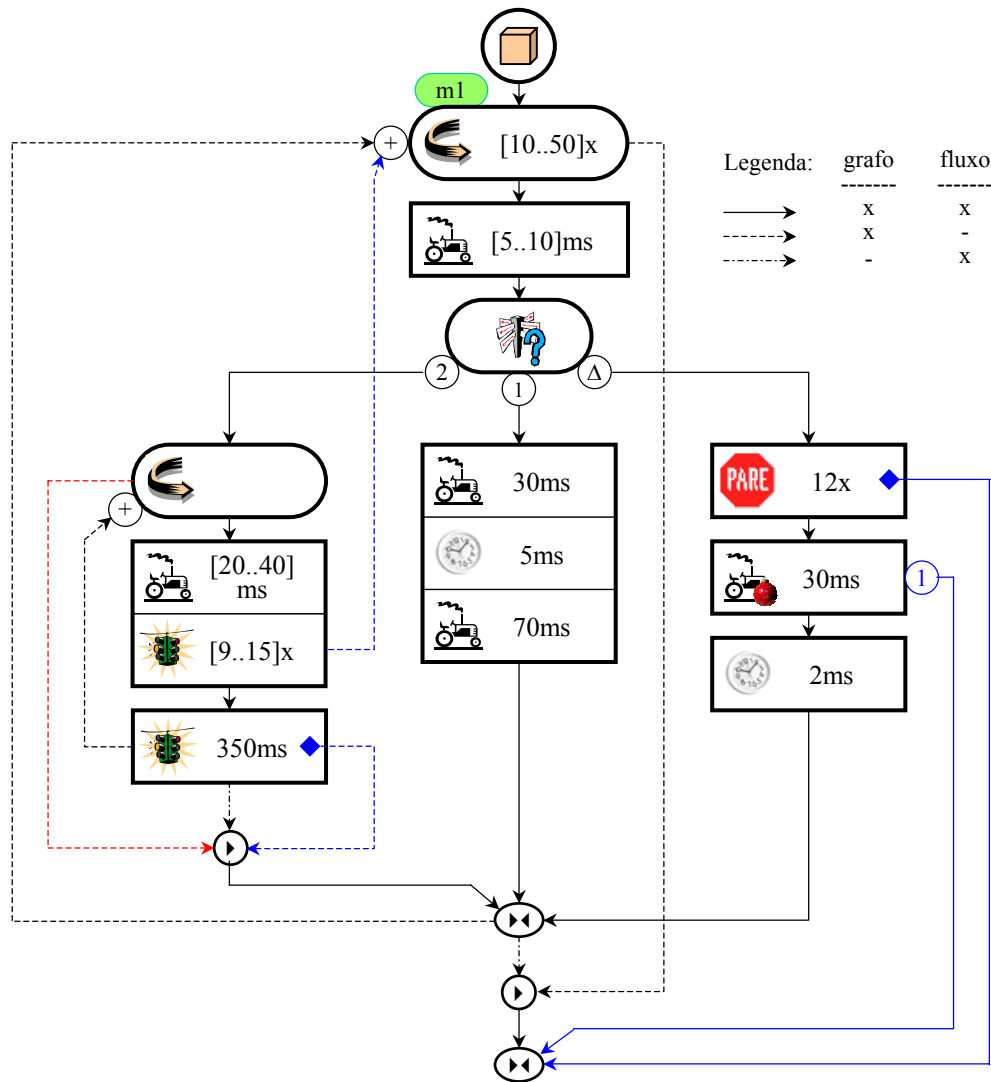


FIGURA 4.12: Diagrama de segmentos do ALGORITMO 4.3

#### 4.2.4 Módulo de análise e simulação de escalonamento

O objetivo principal do módulo de análise e simulação de escalonamento é aplicar analiticamente a teoria clássica de escalonamento sobre conjuntos de tarefas descritas em planilhas de tempos e simular o escalonamento passo a passo com auxílio de um gráfico de tempos (Gantt).

A base para a análise e simulação de escalonamento são as planilhas de tempos geradas a partir de algoritmos escritos na linguagem *AnimaTi* (Seção 4.2.3.2) ou geradas pelo editor de planilhas de tempo. As planilhas de tempos podem ser armazenadas em arquivos XML, facilitando a guarda e intercâmbio de experimentos.



O módulo possui três componentes de acordo com a FIGURA 4.1: a) o editor de planilhas de tempos; b) o analisador de escalonamento; e c) o simulador de escalonamento. Os três componentes interagem dinamicamente. Nas próximas seções descreve-se cada componente.

#### 4.2.4.1 Editor de planilhas de tempos

O editor de planilhas de tempos está dividido em dois elementos funcionais: a) o visor de planilhas de tempos, que mostra as tarefas e suas propriedades arranjadas em linhas e colunas; e b) o editor de propriedades de tarefas.

A FIGURA 4.13 mostra o visor de planilhas de tempos. A parte central lista as tarefas e suas propriedades através de um arranjo em linhas e colunas. A coluna S da planilha indica o estado das tarefas, normal, descartada e inviável (como resultado da análise) através de ícones.

S	Tarefa	Prio	Pi	Ti	Di	Ci	Vi	Ji	Oi	Bi	?Bi	?Ti	?Ri	?ri	?q	Processador	Modo
✓	<tarefa-1>	0	2	100,0000 ms	100,0000 ms	40,0000 ms	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0	p1	P
✓	<tarefa-2>	0	3	200,0000 ms	200,0000 ms	30,0000 ms	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0	p1	P
✓	<tarefa-3>	0	1	350,0000 ms	370,0000 ms	150,0000 ms	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0	p1	P
✗	<tarefa-4>	0	2	100,0000 ms	100,0000 ms	30,0000 ms	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0	p1	P
✗	<tarefa-5>	0	3	200,0000 ms	200,0000 ms	40,0000 ms	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0	p1	P
✗	<tarefa-6>	0	1	350,0000 ms	300,0000 ms	150,0000 ms	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0 s	0	p1	P

Totais: 6 tarefa(s), 3 descartada(s), 0 inviável(is)

FIGURA 4.13: Visor da planilha de tempos

As funcionalidades disponibilizadas pelo visor de planilhas de tempos incluem: a) criação e manutenção de planilhas; b) manutenção da lista de tarefas; c) ordenação por tarefas e colunas de propriedades; d) habilitação da análise e simulação; e e) descarte de tarefas.

As propriedades de tarefas são divididas em dois grupos: a) as propriedades informadas (Seção 4.2.4.5); e b) as propriedades calculadas determinadas pela análise

de escalonamento (Seção 4.2.4.2). A FIGURA 4.14 mostra o editor de tarefa que permite modificar as propriedades informadas.

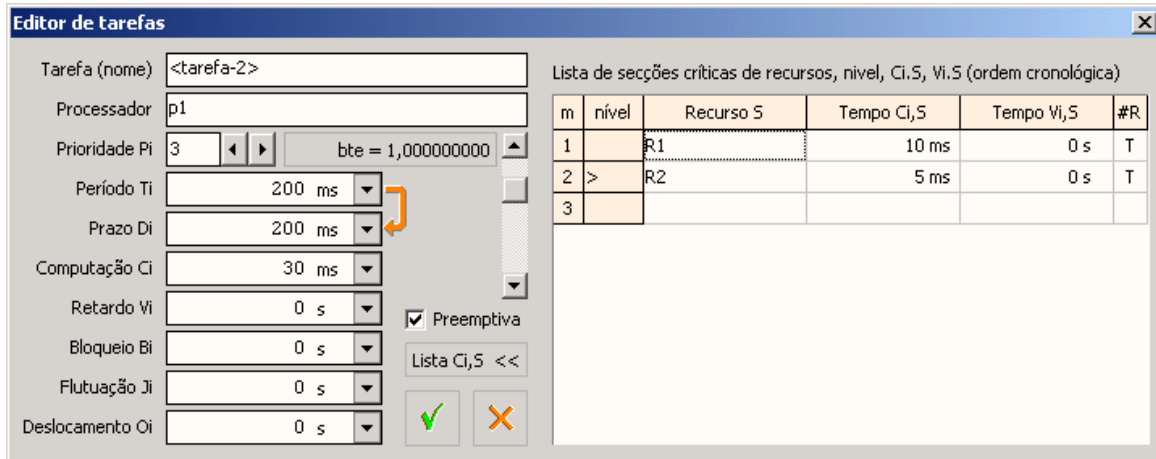


FIGURA 4.14: Editor de tarefas da planilha de tempos

As funcionalidades disponibilizadas pelo editor de tarefas através de um menu de topo incluem: a) edição das propriedades de tarefas; e b) manutenção da lista de secções críticas.

O analisador e simulador de escalonamento operam com as planilhas de tempos seleccionadas pelo editor e refletem os resultados para as propriedades calculadas.

#### 4.2.4.2 Analisador de escalonamento

O analisador de escalonamento considera as tarefas da planilha de tempos seleccionada pelo editor de planilhas de tempos e aplica as equações da teoria clássica de escalonamento (Seção 3.4). Além das propriedades de tarefas, é necessário estabelecer critérios adicionais sobre o ambiente, tais como modo de prioridade e protocolo de escalonamento.

A FIGURA 4.15A mostra o visor com cinco grupos de parâmetros: a) forma de estabelecer prioridades; b) protocolo de escalonamento; c) opções para o ambiente; d) lista de processadores utilizados nas tarefas em análise (FIGURA 4.15B); e e) lista de recursos utilizados nas tarefas em análise (FIGURA 4.15C).

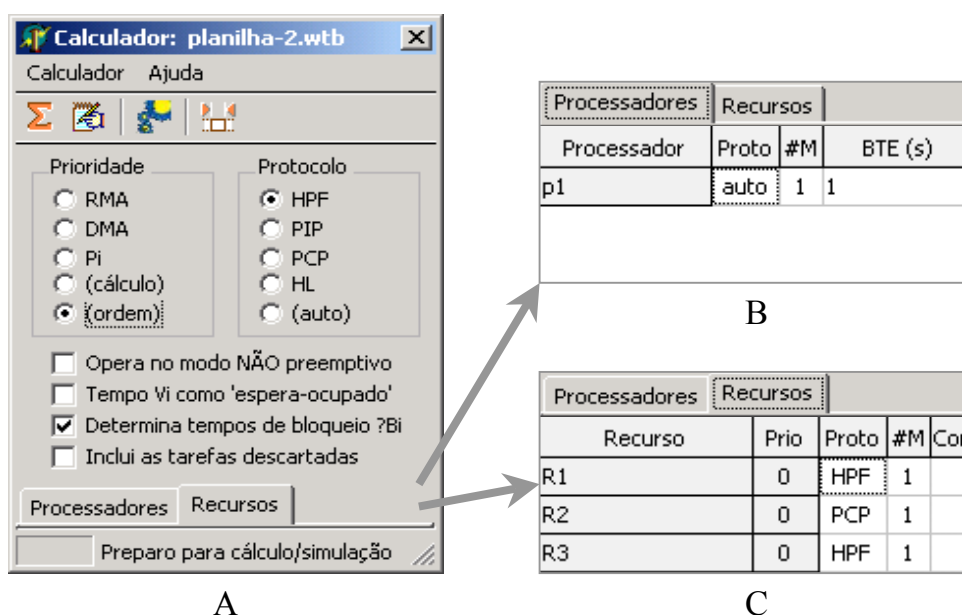


FIGURA 4.15: Visor de parâmetros para análise e simulação de escalonamento

A Seção 4.2.4.5 entra em detalhes a respeito dos valores dos parâmetros. Os problemas que ocorrem durante a análise são reportados através do visor de fatos de análise (Seção 4.2.2.5). A coluna "Cor" da FIGURA 4.15C estabelece a associação visual entre o nome do recurso e as ocorrências das seções críticas no gráfico de tempos (Seção 4.2.4.3).

As funcionalidades disponibilizadas pelo analisador de escalonamento incluem: a) manutenção dos parâmetros; b) acionamento do analisador; e c) simulador.

O analisador de escalonamento mostra os valores obtidos analiticamente no visor de planilhas de tempos (FIGURA 4.13) nas colunas: a) tempo de bloqueio calculado  $?Bi$ ; b) tempo de interferência  $I_i$ ; c) tempo total de resposta  $R_i$ ; d) tempo de resposta  $r_i$ , excluindo flutuação e deslocamento; e e) número de serviços  $q_i$ , simultaneamente ativos para a mesma tarefa.

#### 4.2.4.3 Simulador de escalonamento

O simulador de escalonamento opera diretamente sobre um gráfico de tempos (Seção 4.2.4.6) com interpretação visual para linhas, serviços e estados. Além disso

suporta os marcadores (Seção 2.3.3) para indicar instantes de ativação e término, prazos, situações de decurso de prazo, prioridades nos estados, e o cursor de eventos. A FIGURA 4.16 mostra o visor de gráficos de tempos.

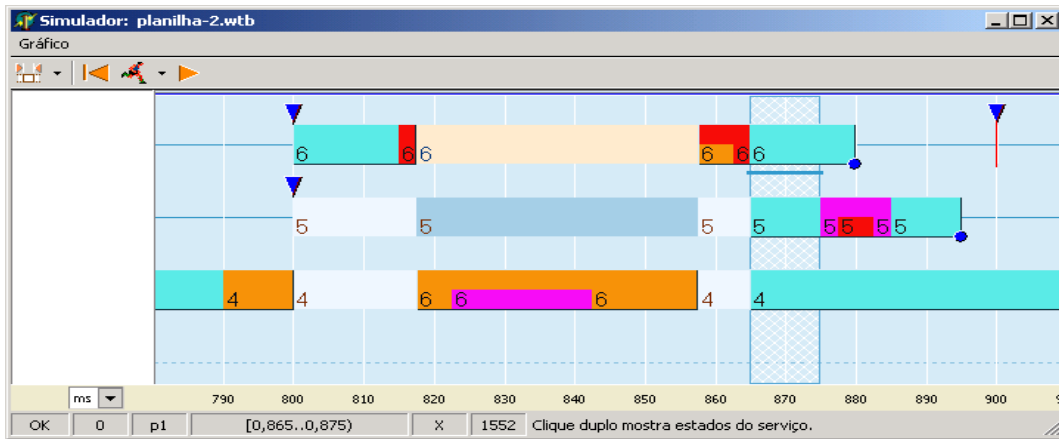


FIGURA 4.16: Visor de gráficos de tempos

No exemplo, o cursor de eventos está localizado em torno do instante 870ms e indica que o serviço mais acima deve receber o controle do processador por 10ms.

As funcionalidades disponibilizadas pelo simulador de escalonamento incluem:

- configuração de marcadores;
- distribuição manual e aleatória de regiões críticas nos serviços;
- distribuição manual e aleatória de tempos de suspensão voluntária  $V_i$ ;
- divisão do gráfico de tempos em diversas visões para comparação;
- execução automática e temporizada;
- execução passo a passo;
- fixação dos instantes de ativação dos primeiros serviços das tarefas;
- modificação da posição e escala do gráfico de tempos.

O conteúdo total ou parcial de gráficos de tempos pode ser registrado em diários de simulação e posteriormente reproduzido pelo animador de gráficos de tempos.

A simulação está sujeita aos mesmos parâmetros da análise fixados pelo visor de parâmetros (FIGURA 4.15). O ALGORITMO 4.5 mostra o funcionamento básico do simulador de escalonamento.

## ALGORITMO 4.5: Funcionamento básico do simulador de escalonamento

---

```
repete
  busca o próximo evento
  abandona quando não encontrou um evento
  se encontrou um evento de ativação de serviços
    ativa o serviço no evento
  senão se existir, sob PIP ou PCP, um estado a ser bloqueado
    modifica a prioridade do estado que detém o recurso
  senão -- demais casos
    trava recursos requeridos pelo estado
    insere/estende estados de suspensão ou bloqueio
    libera recursos disponibilizados pelo estado
  fim de se
fim de repete
```

---

A operação "busca o próximo evento" determina o horizonte de eventos mais próximo dentre os horizontes de eventos das linhas de controle. Seleciona então os estados dos serviços com este horizonte. Dentre estes estados seleciona aquele com a maior prioridade e que não requer algum recurso já travado. Este último estado está associado ao evento. Um quase-serviço marca o instante de ativação do próximo serviço da tarefa e possui um estado com prioridade máxima. O resultado da operação determina quatro casos:

- a) não encontrou um evento: neste caso ocorreu um impasse de bloqueio e o simulador pára;
- b) encontrou um evento associado (indiretamente) a um quase-serviço: neste caso ativa o próximo serviço e reposiciona o quase-serviço no instante de ativação subsequente;
- c) sob o protocolo PIP ou PCP, encontrou um evento, porém, no mesmo horizonte do evento existe pelo menos um estado que requer algum recurso já travado: neste caso estabelece nova prioridade para o estado que detém a trava deste recurso de acordo com o protocolo;
- d) encontrou um evento em condições diferentes de a) e b): neste caso, trava os recursos requeridos pelo estado associado ao evento, insere/estende estados de

suspensão ou bloqueio e libera os recursos disponibilizados pelo estado associado ao evento.

Os estados são classificados em diversos tipos (Seção 4.2.4.6) interpretados por cores para facilidade de leitura. As cores e outros detalhes são explicados pelo visor de detalhes de serviços.

#### 4.2.4.4 Visor de detalhes de serviços

A partir do visor de gráficos de tempos pode-se acionar o visor de detalhes mostrado na FIGURA 4.17.

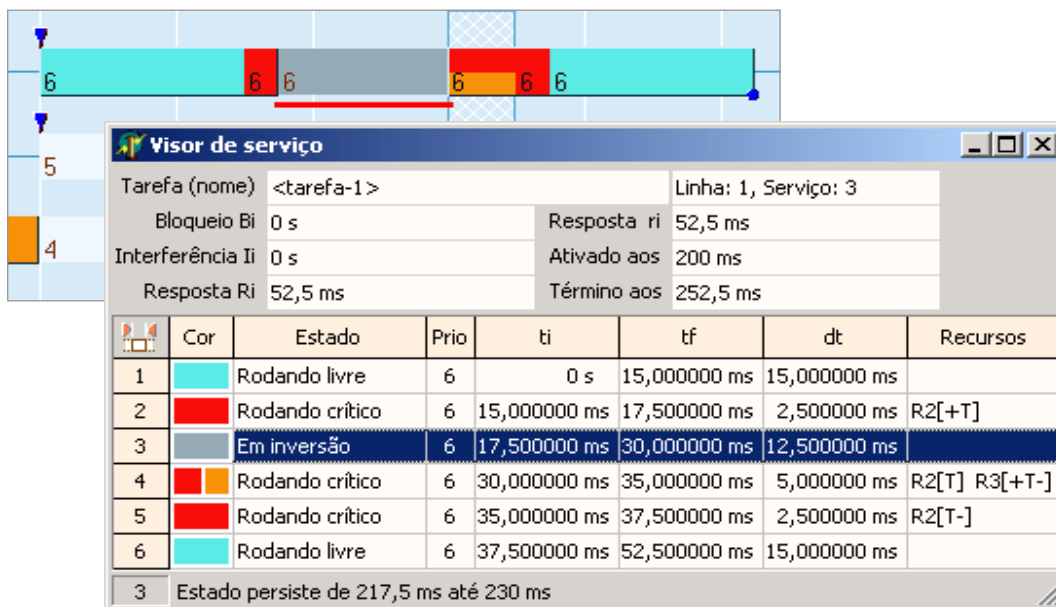


FIGURA 4.17: Visor de detalhes de serviços

O visor apresenta um resumo do serviço selecionado, indicando valores, tais como total de bloqueios, total de interferências e tempo de resposta.

Além disso mostra a lista de estados e valores associados, tais como tipo do estado, intervalo de ocorrência e duração. A coluna "Recursos" detalha as alocações e liberações de recursos. Por exemplo, no estado 4 o recurso R2 (cor vermelha) está bloqueado, e o recurso R3 (cor cáqui) é alocado na entrada do estado (+) e liberado na saída do estado (-).

O visor oferece funcionalidades para navegar ao longo dos estados a partir da lista de estados e do gráfico de tempos.

#### 4.2.4.5 Planilha de tempos

As planilhas de tempos registram os diversos dados a respeito de conjuntos de tarefas cuja escalonabilidade é analisada e simulada pelos respectivos componentes. Os dados podem ser extraídos das árvores de sintaxe pelo gerador de planilhas de tempos, bem como informados através do editor de planilhas de tempos. Além disso, as planilhas podem ser guardadas em arquivos XML para facilitar a memória dos diversos experimentos e intercâmbio estudante-estudante-professor.

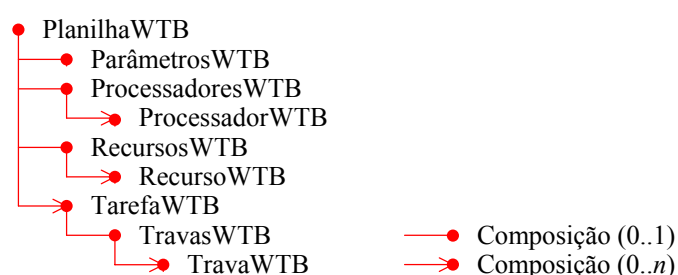


FIGURA 4.18: Esquema de classes da planilha de tempos

A FIGURA 4.18 mostra o esquema (simplificado) de classes que suportam estes dados. As planilhas guardam os parâmetros gerais para a análise e simulação e os dados dos processadores, dos recursos e das tarefas. Os dados das tarefas refletem as medidas de tempos apresentadas na Seção 2.4.

Os parâmetros determinam: a) o modo de atribuição de prioridades RMA, DMA,  $P_i$ , por cálculo e por ordem das tarefas na planilha; b) o protocolo de escalonamento HPF, PIP, PCP, HL e auto para considerar as opções dos processadores; c) o modo de operação preemptivo ou não-preemptivo; d) o tratamento dos tempos  $V_i$  por "espera-ocupado" ou por suspensão voluntária; e) o uso dos tempos  $B_i$  informados ou tempos

$B_i$  calculados a partir das seções críticas; e f) a inclusão das tarefas descartadas. A determinação da prioridade por cálculo segue o algoritmo da Seção 3.5.1.

Os processadores possuem as seguintes propriedades: a) nome; b) multiplicidade; considerando instâncias de processadores simétricos; c) base de tempo multiplicada pelo tempo de execução  $C_i$  para obter o tempo de execução efetivo; e d) protocolo de escalonamento do processador HPF, PIP, PCP, HL e auto para considerar as opções dos recursos.

Os recursos possuem as seguintes propriedades: a) nome; b) multiplicidade, indicando o número de instâncias para acesso total e parcial; e c) protocolo de escalonamento do recurso HPF, PIP, PCP e HL.

As tarefas possuem as seguintes propriedades: a) nome; b) prioridade informada  $P_i$  considerada no modo de atribuição de prioridades; c) período  $T_i$ ; d) prazo  $D_i$ ; e) tempo de execução  $C_i$ ; f) tempo de suspensão voluntária  $V_i$ ; g) flutuação  $J_i$ ; h) deslocamento  $O_i$ ; i) tempo de bloqueio informado  $B_i$ ; j) modo de operação preemptivo ou não-preemptivo; k) processador; e l) indicação de descartadas e/ou inviável. A ordem das tarefas tem importância no modo de atribuição de prioridades.

Além destas propriedades, as tarefas possuem zero ou mais seções críticas envolvendo recursos, com as seguintes propriedades: a) nome do recurso; b) nível de aninhamento; c) tempo de execução  $C_{i,S}$ ; d) tempo de suspensão voluntária  $V_{i,S}$ ; e e) modo de acesso, envolvendo todas as instâncias ou um número exato de instâncias. O aninhamento permite a ocorrência de seções críticas em árvore nos intervalos de tempos de outras seções críticas.

#### **4.2.4.6 Gráfico de tempos**

A simulação de escalonamento é acompanhada passo a passo através de gráficos de tempos (Gantt). Os gráficos podem ser gerados diretamente pelo simulador de escalonamento ou pelo animador de gráficos de tempos (Seção 4.2.5.2).

A FIGURA 4.19 mostra o esquema (simplificado) das classes que suportam a estrutura dos gráficos de tempos. A dinâmica da simulação acompanha os ciclos de



vida descritos nas Seção 2.3.1 e Seção 2.3.2. Portanto, a simulação gera um quadro com uma linha de controle para cada tarefa. Ao longo das linhas de controle são ativados os serviços das tarefas. Os serviços são decompostos em estados, e alguns estados controlam a alocação e liberação de recursos. O quadro possui, ainda, um cursor para controle de eventos.



FIGURA 4.19: Esquema de classes do gráfico de tempos

As linhas de controle são ordenadas de acordo com as prioridades das tarefas nas planilhas de tempos. Quando dois serviços da mesma tarefa estão ativos ao mesmo tempo, abre-se uma linha de controle adicional para a tarefa. Este fato decorre do tratamento de prazos arbitrários para  $q > 0$  (equação 3.23).

Uma das propriedades das linhas de controle define o horizonte de eventos para determinação do próximo evento. Propriedades dos serviços guardam o instante de ativação e o prazo. Propriedades dos estados guardam o instante de início e sua duração, a prioridade vigente e o tipo do estado. Os tipos de estados são listados na Seção 2.3.3, a saber: bloqueado, em deslocamento, em flutuação, em retardo, executando crítico, executando livre, suspenso por tarefa com prioridade maior, suspenso por tarefa com prioridade menor e suspenso por inversão de prioridade. Os estados do tipo "executando crítico" estão associados a recursos através de controles para determinar o instante de alocação e liberação destes recursos. Os estados do tipo "bloqueado" também estão associados a recursos para caracterizar o bloqueio. Propriedades dos eventos guardam o teto de prioridade do sistema (para o protocolo PCP), a duração do próximo passo e a situação do evento.

A FIGURA 4.20 mostra dois fragmentos de gráficos de tempos. No fragmento A observam-se três linhas de controle e o cursor de eventos estacionado sobre os três serviços em andamento. O serviço mais acima tem a maior prioridade, mas requer um recurso que está sendo bloqueado pelo serviço mais abaixo com a menor prioridade. Assim o segundo serviço é selecionado para receber o controle do processador.

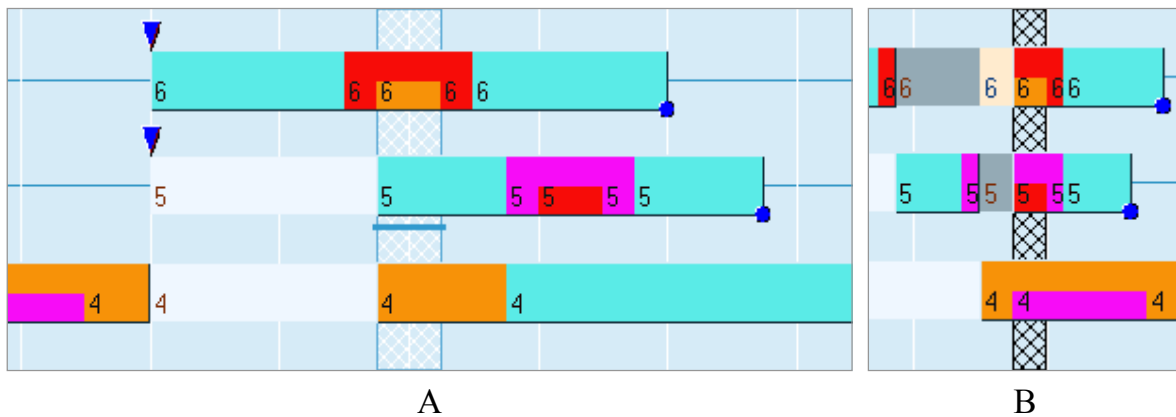


FIGURA 4.20: Exemplos de cursores

O fragmento B mostra o cursor de eventos na posição de impasse de bloqueio. De fato o serviço mais acima bloqueia o recurso em cor vermelha, o segundo serviço bloqueia o recurso em cor magenta e o serviço mais abaixo bloqueia o recurso em cor cáqui, constituindo-se uma cadeia circular de bloqueios. Nestas condições o simulador pára.

#### 4.2.4.7 Outros gráficos e saídas

A partir dos dados dos gráficos de tempos pode-se gerar outros gráficos e saídas, tais como o gráfico de distribuição da carga nos processadores, o gráfico de ocupação dos recursos, o diário de simulação e o extrato do gráfico de tempos.

#### **4.2.5 Módulo de simulação de tarefas e rotinas**

O objetivo principal do módulo de simulação de tarefas e rotinas é simular os algoritmos através de uma máquina virtual capaz de operar com diversos tipos de escalonadores.

O módulo possui dois componentes de acordo com a FIGURA 4.1: a) o simulador de tarefas e rotinas; e b) o animador de gráficos de tempos. As próximas seções descrevem estes componentes.

##### **4.2.5.1 Simulador de tarefas e rotinas**

O simulador de tarefas e rotinas implementa uma máquina virtual para executar o conjunto completo de atores definidos na linguagem *AnimaTi*. Além disso implementa um gestor de escalonamentos aberto que pode operar com diversas políticas de escalonamento, tais como FCFS, RR, EDF e RMS.

As funcionalidades disponibilizadas pelo simulador de tarefas e rotinas incluem: a) operação em ambientes multiprocessados simétricos e assimétricos; b) configuração de prioridades e ordem das tarefas; c) comunicação com o animador de gráficos de tempos para visualização imediata de resultados de tempos; e d) visualização do estado da simulação através o diagrama de segmentos.

Comparado com a análise e simulação de escalonamento, o simulador de tarefas e rotinas permite simular cenários com maior diversidade de algoritmos, tais como ativação dinâmica de tarefas, tratamento de encontros e grafos de segmentos com ciclos. Além disso, não é necessário estabelecer limites em repetições de ciclos nos algoritmos.

##### **4.2.5.2 Animador de gráficos de tempos**

O animador de gráficos de tempos interpreta os eventos de diários de simulação e gera gráficos de tempos (Seção 4.2.4.6). Considerando a ordem cronológica dos

eventos no diário de simulações, possibilita reconstruir e acompanhar passo a passo a evolução das simulações subjacentes.

As funcionalidades disponibilizadas pelo animador de gráficos de tempos incluem: a) visualização simultânea de múltiplos diários de simulação; b) localização de padrões de eventos; c) manipulação de janelas de tempos; e d) operação em cadeia com o simulador de tarefas e rotinas.

### 4.2.5.3 Diário de simulação

Os diários de simulação registram a evolução de gráficos de tempos gerados pelo simulador de escalonamento (Seção 4.2.4.3) e simulador de tarefas e rotinas (Seção 4.2.5.1). Ainda, podem ser guardados em arquivos XML para facilitar a memória dos diversos experimentos e intercâmbio estudante-estudante-professor.

A FIGURA 4.21 mostra o esquema (simplificado) das classes que suportam a estrutura de diário de simulação.

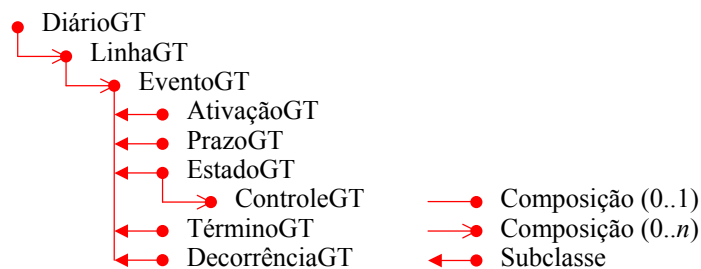


FIGURA 4.21: Esquema de classes do diário de simulação

Da mesma forma que o gráfico de tempos, o diário de simulação possui diversas linhas de controle ordenadas de acordo com as prioridades das tarefas simuladas. Cada linha de controle registra uma série de eventos, tais como instante de ativação de um serviço, início e duração de um estado e decorrência de prazo.

### 4.3 IMPLEMENTAÇÃO DA FERRAMENTA

A abrangência do ambiente de ensino e aprendizado do escalonamento proposto é muito ampla. Para viabilidade deste trabalho, considera-se dois momentos distintos: a especificação funcional da ferramenta como um todo e a implementação da parte essencial para a validação da ferramenta.

Os componentes tracejados na FIGURA 4.1 não estão implementados. O simulador de tarefas e rotinas é de alta complexidade e constitui por si só um trabalho futuro. Além disso, as setas tracejadas indicam fluxos não considerados na implementação, por exemplo, o gerador de grafos de segmentos não gera o diagrama de segmentos. Os demais componentes estão implementados total ou parcialmente. O gerador de planilhas de tempos opera com muitas restrições, por exemplo, não realiza medições de tempos em estruturas de repetição e seções críticas para recursos.

O objetivo básico da implementação é o de apresentar um protótipo suficiente para demonstrar os princípios da ferramenta e realizar alguns ensaios com estudantes.

### 4.4 CONCLUSÃO

Neste capítulo apresentou-se a ferramenta de apoio ao ensino e aprendizado do escalonamento de tarefas em sistemas tempo real, *AnimaTi*. Mostrou-se o conjunto de módulos que compõe a ferramenta e que caracteriza o ambiente completo para abordagem da questão do escalonamento, abrangendo desde a especificação do comportamento temporal e casual de algoritmos, passando pela medição de tempos de melhor e pior casos, até a análise e simulação do escalonamento e simulação dos algoritmos.

Finalmente, apresentou-se o ponto de corte entre a especificação e implementação da ferramenta com as justificativas da implementação incompleta no primeiro instante.



## 5 ESTUDOS DE CASOS

A implementação da ferramenta está adequada a modernas técnicas de construção de aplicativos em que testes não constituem parte determinante na avaliação da correção. No entanto, alguns problemas possuem alta complexidade e são de difícil compreensão. Nestes casos criou-se uma sucessão de protótipos, até atingir resultados satisfatórios.

Na verdade, a ferramenta como está, é um protótipo. Muitas situações não foram consideradas devido à relação complexidade e tempo disponível. Por outro lado, o arcabouço geral da ferramenta é muito amplo e envolveria diversos trabalhos específicos.

Na sua forma atual, a ferramenta pode ser utilizada em laboratório de disciplinas, tais como sistemas embarcados e sistemas operacionais. No entanto, nenhuma experiência foi realizada neste sentido. A ferramenta foi apresentada para alguns alunos da área de engenharia, sendo bem recebida.

Nas próximas seções demonstra-se aspectos da ferramenta através de estudos de casos envolvendo cenários fictícios.

### 5.1 CASO 1: EXPERIMENTO COMPLETO

O objetivo deste caso é mostrar um experimento que abrange todos os módulos da ferramenta (com exceção da simulação de tarefas e rotina). Inicialmente, define-se um arquivo fonte que descreve as tarefas, seus algoritmos e outros elementos. A FIGURA 5.1 mostra o código fonte textual deste algoritmo.

```

1 tarefa t1
2   período := 50ms
3   prazo := 50ms
4 variáveis
5   vp1 (probabilidade) := 0.5
6   vp2 (probabilidade) := 0.2
7 corpo
8   se em vp1 dos casos
9     trabalha [20..25]ms
10    com erro_x em 1% dos casos [0.8..0.9] adentro
11   senão
12     trabalha 10ms
13   fim de se
14   se em 50% dos casos
15     trabalha [0.2..0.22]ms
16   senão
17     trabalha [10..15]ms
18   fim de se
19 captura erro_x
20   trabalha 12ms
21 fim de t1
22
23 tarefa t2
24   período := 80ms
25   prazo := 100ms
26 corpo
27   trabalha 10ms
28 fim de t2
29
30 tarefa t3
31   período := 100ms
32   prazo := 20ms
33 corpo
34   trabalha 5ms
35 fim de t3

```

FIGURA 5.1: Arquivo fonte do caso 1

No caso, as tarefas t2 e t3 são triviais. Já a tarefa t1 possui certa complexidade, envolvendo construtos de seleção e captura de faltas. Em particular, as linhas 9 e 10 declaram um ator `trabalha` que gera um tempo de execução  $C_i$  no intervalo de 20 a 25ms (distribuição uniforme). O tempo de execução pode ser interrompido em 1% dos casos ao completar entre 80 e 90% de sua duração. Esta interrupção (por exemplo, divisão por zero) causa a falta `erro_x`, capturada ao final do corpo do algoritmo.

O arquivo fonte então é compilado e transformado em árvore de sintaxe. Em seguida gera-se o grafo de segmentos, e a partir deste gera-se a planilha de tempos mostrada na FIGURA 5.2. O próximo passo procede a análise e simulação de escalonamento.



5	Tarefa	Prio	Ti	Di	Ci	?Bi	?Ri	?q
✓	t3	3	100,0000 ms	20,0000 ms	5,0000 ms	0 s	5,0000 ms	0
✓	t1	2	50,0000 ms	50,0000 ms	40,0000 ms	0 s	45,0000 ms	0
✓	t2	1	80,0000 ms	100,0000 ms	10,0000 ms	0 s	95,0000 ms	1

M AS Totais: 3 tarefa(s), 0 descartada(s), 0 inviável(is)

FIGURA 5.2: Planilha de tempos do caso 1

A prioridade das tarefas é estabelecida por DMA. Como as tarefas não possuem regiões críticas, os diversos protocolos não influenciam os resultados da análise e simulação de escalonamento. Na planilha de tempos da FIGURA 5.1, a coluna ?Ri mostra os tempos de resposta das tarefas. A coluna ?q mostra que os serviços da tarefa t2 podem se sobrepor uma vez, já que o prazo é maior que o período. As tarefas são viáveis.

A FIGURA 5.3 mostra o gráfico de tempos da simulação de escalonamento a partir do instante crítico de tempo. Para conjuntos de tarefas sem regiões críticas, os resultados da simulação e análise de escalonamento são idênticos.

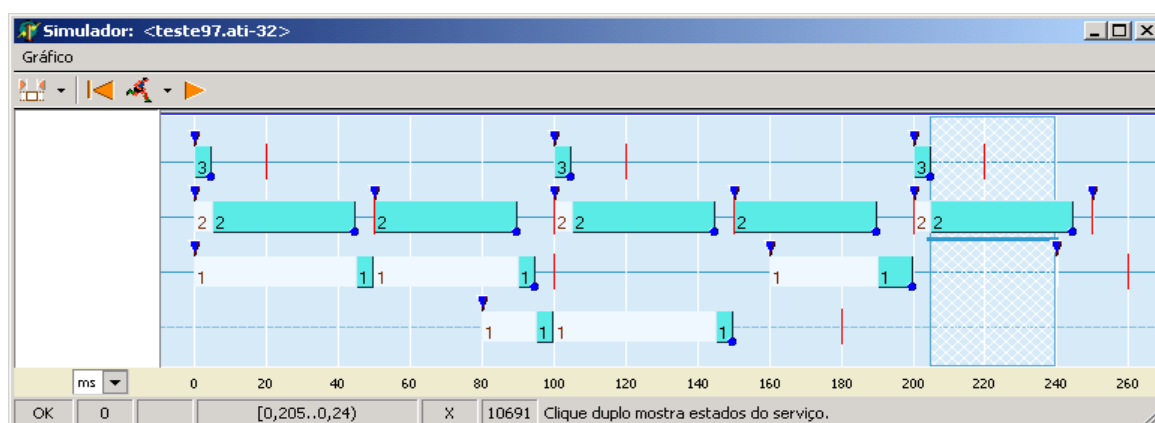


FIGURA 5.3: Gráfico de tempos do caso 1

A FIGURA 5.4 ilustra o grafo de segmentos (textual) durante a geração da planilha de tempos do caso. Além disso mostra a situação dos nodos imediatamente

antes de visitar seus nodos sucessores. Este quadro deve ser substituído pelo diagrama de segmentos gráfico.

Arquivo	Linha	Col	Mensagem
teste97.ati	6	3	variável vp2 não utilizada; [2582]
teste97.ati	7	1	>0 Ci=[0..0] Vi=[0..0] Ri=[0..0] V/P: vp1(0,5..0,5) vp2(0,2..0,2) REL: +(0..0) ARCOS: K0>2 [0..0] [0..0]; [2798]
teste97.ati	8	3	>2 Ci=[0..0] Vi=[0..0] Ri=[0..0] V/P: vp1(0,5..0,5)? vp2(0,2..0,2) REL: +(0..0) ARCOS: C0>4 [0..0] [0..0] C1>6 [0..0] [0..0]; [2798]
teste97.ati	9	5	>4 Ci=[0..0] Vi=[0..0] Ri=[0..0] V/P: vp1(0,5..0,5)? vp2(0,2..0,2) REL: +(0..0) ARCOS: K0>3 [0,02..0,025] [0..0] F0>5 [0,016..0,0225] [0..0]; [2798]
teste97.ati	20	3	>5 Ci=[0,016..0,0225] Vi=[0..0] Ri=[0,016..0,0225] V/P: vp1(0,5..0,5)? vp2(0,2..0,2) REL: +(0..0) ARCOS: K0>1 [0,012..0,012] [0..0]; [2798]
teste97.ati	12	5	>6 Ci=[0..0] Vi=[0..0] Ri=[0..0] V/P: vp1(0,5..0,5)? vp2(0,2..0,2) REL: +(0..0) ARCOS: K0>3 [0,01..0,01] [0..0]; [2798]
teste97.ati	13	3	>3 Ci=[0,01..0,025] Vi=[0..0] Ri=[0,01..0,025] V/P: vp1(0,5..0,5)? vp2(0,2..0,2) REL: +(0..0) ARCOS: K0>7 [0..0] [0..0]; [2798]
teste97.ati	14	3	>7 Ci=[0,01..0,025] Vi=[0..0] Ri=[0,01..0,025] V/P: vp1(0,5..0,5)? vp2(0,2..0,2) REL: +(0..0) ARCOS: C0>9 [0..0] [0..0] C1>10 [0..0] [0..0]; [2798]
teste97.ati	15	5	>9 Ci=[0,01..0,025] Vi=[0..0] Ri=[0,01..0,025] V/P: vp1(0,5..0,5)? vp2(0,2..0,2) REL: +(0..0) ARCOS: K0>8 [0,0002..0,00022] [0..0]; [2798]
teste97.ati	17	5	>10 Ci=[0,01..0,025] Vi=[0..0] Ri=[0,01..0,025] V/P: vp1(0,5..0,5)? vp2(0,2..0,2) REL: +(0..0) ARCOS: K0>8 [0,01..0,015] [0..0]; [2798]
teste97.ati	18	3	>8 Ci=[0,0102..0,04] Vi=[0..0] Ri=[0,0102..0,04] V/P: vp1(0,5..0,5)? vp2(0,2..0,2) REL: +(0..0) ARCOS: K0>1 [0..0] [0..0]; [2798]
teste97.ati	21	1	>1 Ci=[0,0102..0,04] Vi=[0..0] Ri=[0,0102..0,04] V/P: vp1(0,5..0,5)? vp2(0,2..0,2) REL: +(0..0) ARCOS;; [2798]
teste97.ati	26	1	>0 Ci=[0..0] Vi=[0..0] Ri=[0..0] V/P: REL: +(0..0) ARCOS: K0>2 [0..0] [0..0]; [2798]
teste97.ati	27	3	>2 Ci=[0..0] Vi=[0..0] Ri=[0..0] V/P: REL: +(0..0) ARCOS: K0>1 [0,01..0,01] [0..0]; [2798]
teste97.ati	28	1	>1 Ci=[0,01..0,01] Vi=[0..0] Ri=[0,01..0,01] V/P: REL: +(0..0) ARCOS;; [2798]
teste97.ati	33	1	>0 Ci=[0..0] Vi=[0..0] Ri=[0..0] V/P: REL: +(0..0) ARCOS: K0>2 [0..0] [0..0]; [2798]
teste97.ati	34	3	>2 Ci=[0..0] Vi=[0..0] Ri=[0..0] V/P: REL: +(0..0) ARCOS: K0>1 [0,005..0,005] [0..0]; [2798]
teste97.ati	35	1	>1 Ci=[0,005..0,005] Vi=[0..0] Ri=[0,005..0,005] V/P: REL: +(0..0) ARCOS;; [2798]

FIGURA 5.4: Grafo de segmentos textual do caso 1

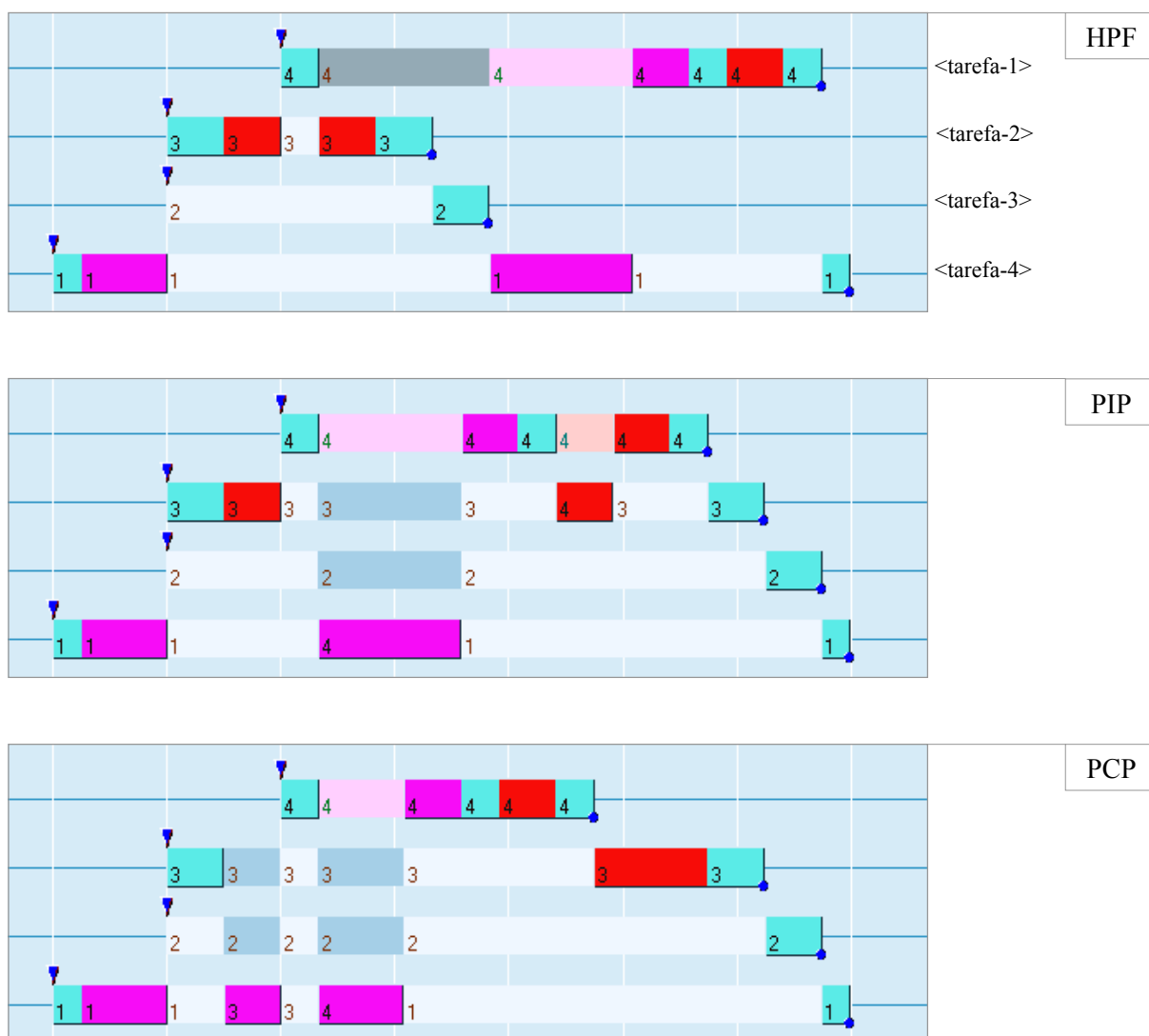
As linhas correspondem aos nodos e contêm as descrições destes (Mensagem) e as referências ao arquivo fonte (Arquivo, Linha, Col). A descrição dos nodos contém: número de identificação do nodo, campos para indicar tempos ( $C_i$ ,  $V_i$ ,  $R_i$ ), valores de variáveis (V/P), tempos de relógios (REL) e arcos de saída (ARCOS) com seus tempos ( $C_i$ ,  $V_i$ ). A FIGURA 4.10 auxilia na compreensão dos grafos.

Por exemplo, o ator `trabalha` das linhas 9 e 10 do arquivo fonte compõe um segmento, representado pelo nodo de número `>4`. Até este ponto, os tempos  $C_i$ ,  $V_i$  e  $R_i$  estão zerados, e as variáveis `vp1` e `vp2` mantêm seus valores iniciais. A partir deste nodo, o arco `K0` passa as fichas ao nodo `>3` por conexão, e o arco `F0` passa as fichas ao nodo `>5` por falta. No caso do arco `K0`,  $C_i$  está no intervalo de 20 a 25ms. No caso do arco `F0`,  $C_i$  está no intervalo de 16 a 22.5ms.

Os grafos iniciam no nodo de número `>0` e terminam no nodo de número `>1`. A ficha deste último nodo contém os valores dos tempos de resposta de melhor e pior caso. No caso da tarefa 1, linha 21 e nodo `>1`,  $R_i$  está no intervalo de 10.2 a 40ms, ou seja,  $WCET=40ms$  e  $BCET=10.2ms$ .

## 5.2 CASO 2: SIMULAÇÃO DE EXEMPLOS DESTE TRABALHO

O objetivo deste experimento é simular o escalonamento para os exemplos discutidos neste trabalho. No Capítulo 3 exemplifica-se os protocolos HPF, PIP, PCP e HL através da FIGURA 3.9, FIGURA 3.10, FIGURA 3.14 e FIGURA 3.15, respectivamente. O experimento define uma planilha de tempos em que os dados das tarefas são arranjados de modo a capturar as características destas figuras. A FIGURA 5.5 mostra os resultados da simulação sob os diversos protocolos. A tarefa <arefa-1> corresponde à tarefa  $\tau_1$ , a tarefa <arefa-2> à tarefa  $\tau_2$ , a tarefa <arefa-3> à tarefa  $\tau_3$  e a tarefa <arefa-4> à tarefa  $\tau_4$ .



(continua)

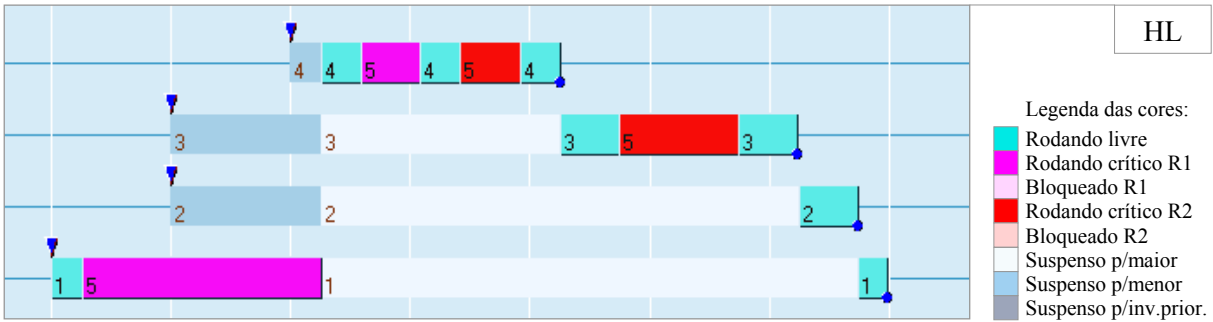
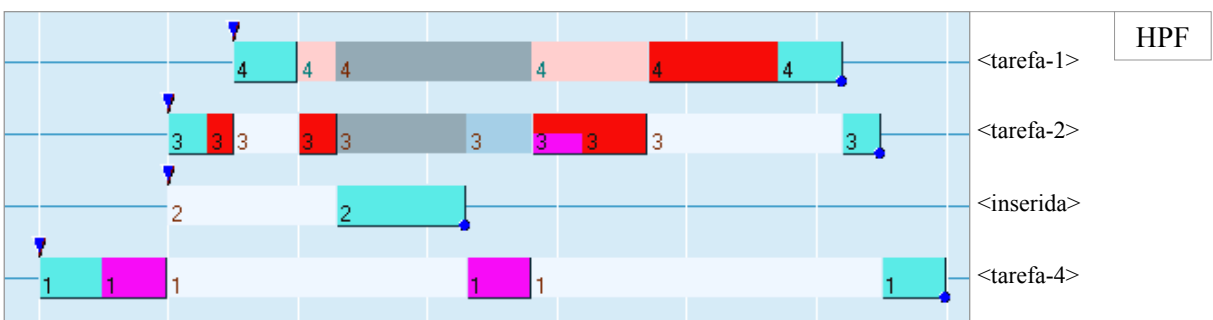


FIGURA 5.5: Gráficos de tempos do caso 2

Os resultados da simulação correspondem às figuras da referência. Sob o protocolo HPF, a tarefa < tarefa-1 > sofre inversão de prioridade devida às tarefas < tarefa-2 > e < tarefa-3 >.

### 5.3 CASO 3: SIMULAÇÃO DE EXEMPLOS DA LITERATURA

O objetivo deste experimento é simular o escalonamento para os exemplos discutidos em BRIAND e ROY (1999). No Capítulo 4 deste livro exemplifica-se os protocolos HPF, PIP, PCP e HL através da FIGURA 4.1, FIGURA 4.4, FIGURA 4.5 e FIGURA 4.6, respectivamente. O experimento define uma planilha de tempos em que os dados das tarefas são arranjados de modo a capturar as características destas figuras. A FIGURA 5.6 mostra os resultados da simulação sob os diversos protocolos. A tarefa < tarefa-1 > corresponde à tarefa  $\tau_C$ , a tarefa < tarefa-2 > à tarefa  $\tau_B$  e a tarefa < tarefa-3 > à tarefa  $\tau_A$ . A tarefa < inserida > é acrescentada para demonstrar a influência dos protocolos sobre tarefas sem regiões críticas.



(continua)

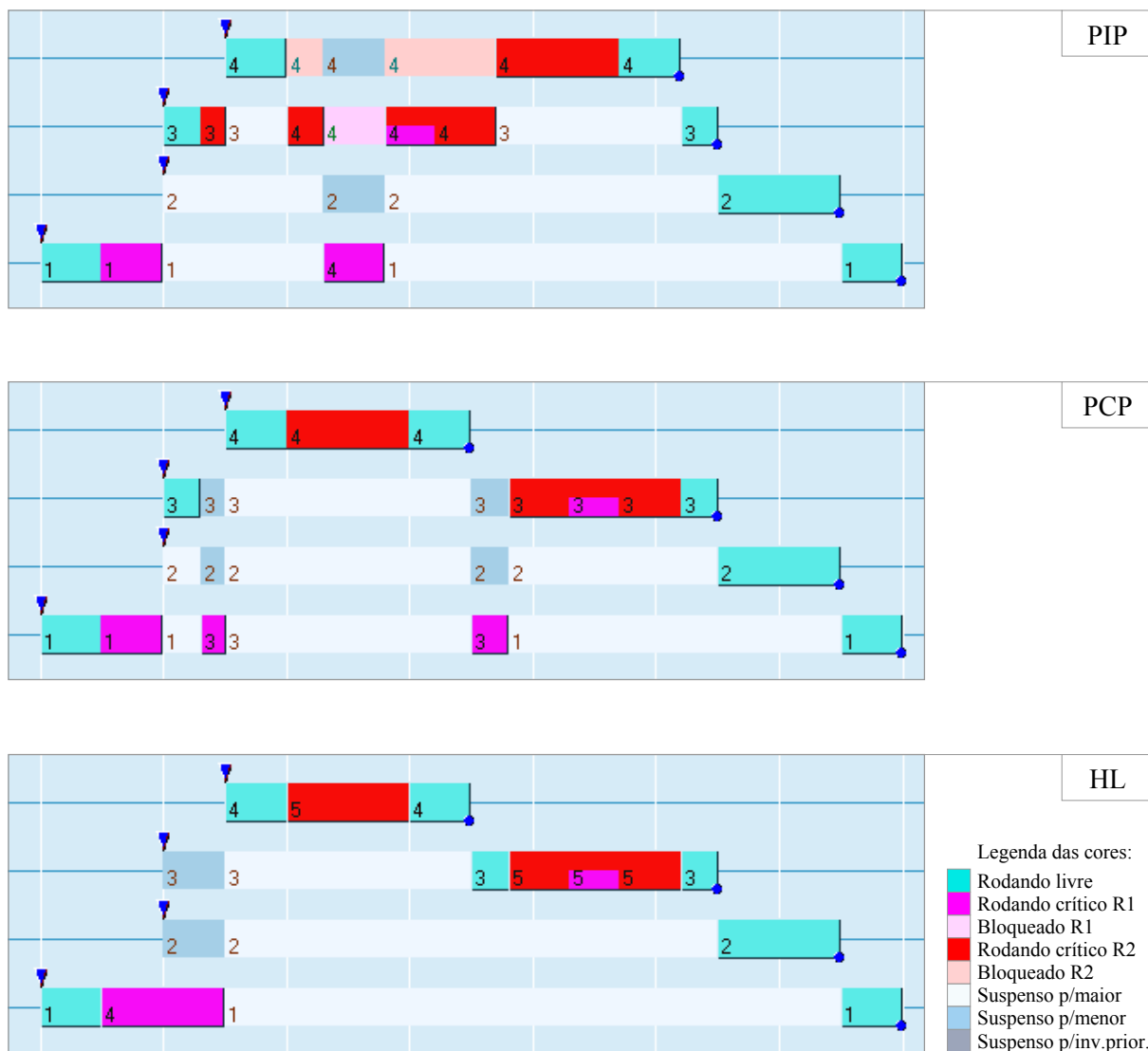


FIGURA 5.6: Gráficos de tempos do caso 3

Os resultados da simulação correspondem às figuras da referência. Sob o protocolo HPF, as tarefas < tarefa-1 > e < tarefa-2 > sofreram inversão de prioridade devido à tarefa < inserida >.

#### 5.4 CASO 4: INVERSÃO DE PRIORIDADE

O objetivo deste experimento é mostrar a ocorrência de inversão de prioridade com conseqüências catastróficas para um sistema tempo real severo. A planilha de tempos da FIGURA 5.7 mostra o conjunto de tarefas do experimento.

S	Tarefa	Prio	Pi	Ti	Di	Ci	?Bi	?Ri
✓	<tarefa-1>	3	3	50,0000 ms	50,0000 ms	10,0000 ms	40,0000 ms	50,0000 ms
✓	<tarefa-2>	2	2	201,0000 ms	201,0000 ms	50,0000 ms	40,0000 ms	120,0000 ms
✓	<tarefa-3>	1	1	239,0000 ms	239,0000 ms	50,0000 ms	0 s	130,0000 ms

FIGURA 5.7: Planilha de tempos do caso 4

Os resultados da análise de escalonamento sob o protocolo HPF indicam que as tarefas são viáveis. Contudo, o analisador de escalonamento emite um alerta, indicando que a tarefa <tarefa-1> pode sofrer inversão de prioridade enquanto espera pelo recurso R1. A FIGURA 5.8 mostra dois gráficos de tempos sob os protocolos HPF e PIP, capturados na mesma região de tempo de simulação da planilha de tempos do caso. Sob o protocolo HPF a tarefa <tarefa-1> sofre inversão de prioridade e perde o prazo.

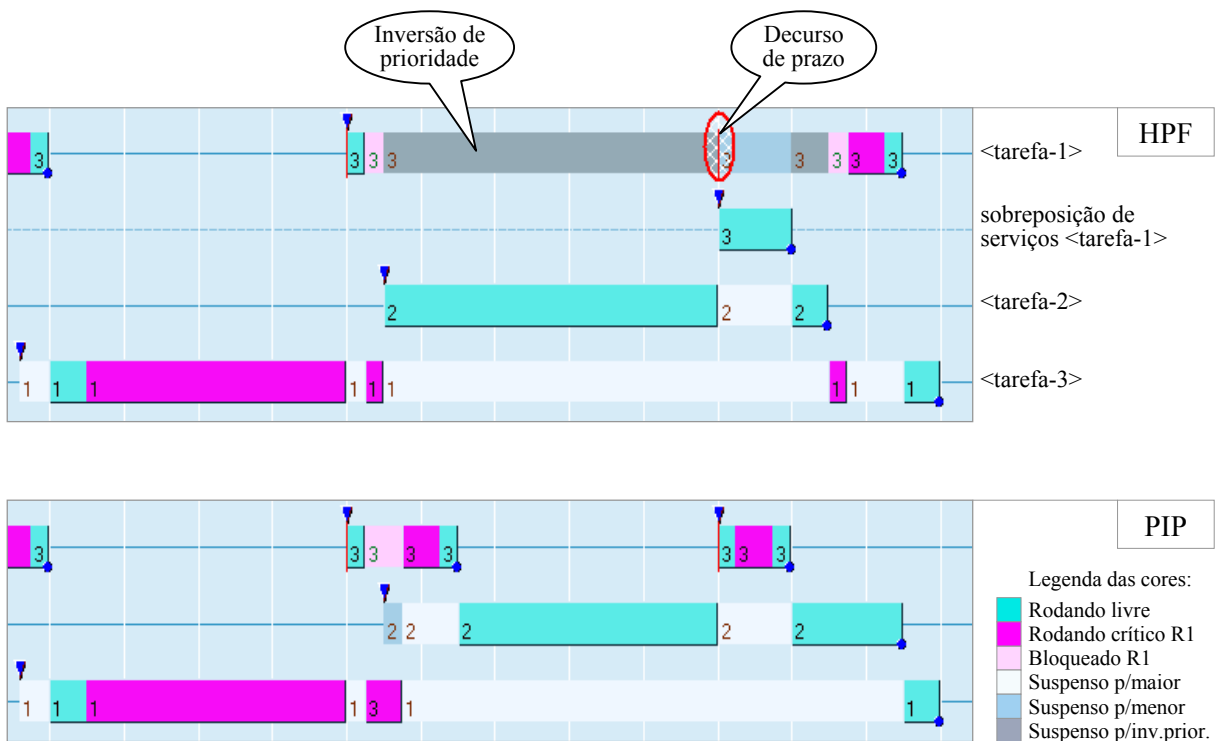


FIGURA 5.8: Gráficos de tempos do caso 4

Sob o protocolo PIP os resultados da análise de escalonamento são idênticos àqueles apresentados na planilha de tempos do caso. Nestas condições não ocorre inversão de prioridade.

### 5.5 CASO 5: ANÁLISE E SIMULAÇÃO DE ESCALONAMENTO

O objetivo deste experimento é avaliar e comparar os resultados da análise e simulação de escalonamento. O experimento baseia-se nos dados da planilha de tempos da FIGURA 5.9. Além dos dados das colunas  $T_i$ ,  $D_i$  e  $C_i$ , a tarefa < tarefa-1 > possui uma região crítica de 10ms para o recurso R2 e uma região crítica aninhada em R2 de 5ms para o recurso R3. Da mesma forma, a tarefa < tarefa-2 > possui uma região crítica de 10ms para o recurso R1 e uma região crítica aninhada em R1 de 5ms para o recurso R2. Finalmente, a tarefa < tarefa-3 > possui uma região crítica de 50ms para o recurso R2 e uma região crítica aninhada em R3 de 20ms para o recurso R1.

S	Tarefa	Prio	$T_i$	$D_i$	$C_i$	?Bi	?Ri
✓	<tarefa-1>	3	100,0000 ms	100,0000 ms	40,0000 ms	55,0000 ms	95,0000 ms
✓	<tarefa-2>	2	200,0000 ms	200,0000 ms	30,0000 ms	50,0000 ms	160,0000 ms
✗	<tarefa-3>	1	350,0000 ms	360,0000 ms	150,0000 ms	0 s	370,0000 ms

FIGURA 5.9: Planilha de tempos do caso 5

A prioridade das tarefas é estabelecida por DMA. O protocolo utilizado é HPF. Então a análise de escalonamento determina os tempos de bloqueio e resposta de pior caso para as tarefas nas colunas ?Bi e ?Ri da planilha, respectivamente. Pode-se observar que a tarefa < tarefa-3 > não é viável, pois, seu tempo de resposta é maior que o prazo.

Nas mesmas condições da análise de escalonamento, a FIGURA 5.10 mostra a evolução da simulação de escalonamento. A tarefa < tarefa-3 > perde o prazo já na região crítica de tempo. Por outro lado, os tempos de resposta da tarefa < tarefa-1 > e < tarefa-2 > são menores que os tempos obtidos na análise de escalonamento no mesmo

intervalo. Isto se deve ao fato dos tempos de bloqueio destas tarefas não estarem situados em seus piores momentos.

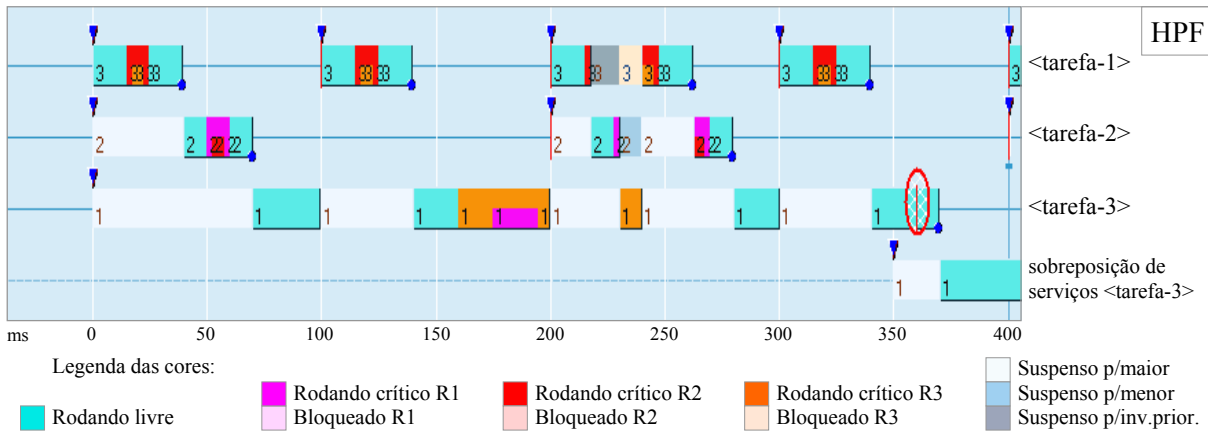


FIGURA 5.10: Gráfico de tempos do caso 5

Os resultados da análise e simulação de escalonamento sob os protocolos PIP, PCP e HL não são piores que os resultados mostrados na planilha de tempos. Em particular, a tarefa <arefa-3> é inviável para todos os protocolos.

## 5.6 CASO 6: IMPASSE DE BLOQUEIO

O objetivo deste experimento é estender o estudo de caso 5 (Seção 5.5), para verificação de impasse de bloqueio. As regiões críticas das tarefas na FIGURA 5.9 estão dispostas em ciclo, isto é, a tarefa <arefa-1> retém o recurso R2 enquanto requer o recurso R3, a tarefa <arefa-2> retém o recurso R1 enquanto requer o recurso R2, e a tarefa <arefa-3> retém o recurso R3 enquanto requer o recurso R1.

A análise de escalonamento sob os protocolos HPF e PIP prevê um possível impasse de bloqueio, envolvendo as três tarefas e os três recursos. A simulação de escalonamento localiza um destes instantes em que ocorre um impasse de bloqueio como mostra a FIGURA 5.11.



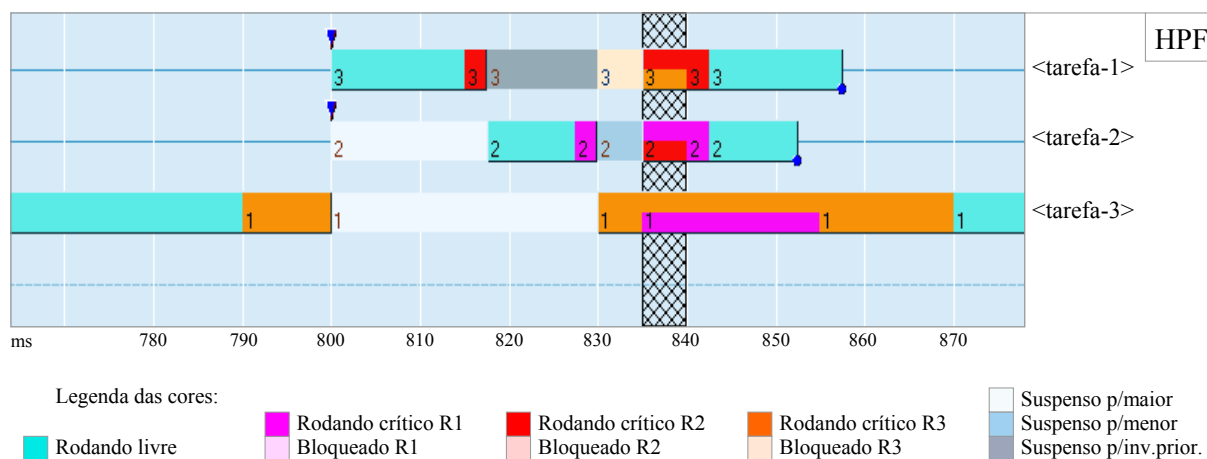


FIGURA 5.11: Gráfico de tempos do caso 6

Na situação de impasse de bloqueio, não é possível determinar um próximo evento no horizonte do cursor, e o algoritmo de simulação pára.

## 5.7 CONCLUSÃO

Os casos apresentados neste capítulo mostram a capacidade da ferramenta em ilustrar cenários em torno do assunto escalonabilidade. Abordam diversas situações, tais como impasse de bloqueio e inversão de prioridade. Embora os aspectos dinâmicos da simulação sejam de máxima relevância para o ensino e aprendizado, os exemplos mostrados não conseguem transmitir estas propriedades da ferramenta.

Fica claro (por exemplo, no caso 5) que a simulação não substitui a análise e serve apenas para animar os mecanismos de escalonamento clássico. No contexto da ferramenta não é possível determinar a escalonabilidade de um conjunto de tarefas através da simulação do escalonamento.



## 6 CONCLUSÃO

A ferramenta *AnimaTi* captura diversos aspectos relacionados ao estudo de escalonamento em sistemas em tempo real. Baseia-se principalmente na teoria clássica de escalonamento em aplicativos cooperativos tal como mostra a revisão bibliográfica nos Capítulos 2 e 3.

O ponto de partida para a definição da ferramenta foi a decomposição dos conteúdos relacionados em: a) criação de algoritmos abstratos, b) compilação dos algoritmos, c) segmentação dos algoritmos e análise de tempos, d) análise e simulação de escalonamento, e e) simulação dos algoritmos através de máquinas virtuais. Esta decomposição motivou a divisão da ferramenta em módulos.

A interface gráfica oferecida pela ferramenta é de fácil uso. Possibilita interagir com os experimentos em diversos pontos e acompanhar as simulações e animações passo a passo. Diversos elementos têm interpretação visual, tais como diagramas de segmentos e gráficos de tempos (Gantt).

Os experimentos podem ser registrados em arquivos XML, que oferecem suporte para criação e troca de listas de exercícios.

A ferramenta não impõe uma ordem na apresentação dos conteúdos relativos ao escalonamento. O professor pode utilizar o recurso de diversos modos e com diversos graus de dificuldade, por exemplo, iniciando com a análise e simulação do escalonamento e depois passando à análise de algoritmos e segmentação.

### 6.1 RESULTADOS

O resultado principal obtido a partir deste trabalho foi a implementação do protótipo da ferramenta *AnimaTi*. Na situação em que se apresenta, a ferramenta pode ser utilizada em laboratório para as disciplinas que tratam do escalonamento, tais como sistemas operacionais e sistemas embarcados.

A ferramenta foi apresentada a alguns alunos da disciplina de sistemas operacionais, que manifestaram interesse e imediatamente perceberam sua utilidade

em exercícios complexos. Os aspectos visuais da animação de escalonamento chamam à atenção especialmente a operação passo a passo dos protocolos de escalonamento.

## 6.2 CONTRIBUIÇÕES

A principal contribuição deste trabalho é a definição de uma ferramenta como arcabouço para os conteúdos relacionados ao estudo do escalonamento em sistemas em tempo real. A versão final da ferramenta deve cobrir diversas fases, começando pela descrição de aplicativos abstratos por algoritmos escritos na linguagem *AnimaTi*, que são decompostos em segmentos para então analisar os tempos de execução, de espera e as seções críticas de recursos. As planilhas de tempos obtidas são analisadas e simuladas. Além disso, os algoritmos podem ser simulados por máquinas virtuais sob controle de diversos escalonadores.

Como ferramenta de apoio ao ensino e aprendizado, enfatiza a relação estudante-estudante-professor, oferecendo funcionalidades para o intercâmbio de programas, planilhas e gráficos. A simulação e animação podem operar de forma automática ou manual. No primeiro caso, a ferramenta decide sobre o escalonamento, e no segundo caso, o estudante ou professor propõem ações que são validadas pela ferramenta.

Um aspecto importante diz respeito à língua nacional. Todos os conceitos, idéias e termos são apresentados em português. Não há pretensão transnacionalizante.

## 6.3 TRABALHOS FUTUROS

A versão atual da ferramenta constitui-se em apenas um protótipo para demonstrar e validar a parte essencial dos módulos especificados exceto o simulador de tarefas e rotinas (FIGURA 4.1). Para completar a ferramenta são previstos diversos trabalhos, tais como:

- estender o gerador de grafos de segmentos, possibilitando criar diagramas de segmentos com capacidade de visualizar as atividades do gerador de planilhas de tempos e acompanhar o simulador de tarefas e rotinas;

- rever o gerador de planilhas de tempos e completar o tratamento de todos os construtos da linguagem *AnimaTi*;
- incluir construtos na linguagem *AnimaTi* para tratar de regiões críticas e monitores;
- incluir o tratamento de ambientes com múltiplos processadores no módulo 4 tanto simétricos como assimétricos; verificar a possibilidade de operar com protocolos de escalonamento distintos nos diversos processadores;
- incluir o tratamento de recursos com múltiplas instâncias no módulo 4; verificar a possibilidade de operar com protocolos de escalonamento distintos para os diversos recursos;
- criar o simulador de tarefas e rotinas, o diário de simulação e o animador de gráficos de tempos no módulo 5; incluir outros escalonadores capazes de tratar a ativação dinâmica de tarefas, os protocolos como EDF e LLF e a comunicação e sincronização por encontros;
- rever o algoritmo de distribuição de seções críticas nos serviços; possibilitar a distribuição manual e aleatória; implementar as funcionalidades mostradas na Seção 4.2.4.3;
- incluir no simulador de escalonamento os modos: simulação automática e verificação de simulação; no modo de verificação o usuário deve prever o próximo passo do simulador, e este por sua vez valida a previsão;
- criar o editor de arquivos fonte XML como exercício de implementação de suporte para programação através de construtos gráficos (Apêndice D);
- desenvolver um esquema genérico de escalonadores em Redes de Petri (RdP) (temporizadas e/ou temporais) às quais se pode acoplar RdPs representando tarefas e rotinas; estas últimas devem ser geradas diretamente dos grafos de segmentos;

Os trabalhos futuros propostos acima estão relacionados com a ferramenta no sentido de apoio ao ensino e aprendizado. Trabalhos podem ser realizados na tentativa de traduzir programas escritos em linguagens como C++ para a linguagem *AnimaTi*. Desta forma estes poderiam ser analisados e simulados.

Mais ainda, problemas de escalonamento não são restritos a sistemas em tempo real. Como a linguagem *AnimaTi* é abstrata quanto aos aspectos reais dos problemas a serem analisados e simulados, é possível modelar diversos ambientes com serviços concorrentes e paralelos, por exemplo, linhas de produção.

**REFERÊNCIAS BIBLIOGRÁFICAS**

AHO, A. V.; SETHI, R., ULLMAN, J. D. **Compilers, Principles, Techniques, and Tools**. Addison-Wesley. 1986.

ANCILOTTI, P.; BUTTAZZO, G.; NATALE, M. di; SPURI, M. **Design and Programming Tools for Time Critical Applications**. Real-Time Systems, Vol. 14, No. 3, pp. 251-267, 1998.

AONIX. **ActivAda Real-Time**, Disponível em: <<http://www.aonix.de/activada.html>>. Acessado em: 26 abr 2006.

AUDSLEY, N. C.; BURNS, A.; RICHARDSON, M.; TINDELL, K. W.; WELLINGS, A. J. **Applying new scheduling theory to static priority preemptive scheduling**. Software Engineering Journal 8, set. 1993.

AUDSLEY, N. C.; BURNS, A.; RICHARDSON, M. F.; WELLINGS, A. J. **STRESS: a Simulator for Hard Real-Time Systems**. Software - Practice and Experience. 24(6):543-564, 1994.

AUDSLEY, N. C.; BURNS, A.; DAVIS, R. I.; TINDELL, K. W.; WELLINGS, A. J. **Fixed priority preemptive scheduling: An historical perspective**. Real-Time Systems 8, 1995.

BLUMENTHAL, J.; GOLATOWSKI, F.; HILDEBRANDT, J.; TIMMERMANN, D. **Framework for Validation, Test and Analysis of Real-Time Scheduling Algorithms and Scheduler Implementations**. 13th IEEE International Workshop on Rapid System Prototyping (RSP'02), 2002.

BLUMENTHAL, J. **Automatisches Entwurfs- und Entwicklungssystem für harte Echtzeitsysteme**. Diplomarbeit, Universität Rostock, 2002.

BREGANT, E. G. **Análise de Escalonamento de Tarefas no Ambiente Perf**, Dissertação de Mestrado no Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial, CPGEI, UTFPR, Curitiba, 2002.

BRIAND, L. P.; ROY, D. M. **Meeting Deadlines in Hard Real-Time Systems: The Rate Monotonic Approach**. IEEE Computer Society Press, 1999.

BURNS, A.; WELLINGS, A. J. **Advanced fixed priority scheduling**. In M. Joseph Ed., Real-Time Systems: Specification, Verification and Analysis, Prentice-Hall, 1996.

BURNS, A.; WELLINGS, A. J. **Real-Time Systems and Programming Languages**. Addison-Wesley, 1st edition 1990, 2nd edition 1996, 3rd edition 2001.

BUTTAZZO, G. **Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications**. Kluwer Academic Publishers, 1997.

CARBONKERNEL. **CarbonKernel Real-time Operating System Simulator**. Version 1.4, User Manual, 2001.

CARBONKERNEL. Disponível em: <<http://download.gna.org/carbonkernel/doc/>>. Acessado em: 03/04/2006.

CHATTERJEE, S.; BRADLEY, K.; MADRIZ, J.; COLQUIST, J. A.; STROSNIDER, J. **SEW: A Toolset for Design and Analysis of Distributed Real-Time Systems**. IEEE, In Proceedings RTAS'97, 1997.

DECOTIGNY, D.; PUAUT, I. **ARTISST: An Extensible and Modular Simulation Tool for Real-Time Systems**. 5th IEEE International Symposium on Object-oriented Real-time distributed Computing, apr. 2002.

EKER, J.; CERVIN, A. **A Matlab Toolbox for Real-Time and Control Systems Co-Design**. 6th International Conference on Real-Time and Control Systems, Hong Kong, P.R. China, 1999.

FALARDEAU, J. D. G. **Schedulability analysis in rate monotonic based systems with application to the CF-188**. Master's thesis, Department of Electrical and Computer Engineering, Royal Military College of Canada, 1994.

FIDGE, C. J. **Real-Time Scheduling Theory**. School of Information Technology and Electrical Engineering, The University of Queensland, Queensland 4072, Australia, 2002.

GIERING III, E.; BAKER, T. P. **A tool for deterministic scheduling of real-time programs implemented as periodic Ada tasks**. ACM Ada letters XIV, 1994.

GHOSE, K.; AGGARWAL, S.; VASEK, P.; CHANDRA, S.; RAGHAV, A.; GHOSH, A.; VOGEL, D. R. **ASSERTS: A Toolkit for Real-Time Software Design, Development and Evaluation**. 9th Euromicro Workshop on Real Time Systems (euromicro-rts '97) 1997.

GÓES, J. A. **Perf: Ambiente de Desenvolvimento e Estimação Temporal de Sistemas em Tempo Real**, Dissertação de Mestrado no Curso de Pós-Graduação em Engenharia Elétrica e Informática Industrial, CPGEI, UTFPR, Curitiba, 2001.



- GRIGG, A.; AUDSLEY, N. C. **Towards a scheduling and timing analysis solution for integrated modular avionics systems**. *Microprocessors and Microsystems* 22, 1999.
- HERLIHY, M. P.; LISKOV, B. **A Value Transmission Method for Abstract Data Types**. *ACM Transactions on Programming Languages and Systems*, Vol.4, No.4, oct. 1982.
- HUMPHREY, M.; STANKOVIC, J. A. **CAISARTS: A Tool for Teal-Time Scheduling Assistance**. In *Proceedings RTAS 96*, 1996.
- INTEL. **MultiProcessor Specification**. Version 1.4, 1997.
- KNUTH, D. E. **The Art of Computer Programming**. Vol 2, 3rd ed. Boston: Addison-Wesley. 1998. Seção 3.4.1, p. 133.
- LARSON, J. **Schedulite: A Fixed Priority Shceduling Analysis Tool**. MSc Thesis, ASTEC-RT, Universidade de Upsala, Suécia, 1996.
- LEHOCZKY, J. P.; SHA, L.; DING, Y. **The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior**. Tech. Report, Department of Statistics, Carnegie-Mellon University, Pittsburgh, Pa., 1987.
- LEUNG, J. L.; WHITEHEAD, J. **On the Complexity of Fixed-Priority Scheduling of Periodic, Real-Time Tasks**. *Performance Evaluation* Vol. 2, No. 1, dez. 1982.
- LIU, C. L.; LAYLAND, J. W. **Scheduling algorithms for multiprogramming in a hard real-time environment**. *Journal of the ACM* 20, n. 1, p. 46–61, 1973.
- LOCKE, C. D. **Software architecture for hard real-time applications: Cyclic executives vs. fixed priority executives**. *Journal of ACM* 20, 1, 46-61, 1992.
- MORON, C. E.; RIBEIRO, J. R. P.; SILVA, N. C. da. **A Teaching Environment for Development of Parallel Real-Time Programs**. 28th Annual FIE'98, 1998.
- RTCA. **Software Considerations in Airborne Systems and Equipment Certification**. RTCA Inc. Special Committee 167 Documento No. RTCA/DO-178B, 1992.
- SINGHOFF, F.; LEGRAND, J.; NANA, L.; MARCÉ, L. **Cheddar: a Flexible Real Time Scheduling Framework**. SIGAda'04, Atlanta, Georgia, USA, 2004.
- SPRUNT, B.; SHA, L.; LEHOCZKY, J. P. **Aperiodic task scheduling for hard real-time systems**. *Journal of Real-Time Systems* 1, jul. 1989.

STANKOVIC, J. A. **Misconceptions about real-time computing**. IEEE Computer, v. 21, out. 1988.

STEWART, D. B.; ARORA, G. **A Tool for Analyzing and Fine Tuning the Real-Time Properties of an Embedded System**. IEEE Transactions on Software Engineering, vol. 29, No. 4, 2003.

STORCH, M. F.; LIU, J. W.-S. **DRTSS: A Simulation Framework for Complex Real-Time Systems**. Real-Time Technology and Applications Symposium, 1996.

TAFT, S. T.; DUFF, R. A. **Ada 95 Reference Manual: Language and Standard Libraries**. Springer-Verlag, 1997.

TANENBAUM, A. S. **Computer Networks**. Prentice-Hall, 1988.

TANENBAUM, A. S.; WOODHULL, A. S. **Sistemas Operacionais - Projeto e Implementação**. 2a. Edição, 2000.

TIMESYS. **TimeWiz: Model, Analyse and Simulate Teal-Time System**. Disponível em: <[http://www.timesys.com/\\_content/media/docs/prodlit/TimeWiz%20Data%20Sheet.pdf](http://www.timesys.com/_content/media/docs/prodlit/TimeWiz%20Data%20Sheet.pdf)>. Acessado em: 26 abr 2006.

TINDELL, K. W.; BURNS, A.; WELLINGS, A. J. **An extendible approach for analyzing fixed priority hard real-time tasks**. Real-Time Systems 6, 1994.

TINDELL, K. W. **Deadline Monotonic Analysis**. Embedded Systems Programming 13, jun. 2000.

TOKUDA, H.; KOTERA, M.. **Scheduler 1-2-3: An Interactive Schedulability Analyzer for Real-Time Systems**. In Proceedings Compsac 88, 1988.

TRI-PACIFIC. Tri-Pacific Software: **RapidRMA Tutorial**. Disponível em: <[http://www.tripac.com/html/downloads.html#rma\\_section\\_tutorial](http://www.tripac.com/html/downloads.html#rma_section_tutorial)>. Acessado em: 25 abr 2006.

UTSA. **Using UTSA Process Scheduling Simulator**. Disponível em: <[http://vip.cs.utsa.edu/simulators/guides/ps/ps\\_doc.html](http://vip.cs.utsa.edu/simulators/guides/ps/ps_doc.html)>. Acessado em: 12 abr. 2006.

## APÊNDICE A LINGUAGEM *ANIMATI*

A linguagem *AnimaTi* permite descrever experimentos através de algoritmos abstratos operando com acaso e tempo. Nas próximas seções apresenta-se os aspectos sintáticos da linguagem. A semântica e as restrições são objeto de um manual de referência da linguagem, embora sejam facilmente inferidas pela semelhança com linguagens naturais e elementos de linguagens de programação comuns.

### A.1 Descrição de um experimento

Um experimento é descrito por um ou mais arquivos fonte textuais ou XML. Um dos arquivos figura como arquivo fonte principal. Através da inclusão de arquivos a partir do arquivo principal, pelo uso recorrente do construto "arquivo" (ou do elemento "arquivo" nos arquivos fonte XML), pode-se compor uma hierarquia de arquivos fonte. Cada arquivo só pode ser incluído uma única vez.

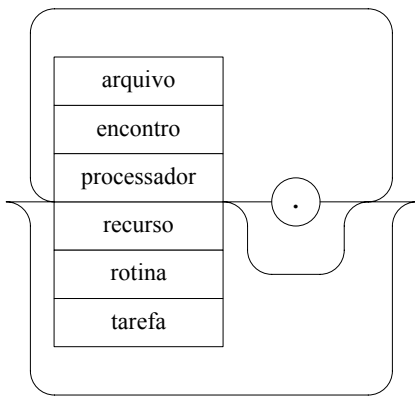
A linguagem requer que os elementos de encontros, processadores, recursos e rotinas estejam definidos antes de referenciados. No caso de rotinas, isto implica que estas não poderão ser executadas com recorrência cíclica, que tornaria praticamente impossível limitar a profundidade de recorrência. Para a análise de escalonabilidade, os algoritmos devem operar com tempos limitados.

No Apêndice B descreve-se a sintaxe de arquivos fonte XML. Tanto arquivos fonte textuais, como arquivos fonte XML podem ser incluídos alternadamente.

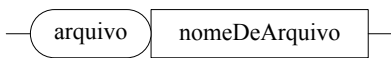
### A.2 Definição da sintaxe da linguagem *AnimaTi*

A seguir, apresenta-se a definição da sintaxe da linguagem para descrição de tarefas, rotinas, recursos, processadores e pontos de encontro, utilizando a notação para grafos de sintaxe (proposta por Niklaus Wirth) definida na próxima seção.

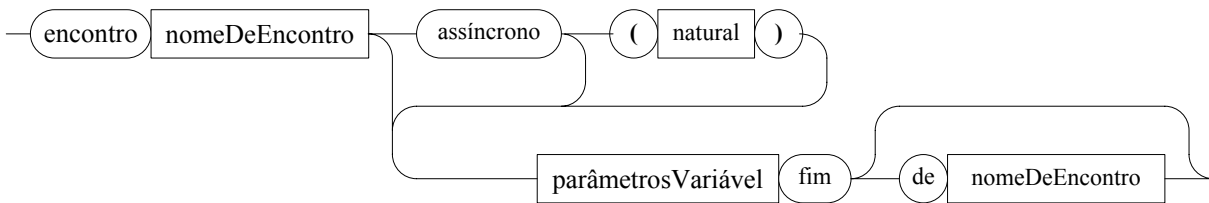
fonteTextual -----



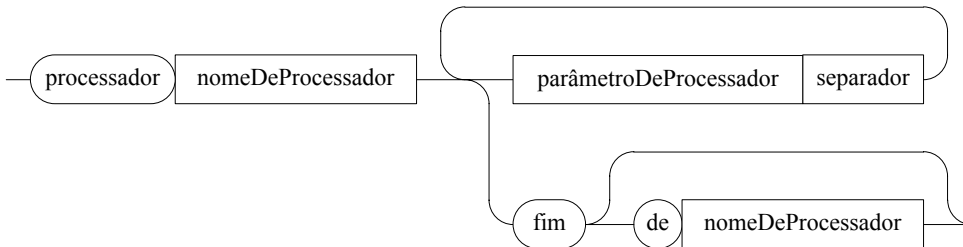
arquivo -----



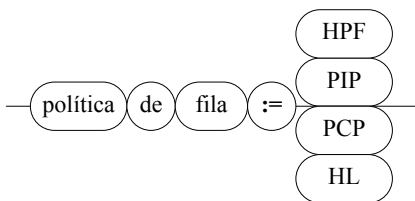
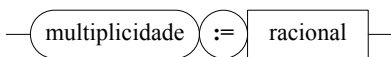
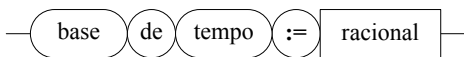
encontro -----



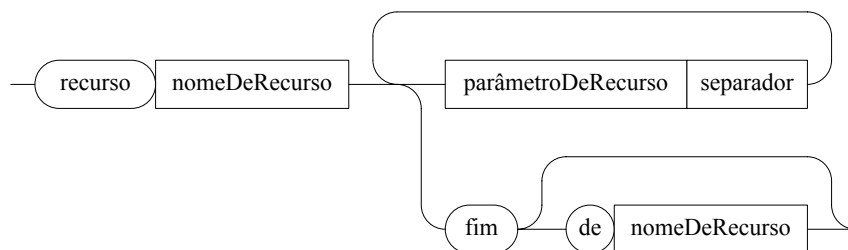
processador -----



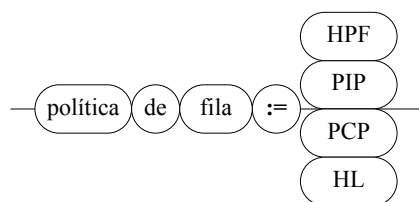
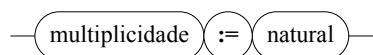
parâmetroDeProcessador -----



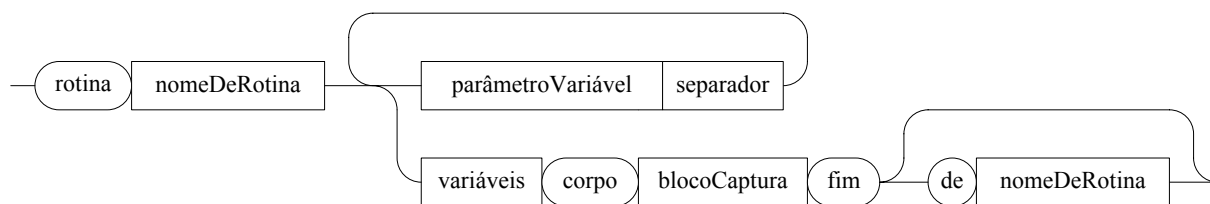
## recurso -----



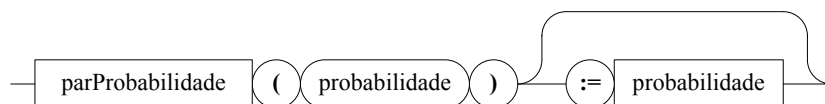
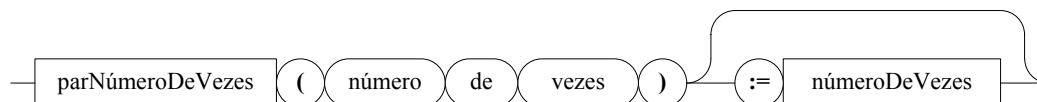
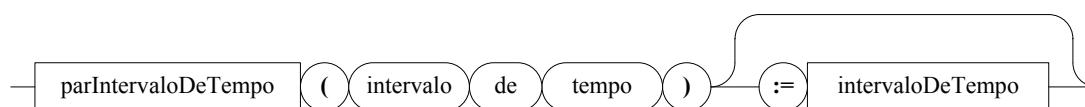
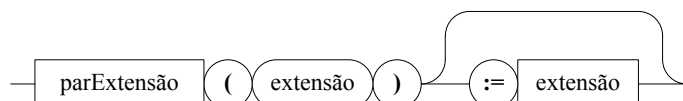
## parâmetroDeRecurso -----



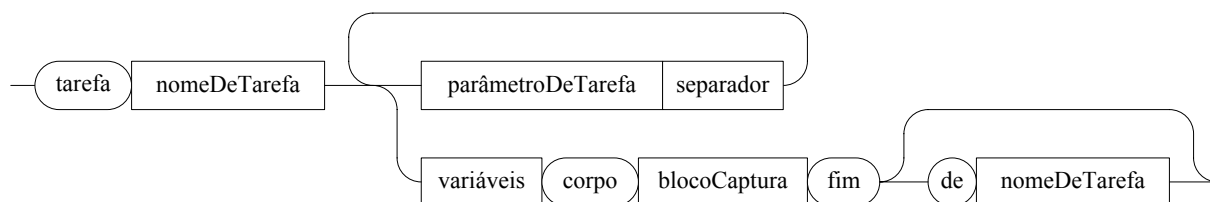
## rotina -----



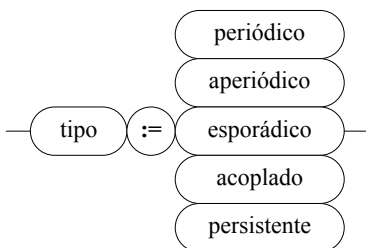
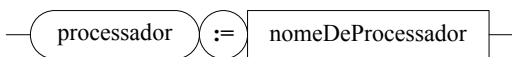
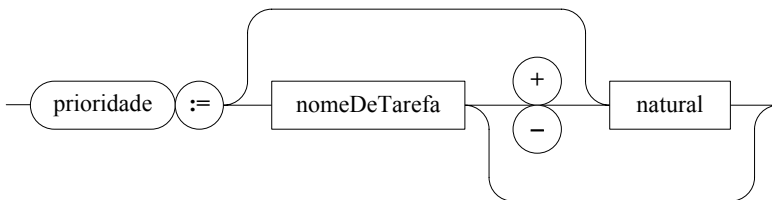
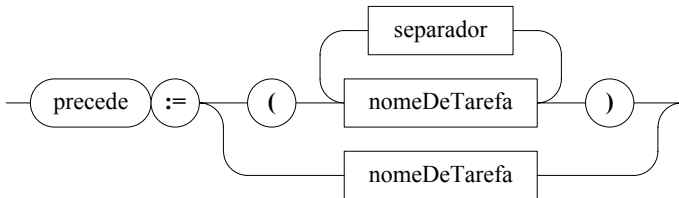
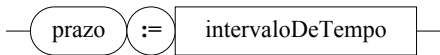
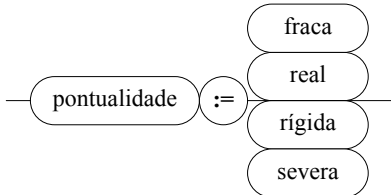
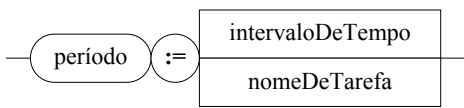
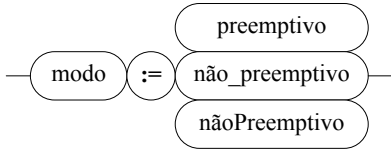
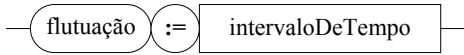
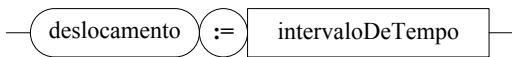
## parâmetroVariável -----



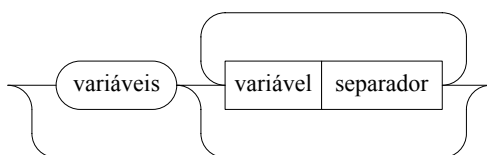
## tarefa -----



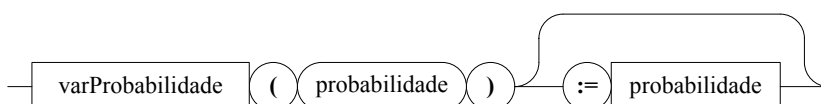
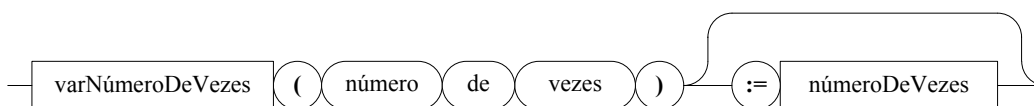
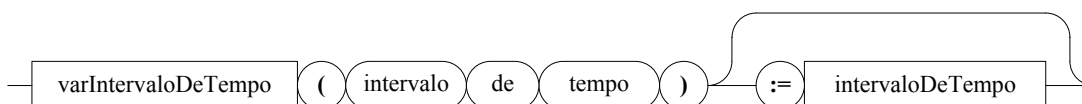
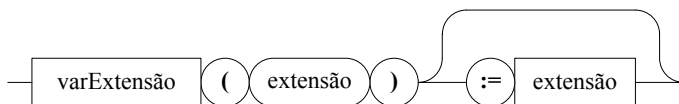
parâmetroDeTarefa-----



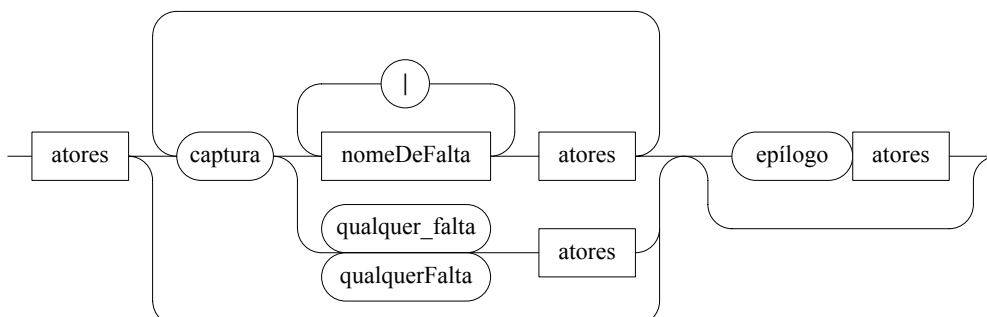
## variáveis



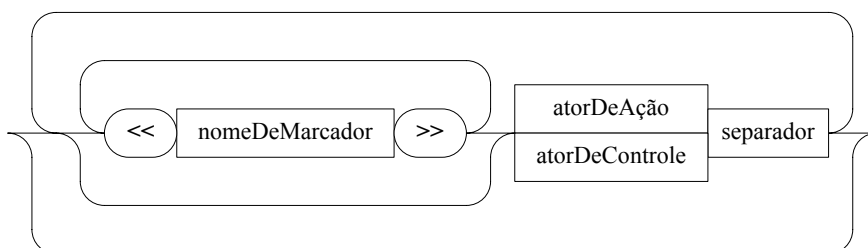
## variável



## blocoCaptura



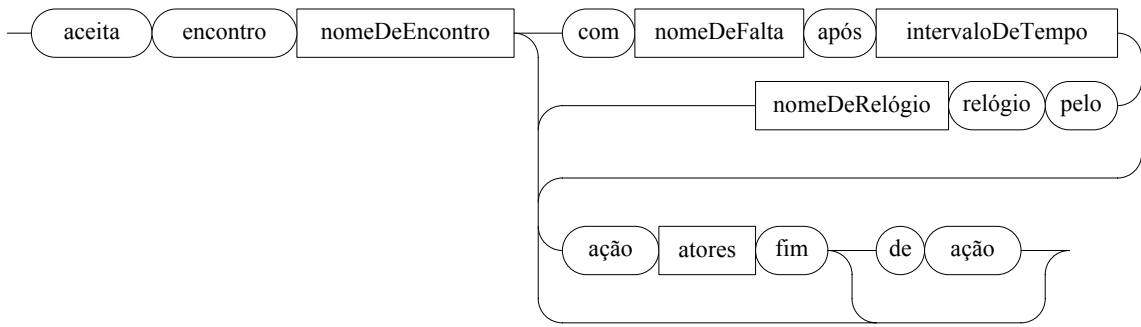
## atores



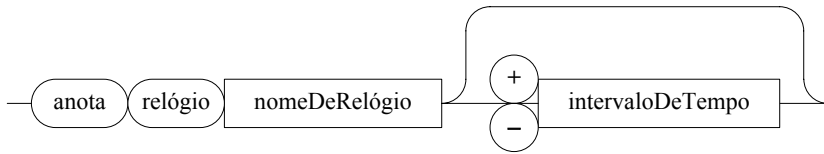
## atorAbandona (atorDeControle)



atorAceitaEncontro (atorDeControle)-----



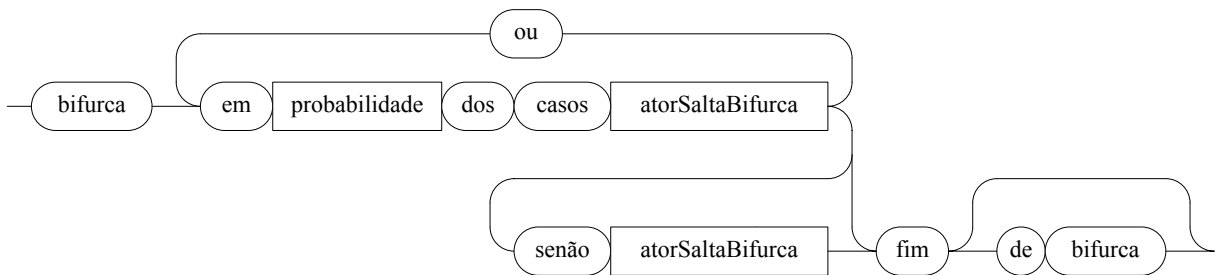
atorAnotaRelógio (atorDeControle)-----



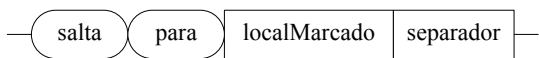
atorAtivaTarefa (atorDeControle)-----



atorBifurca (atorDeControle)-----



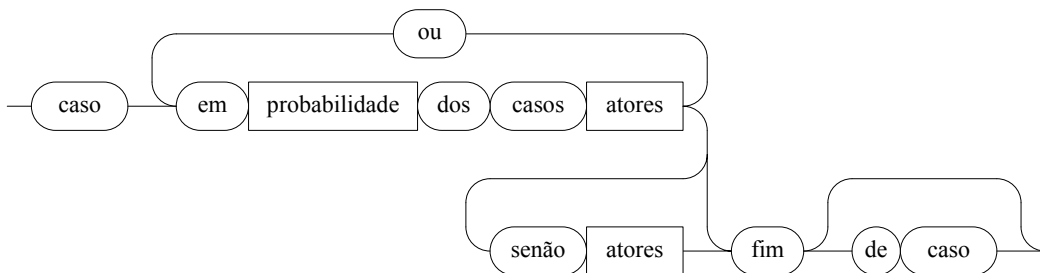
atorSaltaBifurca-----



atorBloco (atorDeControle)-----

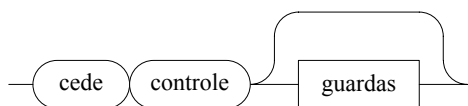


atorCaso (atorDeControle)-----

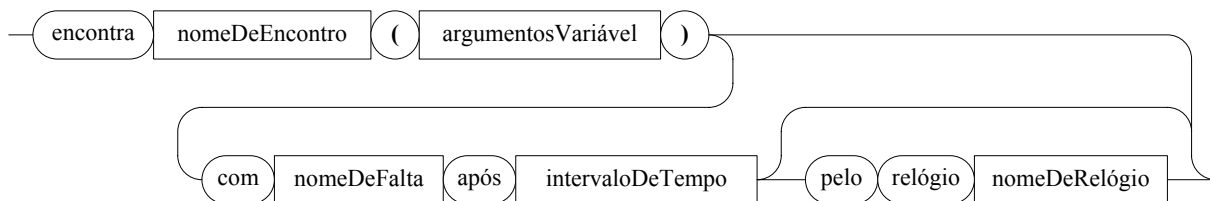




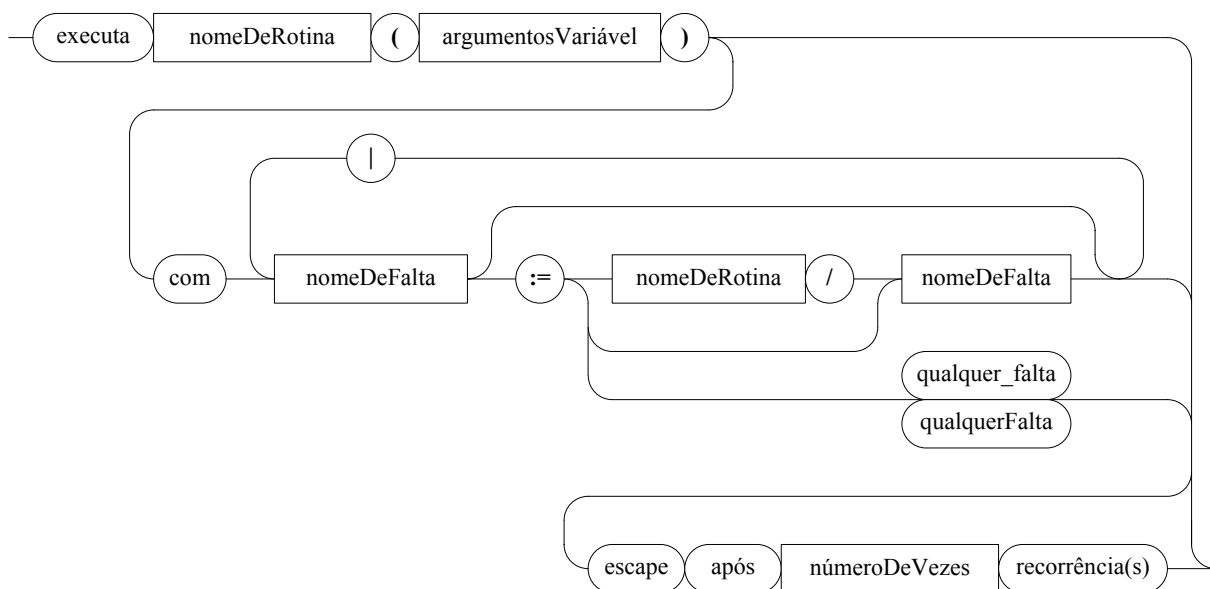
atorCedeControle (atorDeControle) -----



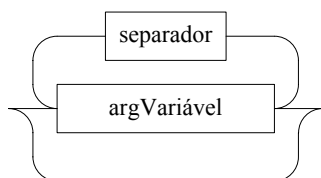
atorEncontra (atorDeControle) -----



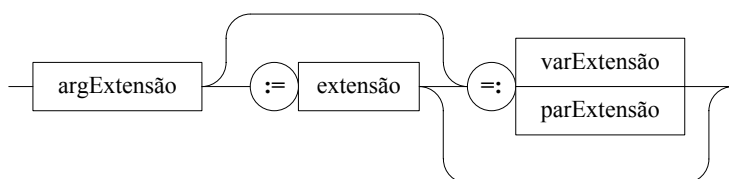
atorExecuta (atorDeControle) -----

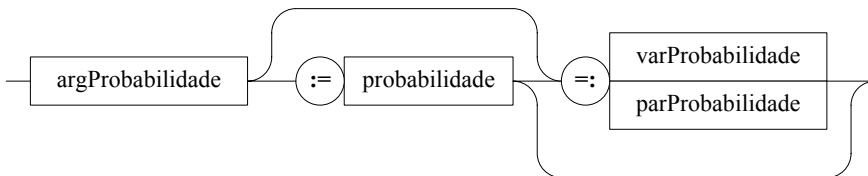
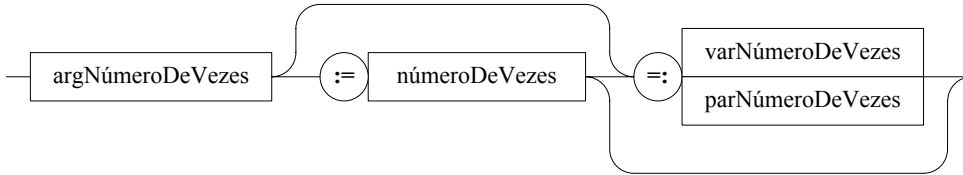
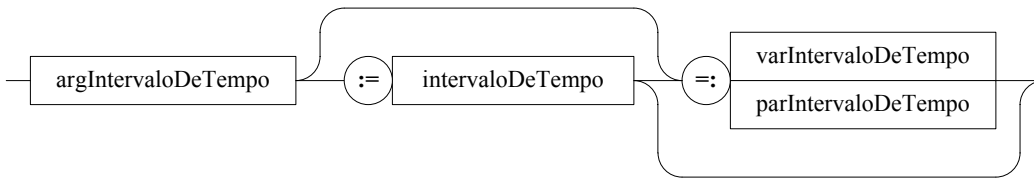


argumentosVariável -----

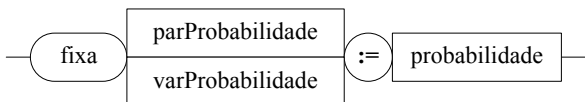
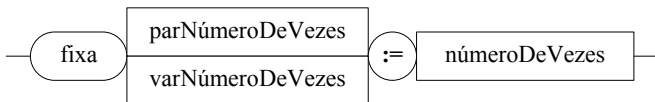
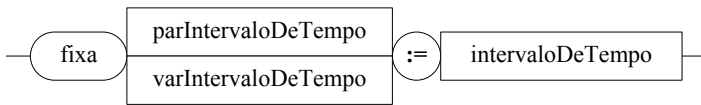
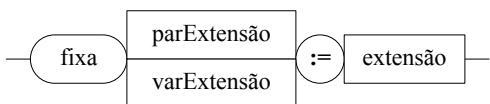


argVariável -----





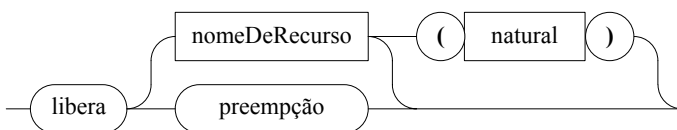
atorFixa (atorDeControle) -----



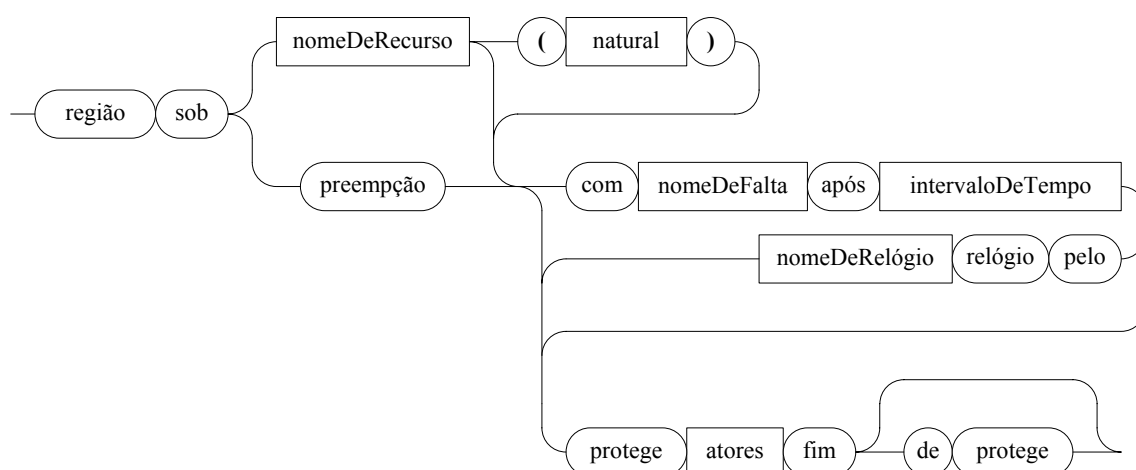
atorItera (atorDeControle) -----



atorLibera (atorDeControle) -----



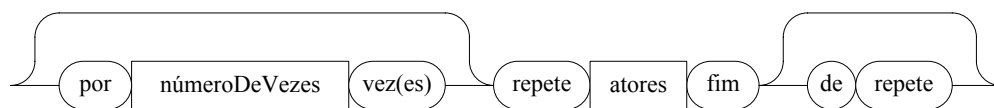
atorProtege (atorDeControle)-----



atorProvocaFalta (atorDeControle)-----



atorRepete (atorDeControle)-----



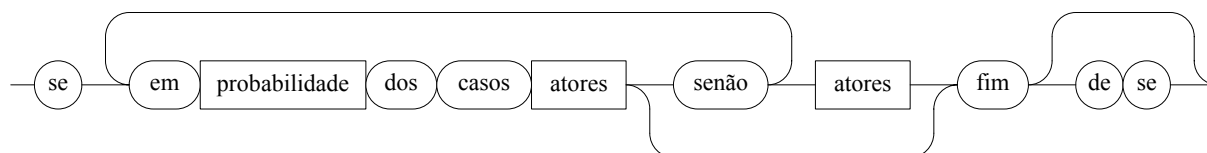
atorRetarda (atorDeAção)-----



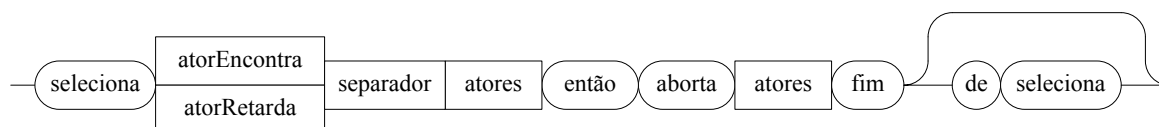
atorSalta (atorDeControle)-----



atorSe (atorDeControle)-----

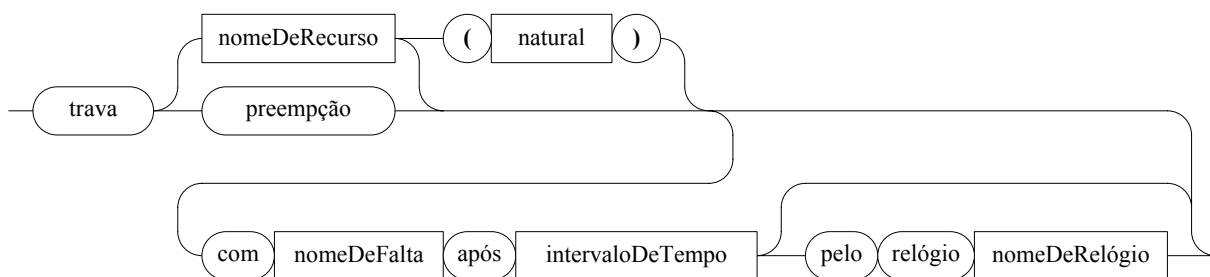


atorSelecionaAssíncrono (atorDeControle)-----

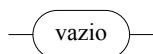




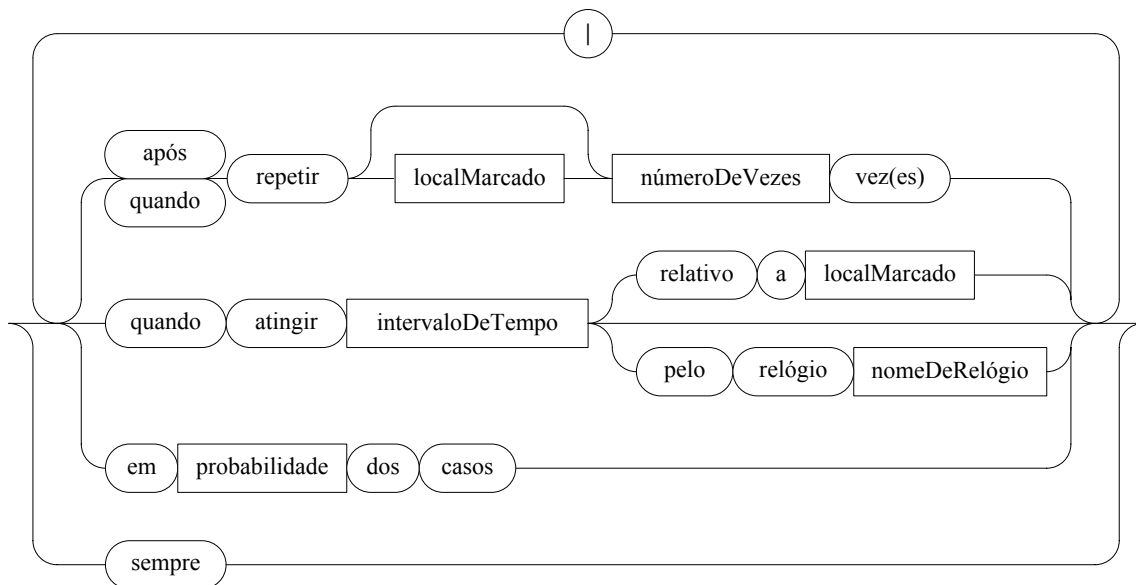
atorTrava (atorDeControle)-----



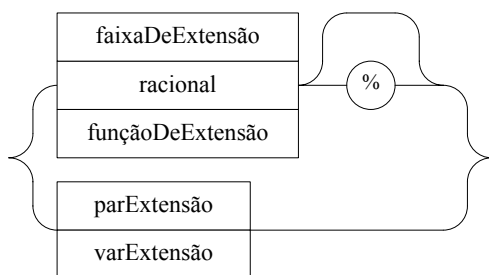
atorVazio (atorDeControle)-----



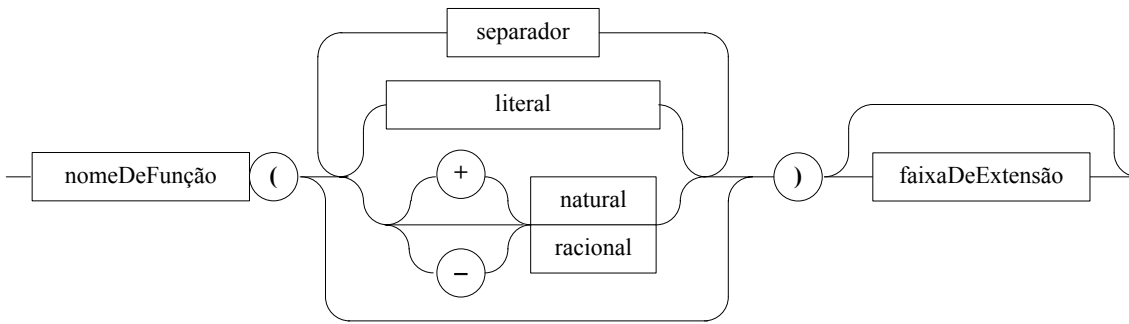
guardasDeEscape, guardas-----



extensão -----



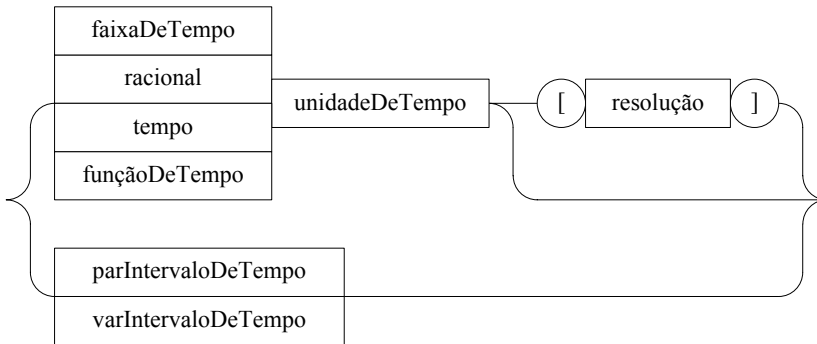
funçãoDeExtensão



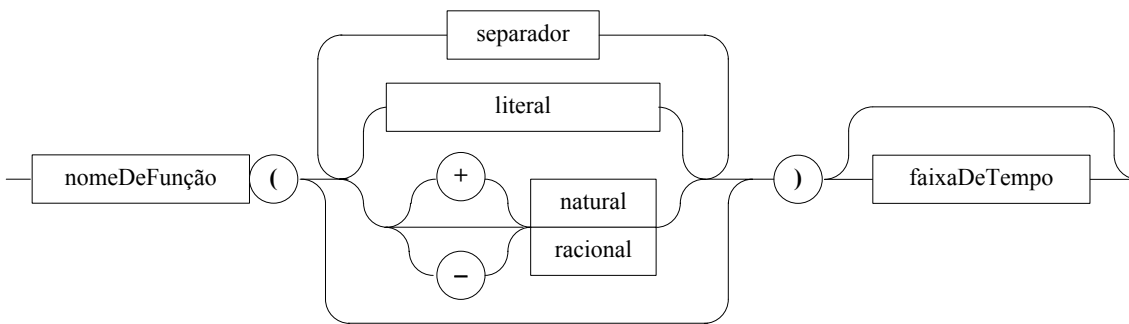
faixaDeExtensão



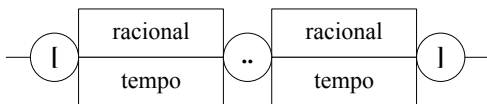
intervaloDeTempo



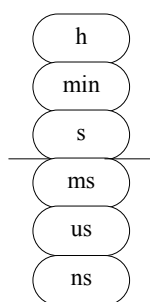
funçãoDeTempo



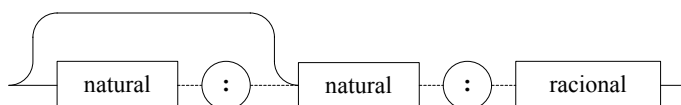
faixaDeTempo



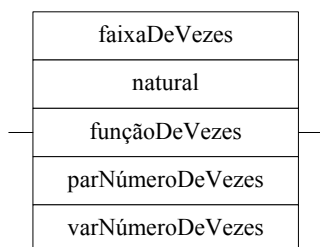
unidadeDeTempo -----



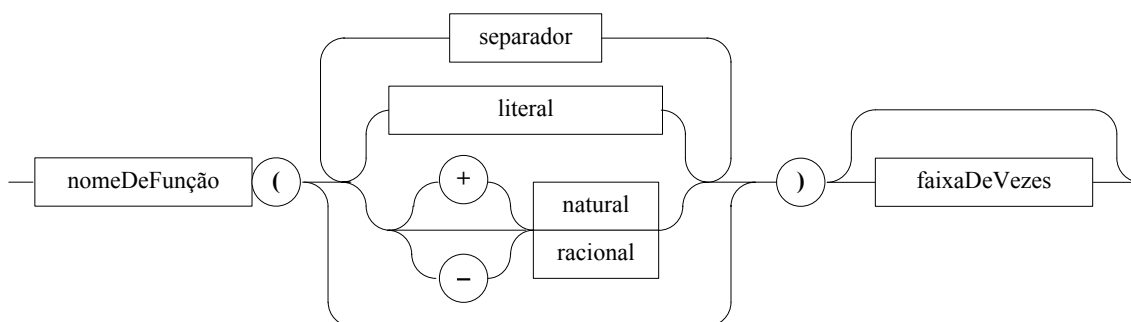
tempo -----



númeroDeVezes -----



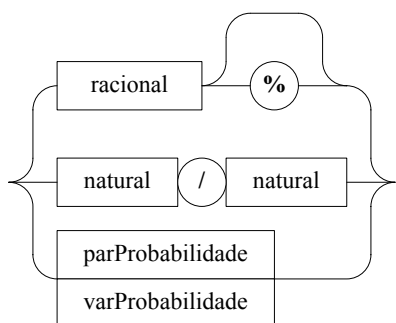
funçãoDeVezes -----



faixaDeVezes -----



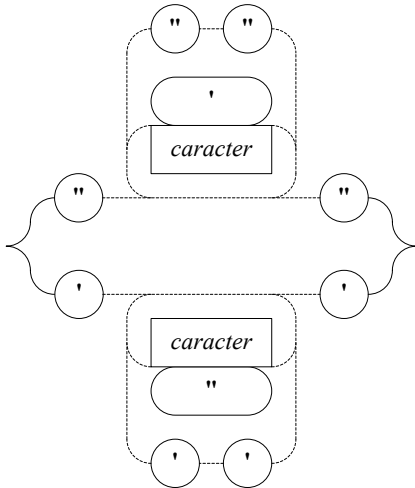
probabilidade -----



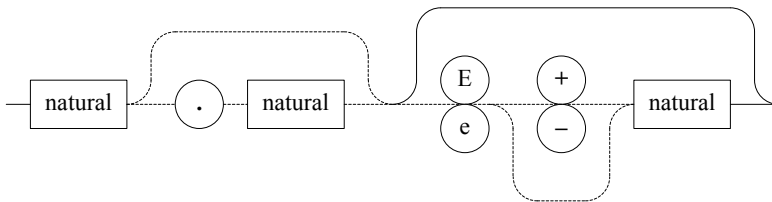
localMarcado-----



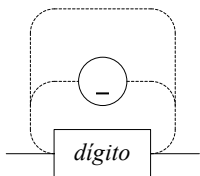
literal, nomeDeArquivo-----



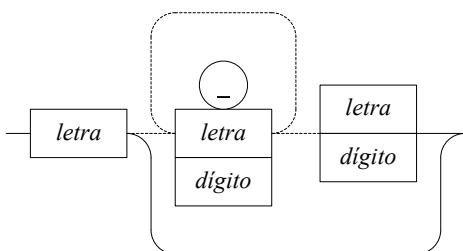
racional, resolução-----



natural-----

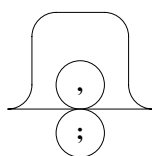


argExtensão, argIntervaloDeTempo, argNúmeroDeVezes, argProbabilidade,  
 nomeDeEncontro, nomeDeFalta, nomeDeFunção, nomeDeMarcador,  
 nomeDeProcessador, nomeDeRecurso, nomeDeRelógio, nomeDeRotina,  
 nomeDeTarefa, parExtensão, parIntervaloDeTempo, parNúmeroDeVezes,  
 parProbabilidade, varExtensão, varIntervaloDeTempo, varNúmeroDeVezes,  
 varProbabilidade-----



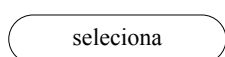


separador -----

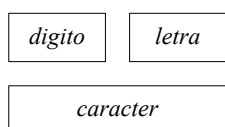


### A.3 Notação para grafos de sintaxe

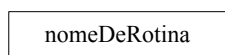
A definição da sintaxe da linguagem *AnimaTi* baseia-se em esquemas de construção montados a partir de símbolos e grafos dirigidos. Esquemas são identificados por um ou mais nomes. Os nomes podem identificar grupos de esquemas alternativos, por exemplo, *parâmetroDeTarefa*. A seguir descreve-se os diversos símbolos, esquemas e arcos utilizados:



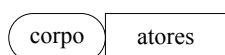
a) símbolo terminal, que produz uma instância literal de sua inscrição; produções são sensíveis a maiúsculo e minúsculo;



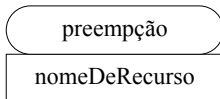
b) símbolo não-terminal informal, que produz uma instância do esquema inscrito: *dígito* para 0..9, *letra* para a..z, A..Z com adornos (acentos, cedilha, etc.), e *caracter* para qualquer carácter gráfico ASCII, exceto aspa e plica;



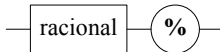
c) símbolo não-terminal formal, que produz uma instância do esquema inscrito;



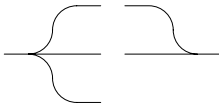
d) esquema que produz uma seqüência pela concatenação de uma instância de cada símbolo; a ordem de leitura dos símbolos depende da posição do esquema no grafo; as cadeias de símbolos podem ser arbitrariamente longas;



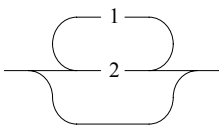
e) esquema que produz uma instância de um e somente um dos símbolos; as cadeias verticais de símbolos podem ser arbitrariamente longas;



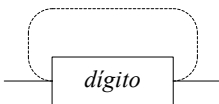
f) esquema representado por um grafo, que produz uma seqüência pela concatenação de uma instância de cada símbolo; a leitura dos símbolos ocorre naturalmente da esquerda para a direita, a partir do arco de entrada até ao arco de saída; arcos ligam símbolos entre si;



g) figuras de arcos que representam a bifurcação e a junção, respectivamente; representam início e término de produções opcionais e alternativas;



h) figura de arcos que representam afluxo e efluxo em uma iteração (na parte superior), e a possibilidade de zero iterações (na parte inferior); para evitar conflitos na ordem de leitura dos símbolos, cada arco possui somente um sentido; a leitura em 1 ocorre da direita para a esquerda, e em 2 da esquerda para a direita;



i) instâncias dos símbolos interligados por arcos tracejados devem ser concatenadas sem espaços; os caracteres ( ) | / . [ ] < > << >> ; , . : + - % := =: não requerem espaços antes e/ou depois; instâncias dos símbolos "racional" e "tempo" podem ser concatenadas a instâncias do símbolo "unidadeDeTempo" sem espaços; as demais concatenações requerem espaços.

Não há restrições quanto à distribuição de frases nas linhas do texto. Comentários podem ser inseridos em qualquer ponto. Iniciam com os caracteres "--", e estendem-se

até o final da linha. Além disso devem estar separados do símbolo anterior por pelo menos um espaço.



## APÊNDICE B DEFINIÇÃO DE ARQUIVOS FONTE XML

De acordo com o Apêndice A, partes de experimentos podem ser descritos em arquivos fonte XML. A seqüência de tarefas, rotinas, recursos, processadores e encontros tem a mesma relevância nos arquivos fonte, tanto textuais, como XML. Do ponto de vista dos algoritmos de tarefas e rotina, os arquivos fonte XML representam a árvore de sintaxe com segmentação. Estes arquivos não necessitam de compilação, e sim, de verificação da semântica, no ato da carga para reconstrução da árvore de sintaxe. Nas próximas seções apresenta-se os aspectos sintáticos destes arquivos.

### B.1 Definição da sintaxe

A seguir apresenta-se a definição da sintaxe de arquivos fonte XML, utilizando uma notação EBNF (*Extended Bakus-Naur Form*), definida na próxima seção.

<u>fonteXML</u>	::= { arquivo   encontro   processador   recurso   rotina   tarefa }
<u>arquivo</u>	::= nome=nomeDeArquivo
<u>encontro</u>	::= nome=nomeDeEncontro [ <i>assíncrono</i> =natural ] { parâmetro<1> }
<u>processador</u>	::= nome=nomeDeProcessador [ <i>baseDeTempo</i> =racional ] [ <i>multiplicidade</i> =natural ] [ <i>políticaDeFila</i> =( 'HPF'   'PIP'   'PCP'   'HL' ) ]
<u>recurso</u>	::= nome=nomeDeRecurso [ <i>multiplicidade</i> =natural ] [ <i>políticaDeFila</i> =( 'HPF'   'PIP'   'PCP'   'HL' ) ]
<u>rotina</u>	::= nome=nomeDeRotina { parâmetro<1> } { variável } [ segmento ]
<u>tarefa</u>	::= nome=nomeDeTarefa { parâmetro<2> } { variável } [ segmento ]
<u>segmento</u>	::= marcadores { salto } { ator }

ator ::= nome='abandona' [ *localMarcado*=nomeDeMarcador ] { guarda }  
 | nome='aceitaEncontro' *encontro*=nomeDeEncontro [ geraFalta<1> ] { segmento }  
 | nome='anotaRelógio' *relógio*=nomeDeRelógio [ *senal*=( '+' | '-' ) intervaloDeTempo ]  
 | nome='ativaTarefa' *tarefa*=nomeDeTarefa  
 | nome='bifurca' { para }<sup>+</sup>  
 | nome='bloco' { segmento } { captura } [ epílogo ]  
 | nome='caso' { caso }<sup>+</sup>  
 | nome='cedeControle' { guarda }  
 | nome='encontra' *encontro*=nomeDeEncontro { argVariável } [ geraFalta<1> ]  
 | nome='executa' *rotina*=nomeDeRotina [ númeroDeVezes ] { argVariável } { geraFalta<2> }  
 | nome='fixa' *alvo*=( varExtensão | parExtensão ) extensão  
 | nome='fixa' *alvo*=( varIntervaloDeTempo | parIntervaloDeTempo ) intervaloDeTempo  
 | nome='fixa' *alvo*=( varNúmeroDeVezes | parNúmeroDeVezes ) númeroDeVezes  
 | nome='fixa' *alvo*=( varProbabilidade | parProbabilidade ) probabilidade  
 | nome='itera' [ *localMarcado*=nomeDeMarcador ] { guarda }  
 | nome='libera' bloqueio  
 | nome='protege' bloqueio [ geraFalta<1> ] { segmento }  
 | nome='provocaFalta' *falta*=nomeDeFalta { guarda }  
 | nome='repete' [ númeroDeVezes ] { segmento }  
 | nome='retarda' [ *relógio*=nomeDeRelógio ] intervaloDeTempo  
 | nome='se' { caso }<sup>+</sup>  
 | nome='seleciona' *modo*='assíncrono' { caso }<sup>2</sup>  
 | nome='seleciona' *modo*='caso' { caso }<sup>+</sup>  
 | nome='seleciona' *modo*='condicional' { caso }<sup>2</sup>  
 | nome='seleciona' *modo*='temporizado' { caso }<sup>2</sup>  
 | nome='termina' [ *falta*=nomeDeFalta ] { guarda }  
 | nome='trabalha' intervaloDeTempo { geraFalta<3> }  
 | nome='trava' bloqueio [ geraFalta<1> ]  
 | nome='vazio'

argVariável ::= nome=argExtensão [ extensão ] [ *retorno*=( varExtensão | parExtensão ) ]  
 | nome=argIntervaloDeTempo [ intervaloDeTempo ]  
 [ *retorno*=( varIntervaloDeTempo | parIntervaloDeTempo ) ]  
 | nome=argNúmeroDeVezes [ númeroDeVezes ]  
 [ *retorno*=( varNúmeroDeVezes | parNúmeroDeVezes ) ]  
 | nome=argProbabilidade [ probabilidade ]  
 [ *retorno*=( varProbabilidade | parProbabilidade ) ]

bloqueio ::= *recurso*=nomeDeRecurso [ *quantidade*=natural ] | *recurso*='preempção'

captura ::= ( { *faltai*=nomeDeFalta } | *falta*='qualquerFalta' ) { segmento }

<u>caso</u>	::= [ probabilidade ] { segmento }
<u>epílogo</u>	::= { segmento }
<u>extensão</u>	::= <i>mínimo</i> =racional <i>máximo</i> =racional [ <i>leitura</i> ='% ' ]   <i>exato</i> =racional [ <i>leitura</i> ='% ' ]   função [ <i>mínimo</i> =racional <i>máximo</i> =racional ] [ <i>leitura</i> ='% ' ]   <i>variável</i> =( varExtensão   parExtensão )
<u>função</u>	::= <i>função</i> =nomeDeFunção { <i>argi</i> =argValor }
<u>geraFalta</u> <1>	::= <i>falta</i> =nomeDeFalta [ <i>relógio</i> =nomeDeRelógio ] intervaloDeTempo
<u>geraFalta</u> <2>	::= <i>falta</i> =nomeDeFalta ( <i>origem</i> ='qualquerFalta'   <i>origem</i> =nomeDeFalta [ <i>rotinaOrigem</i> =nomeDeRotina ] )
<u>geraFalta</u> <3>	::= <i>falta</i> =nomeDeFalta [ <i>modo</i> ='retro' ] probabilidade ( extensão   intervaloDeTempo )
<u>guarda</u>	::= <i>tipo</i> =( 'exato'   'após' ) [ <i>localMarcado</i> =nomeDeMarcador ] númeroDeVezes   <i>tipo</i> ='tempo' [ <i>relógio</i> =nomeDeRelógio ] intervaloDeTempo   <i>tipo</i> ='tempo' [ <i>localMarcado</i> =nomeDeMarcador ] intervaloDeTempo   <i>tipo</i> ='em' probabilidade
<u>intervaloDeTempo</u>	::= <i>mínimo</i> =( racional   tempo ) <i>máximo</i> =( racional   tempo ) unidadeResolução   <i>exato</i> =( racional   tempo ) unidadeResolução   função [ <i>mínimo</i> =( racional   tempo ) <i>máximo</i> =( racional   tempo ) ] unidadeResolução   <i>variável</i> =( varIntervaloDeTempo   parIntervaloDeTempo )
<u>marcadores</u>	::= { <i>marcadori</i> =nomeDeMarcador }
<u>númeroDeVezes</u>	::= <i>mínimo</i> =natural <i>máximo</i> =natural   <i>exato</i> =natural   função [ <i>mínimo</i> =natural <i>máximo</i> =natural ]   <i>variável</i> =( varNúmeroDeVezes   parNúmeroDeVezes )
<u>para</u>	::= [ probabilidade ] <i>localMarcado</i> =nomeDeMarcador
<u>parâmetro</u> <1>	::= <i>nome</i> =parExtensão <i>tipo</i> ='extensão' [ extensão ]   <i>nome</i> =parIntervaloDeTempo <i>tipo</i> ='intervaloDeTempo' [ intervaloDeTempo ]   <i>nome</i> =parNúmeroDeVezes <i>tipo</i> ='númeroDeVezes' [ númeroDeVezes ]   <i>nome</i> =parProbabilidade <i>tipo</i> ='probabilidade' [ probabilidade ]

<u>parâmetro</u> <2>	<pre> ::= nome='deslocamento' valor=racional unidade=unidadeDeTempo   nome='flutuação' valor=racional unidade=unidadeDeTempo   nome='modo' valor=( 'preemptivo'   'nãoPreemptivo' )   nome='período' valor=racional unidade=unidadeDeTempo   nome='período' tarefa=nomeDeTarefa   nome='pontualidade' valor=( 'fraca'   'real'   'rígida'   'severa' )   nome='prazo' valor=racional unidade=unidadeDeTempo   nome='precede' { tarefa=nomeDeTarefa }+   nome='prioridade' tarefa=nomeDeTarefa   nome='prioridade' tarefa=nomeDeTarefa sinal=( '+'   '-' ) valor=natural   nome='prioridade' valor=natural   nome='processador' valor=nomeDeProcessador   nome='tipo' valor=( 'periódico'   'aperiódico'   'esporádico'   'acoplado'   'persistente' ) </pre>
<u>probabilidade</u>	<pre> ::= exato=racional [ leitura='%']   para=natural entre=natural   variável=( varProbabilidade   parProbabilidade ) </pre>
<u>salto</u>	<pre> ::= { salto=nomeDeMarcado } { guarda } </pre>
<u>unidadeResolução</u>	<pre> ::= unidade=unidadeDeTempo [ resolução=racional ] </pre>
<u>variável</u>	<pre> ::= nome=varExtensão tipo='extensão' [ extensão ]   nome=varIntervaloDeTempo tipo='intervaloDeTempo' [ intervaloDeTempo ]   nome=varNúmeroDeVezes tipo='númeroDeVezes' [ númeroDeVezes ]   nome=varProbabilidade tipo='probabilidade' [ probabilidade ] </pre>

Os tipos de valores referenciados na definição da sintaxe de arquivos fonte XML estão definidos no Apêndice A, a saber: `argExtensão`, `argIntervaloDeTempo`, `argNúmeroDeVezes`, `argProbabilidade`, `nomeDeEncontro`, `nomeDeFalta`, `nomeDeFunção`, `nomeDeMarcador`, `nomeDeProcessador`, `nomeDeRecurso`, `nomeDeRelógio`, `nomeDeRotina`, `nomeDeTarefa`, `parExtensão`, `parIntervaloDeTempo`, `parNúmeroDeVezes`, `parProbabilidade`, `unidadeDeTempo`, `varExtensão`, `varIntervaloDeTempo`, `varNúmeroDeVezes`, `varProbabilidade`, `natural`, `racional`, e `tempo`.

O tipo de valores `nomeDeArquivo` é formado por caracteres, e o tipo de valores `argValor` pode ser formado por caracteres, números naturais e racionais com sinal.



## B.2 Notação EBNF

A definição da sintaxe dos arquivos fonte XML baseia-se em esquemas de construção de elementos e seus atributos. Cada elemento possui uma lista opcional de atributos, e uma lista opcional de elementos aninhados. A seguir descreve-se a meta-linguagem utilizada:

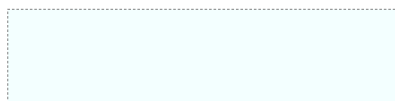
<u>nome</u> ::= expr	definição da expressão singular <u>nome</u> por expr, representando o elemento <nome/>; o resultado de expr não pode conter nomes de atributos repetidos; as ocorrências de elementos em expr formam uma lista ordenada;
nome ::= expr	definição da expressão singular nome por expr; permite isolar e/ou nomear partes comuns da definição de sintaxe;
expr	expressão formada por uma lista de expressões singulares;
[ expr ]	expressão formada pela ocorrência opcional de expr;
{ expr }	expressão formada por zero ou mais ocorrências de expr;
{ expr } <sup>k</sup>	expressão formada por <i>k</i> ocorrências de expr; + indica uma ou mais ocorrências; e <i>i</i> indica um número exato de ocorrências;
expr1   expr2	expressão formada ou por expr1 ou expr2;
( expr )	expressão singular formada por expr;
nome=tipo	expressão singular para representação do atributo <i>nome</i> , instanciado por um valor do conjunto tipo;

<i>nome</i> ='valor'	expressão singular para representação do atributo <i>nome</i> , instanciado literalmente por 'valor';
<i>nome</i> =(tipo   'valor')	expressão singular que equivale à ( <i>nome</i> =tipo   <i>nome</i> ='valor'); para um número arbitrário de opções de diversos tipos e valores;
<i>nome</i> <i>i</i>	nome de atributo instanciado para <i>nome</i> <sub>1</sub> , <i>nome</i> <sub>2</sub> , ... <i>nome</i> <sub>N</sub> em expressões com repetição, na resolução de definições;
<u>nome</u> <q>	representação de elementos com o mesmo nome e definições sintáticas distintas; <q> pode ser qualquer valor.

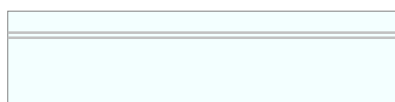
Os nomes de elementos e atributos devem ser formados por caracteres sem adornos (acentos, cedilha, etc.), embora estejam presentes na definição da sintaxe dos arquivos fonte XML. A restrição deve-se ao fato dos aplicativos DOM (*Document Object Model*) apresentarem problemas com tais caracteres.

## APÊNDICE C ELEMENTOS DO DIAGRAMA DE SEGMENTOS

Neste apêndice descreve-se as figuras que compõem os diagramas de segmentos. Tarefas e rotinas possuem grafos de segmentos independentes. A lista a seguir relaciona elementos e esquemas gerais para construção dos grafos:



reserva de um espaço que pode conter um segmento, ou um grafo de segmentos; atores complexos possuem destes espaços aninhados;



reserva de um espaço para fechamento do segmento anterior, por exemplo, no atorSeleciona..., que pode ser parte inicial de um grafo de segmentos;



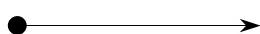
elementos que representam marcadores, apostos aos segmentos; o elemento tracejado indica um marcador inatingível;



elementos que representam pontos de captura de faltas, apostos aos segmentos de tratamento das faltas; o elemento tracejado indica uma captura inatingível;



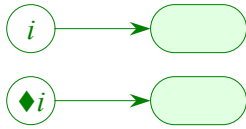
elementos que representam arcos entre segmentos; indicam o fluxo de controle na simulação de tarefas e rotinas (traço cheio e tracejado), e o fluxo de controle na análise de tempos (traço cheio e traço-ponto-traço);



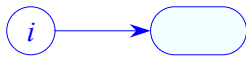
elemento que representa o ponto de partida em um grafo de segmentos;



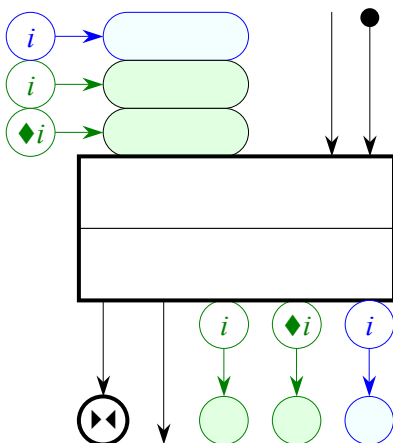
elemento que representa arcos inviáveis, de qualquer tipo: traço cheio, tracejado e traço-ponto-traço;



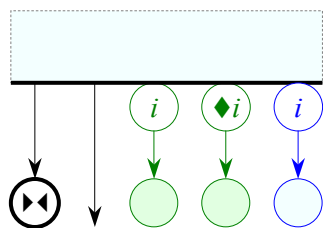
esquemas que representam saltos gerados pelo ator *Salta* ao final de segmentos, partindo em direção a marcadores; são formados por: a) elemento apostro a segmentos que indica o salto de número *i*; b) arco; e c) marcador de destino; no primeiro caso, o salto é incondicional e no segundo caso é condicionado a guardas;



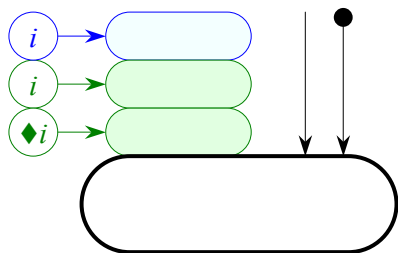
esquema que representa saltos gerados por faltas ao final de segmentos, partindo em direção a capturas de faltas; é formado por: a) elemento apostro a segmentos que indica a falta de número *i*; b) arco; e c) ponto de captura da falta;



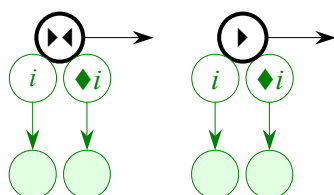
esquema que representa segmentos formados por zero ou mais atores simples, em que somente o último ator pode gerar faltas, escapes, saltos, ou términos; além disso, os segmentos podem ser nominados por um ou mais marcadores; o controle afluí para os segmentos através de arcos diretos, ou por esquemas de saltos, e efluí dos segmentos por arcos diretos, saltos, escapes, términos, e faltas geradas pelo último ator dos segmentos; o primeiro segmento de capturas de faltas constitui um caso especial, possui elementos de faltas, e não possui arcos diretos de afluí de controle;



esquema que representa o fechamento do espaço de segmentos duplos, ou seja, o primeiro segmento deste espaço possui as mesmas características de efluxo de controle dos segmentos de atores simples;



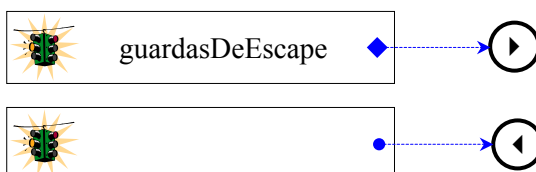
esquema que representa os segmentos iniciais de atores complexos; atores complexos são desdobrados em dois: a) segmentos iniciais que distribuem o fluxo de controle segundo suas estruturas; e b) segmentos terminais que juntam novamente o fluxo num ponto; o afluxo de controle tem as mesmas características dos segmentos de atores simples;



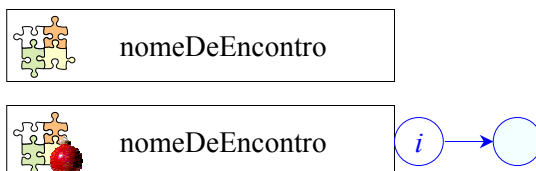
esquemas que representam segmentos terminais de atores complexos; possuem efluxo de controle por arcos diretos e saltos; o esquema a esquerda junta o fluxo de casos (atorSe e outros), e o esquema a direita junta o fluxo de repetição (atorRepete).










































No Apêndice A apresenta-se a definição da linguagem *AnimaTi*, que complementa a compreensão dos esquemas dos atores. Na lista a seguir descreve-se os esquemas para os atores individualmente:

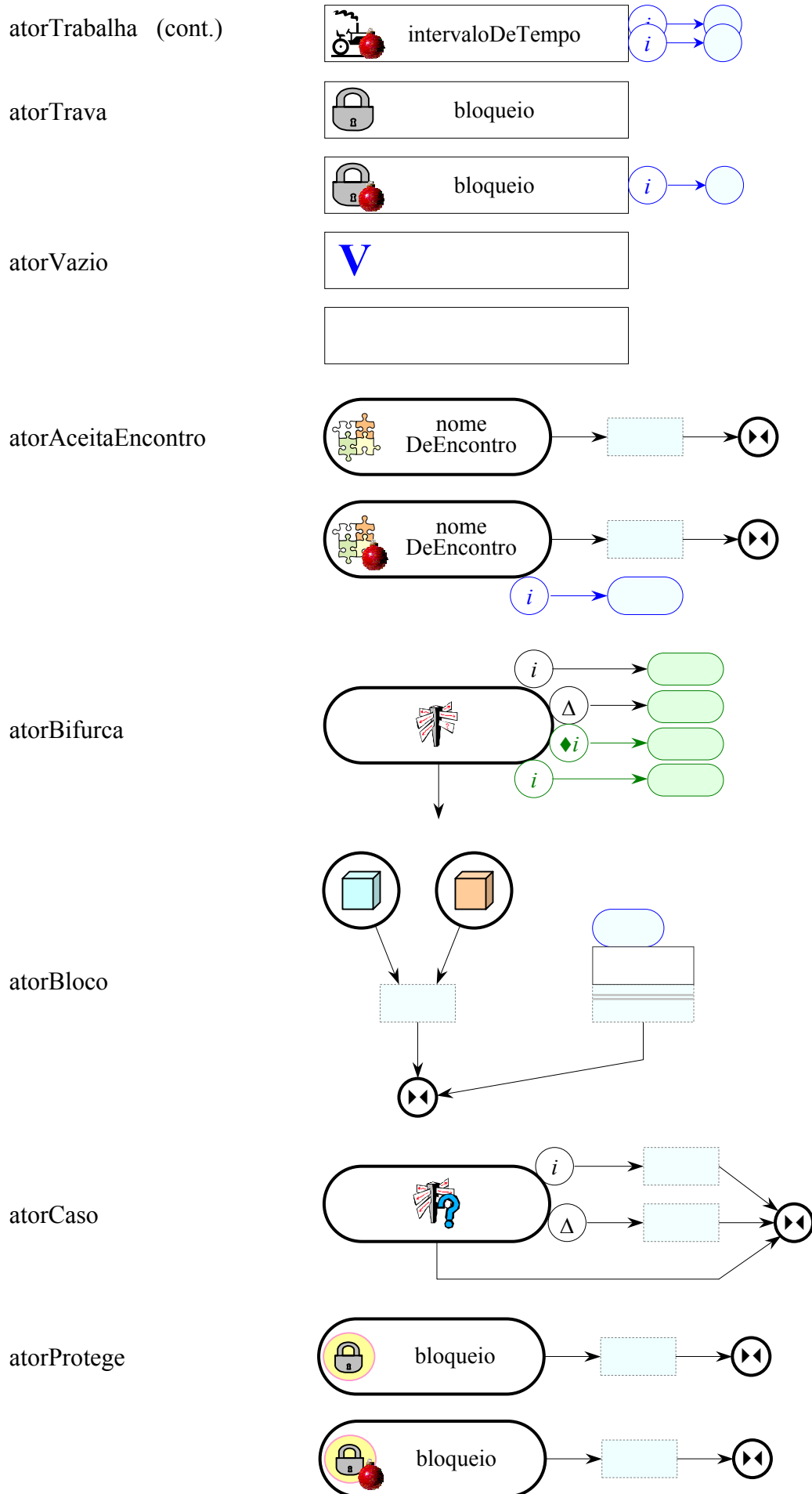
atorAbandona



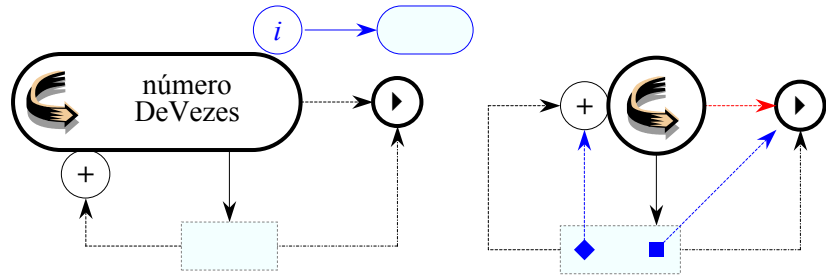
atorAceitaEncontro



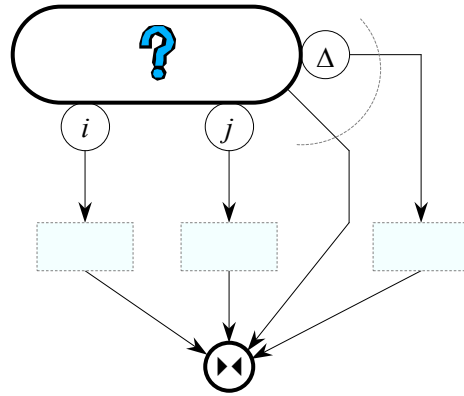
atorAnotaRelógio	 anotaçãoDeTempo	
atorAtivaTarefa	 tarefa	
atorCedeControle	 guardas 	
		
atorEncontra	 nomeDeEncontro	
	 nomeDeEncontro  → 	
atorExecuta	 rotina	
	 rotina  → 	
atorFixa	 variável	
atorItera	 guardasDeEscape  → 	
	 	
atorLibera	 bloqueio	
atorProvocaFalta	 falta; guardas   → 	
	 falta  → 	
atorRetarda	 intervaloDeTempo	
atorTermina	 guardas  → 	
	 guardas  → 	
	  → 	
	  → 	
atorTrabalha	 intervaloDeTempo	



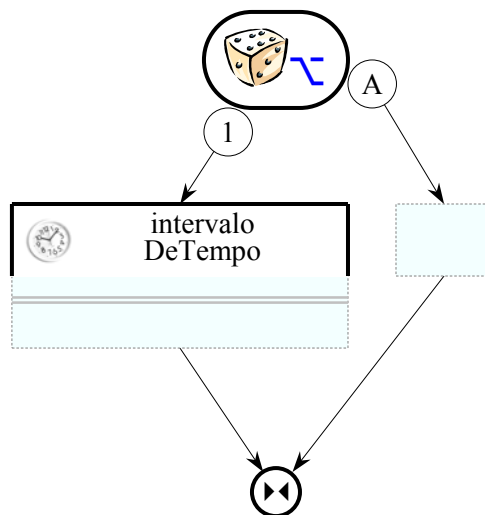
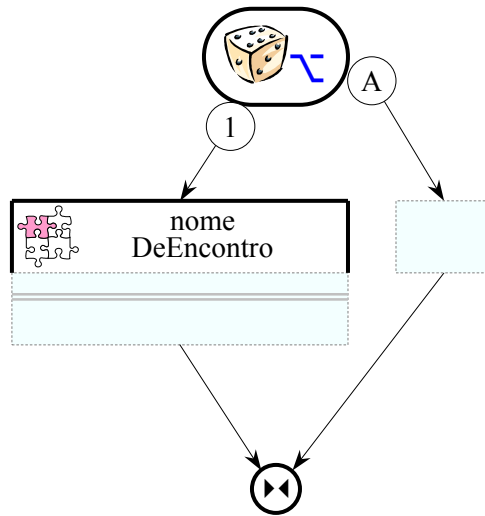
atorRepete



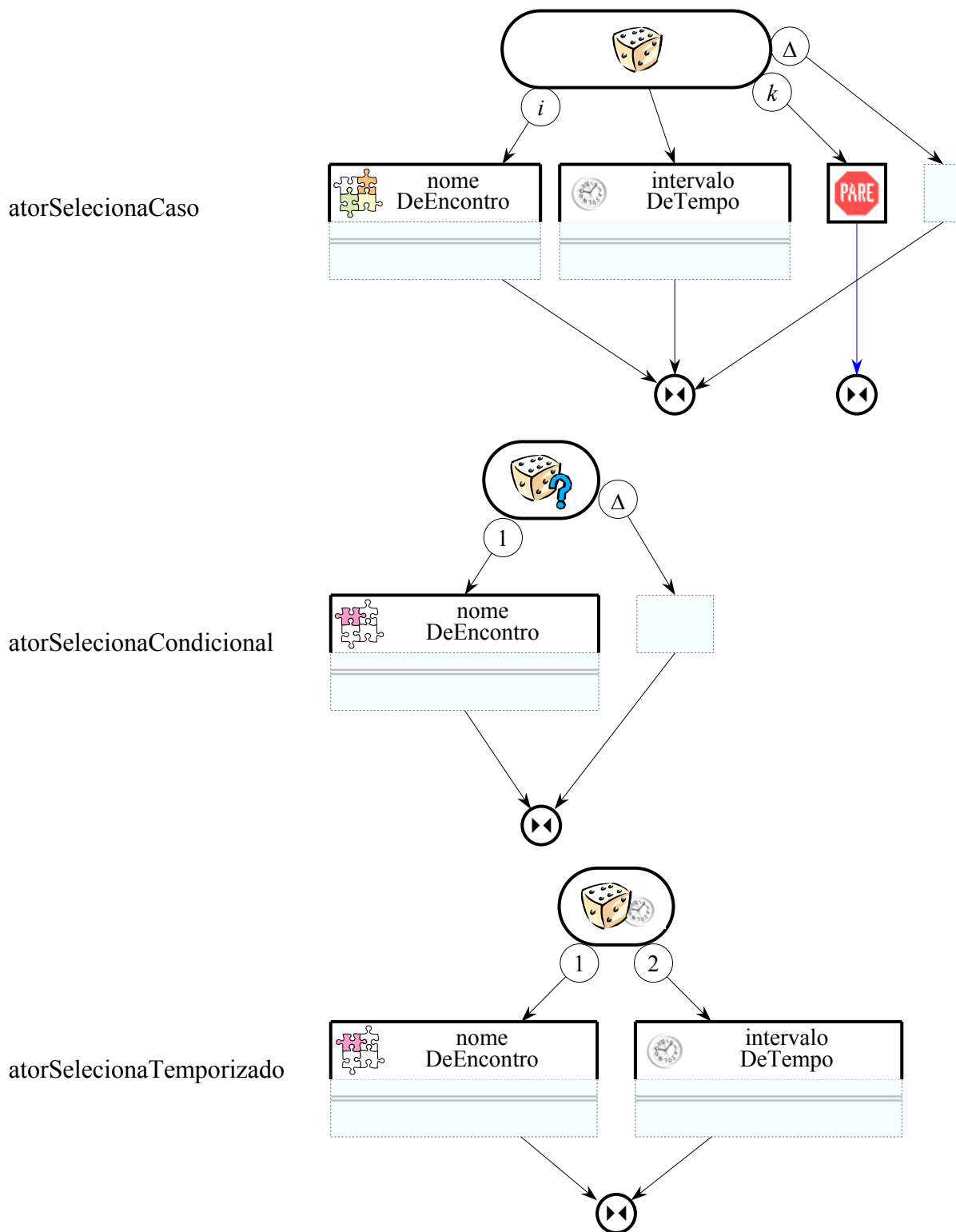
atorSe



atorSelecionaAssíncrono





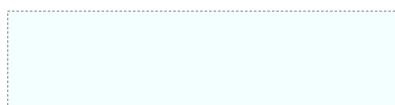


Alguns atores possuem inscrições com: o símbolo  $\blacklozenge$  que significa que os atores estão associados a guardas e devem executar somente em determinados casos, e o símbolo  $\Delta$  que aparece em atores que selecionam um entre diversos casos e representa os demais casos, por exemplo, a cláusula *senão* do atorCaso.



## APÊNDICE D ELEMENTOS DO DIAGRAMA DE NS

Neste apêndice descreve-se as figuras que compõem os diagramas estruturados Nassi-Shneiderman. Os diagramas contêm elementos de grafos para preservar os atores de salto da linguagem *AnimaTi*. Tarefas e rotinas possuem diagramas NS independentes. A lista a seguir relaciona elementos e esquemas gerais dos diagramas:



reserva de um espaço que pode conter um segmento, ou um grafo de segmentos; atores complexos possuem destes espaços aninhados;



elementos que representam marcadores, apostos aos segmentos; o elemento tracejado indica um marcador inatingível;



elemento que representa arcos entre segmentos, que indicam o fluxo de controle;



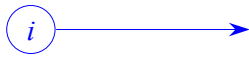
elemento que representa o ponto de partida em um grafo de segmentos;



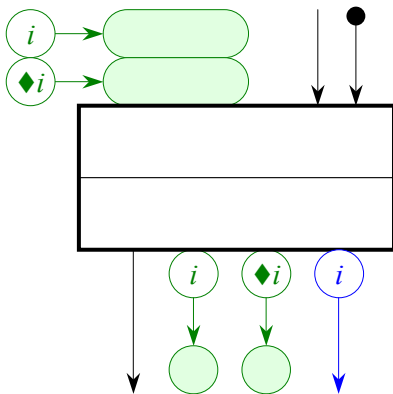
elemento que representa arcos de fluxo de controle inviáveis;



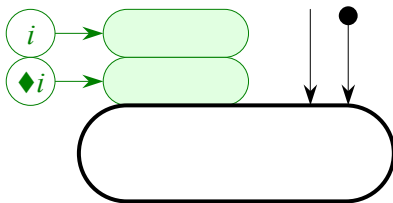
esquemas que representam saltos gerados pelo ator *atorSalta*, partindo em direção a marcadores; são formados por: a) elemento apostos a segmentos para indicar o salto de número  $i$ ; b) arco; e c) marcador de destino; no primeiro caso, o salto é incondicional e no segundo caso está condicionado a guardas;



esquema que representa saltos gerados por faltas ao final de segmentos, partindo em direção a capturas de faltas; é formado por: a) elemento apostro a segmentos que indica a falta de número *i*; e b) arco que leva ao bloco da captura da falta;



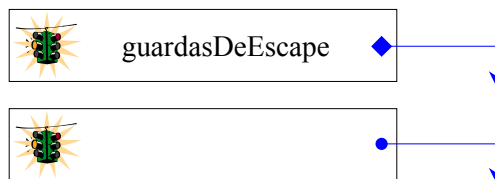
esquema que representa segmentos formados por zero ou mais atores simples em que somente o último ator pode gerar faltas, escapes, saltos, ou terminos; além disso, os segmentos podem ser nominados por um ou mais marcadores; o controle afluí para os segmentos através de arcos diretos ou por esquemas de saltos e efluí dos segmentos por arcos diretos, saltos, escapes, e faltas geradas pelo último ator dos segmentos;



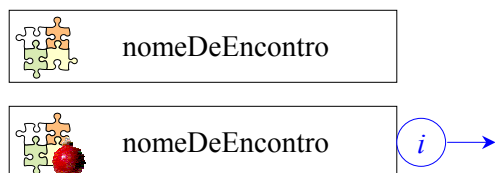
esquema que representa o segmento formado pelo ator atorBifurca; o segmento pode ser nominado por um ou mais marcadores; o controle afluí para o segmento através de arcos diretos ou por esquemas de saltos;





















No Apêndice A apresenta-se a definição da linguagem *AnimaTi*, que complementa a compreensão dos esquemas dos atores. Na lista a seguir descreve-se os esquemas para os atores individualmente:

atorAbandona



atorAceitaEncontro

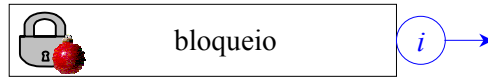
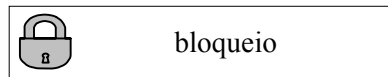


atorAnotaRelógio	 anotaçãoDeTempo	
atorAtivaTarefa	 tarefa	
atorCedeControle	 guardas	◆
		
atorEncontra	 nomeDeEncontro	
	 nomeDeEncontro	<i>i</i> →
atorExecuta	 rotina	
	 rotina	<i>i</i> → <i>i</i> →
atorFixa	 variável	
atorItera	 guardasDeEscape	◆ →
		● →
atorLibera	 bloqueio	
atorProvocaFalta	 falta; guardas	◆ <i>i</i> →
	 falta	<i>i</i> →
atorRetarda	 intervaloDeTempo	
atorTermina	 guardas	◆
	 guardas	◆
		
		
atorTrabalha	 intervaloDeTempo	

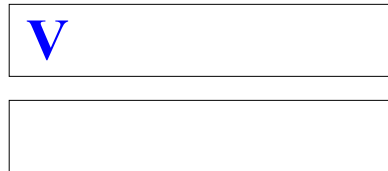
atorTrabalha (cont.)



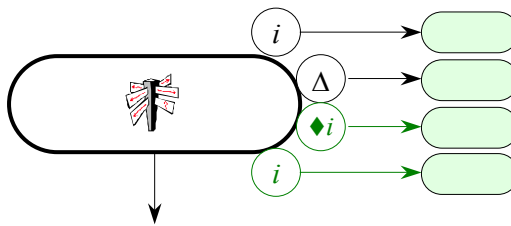
atorTrava



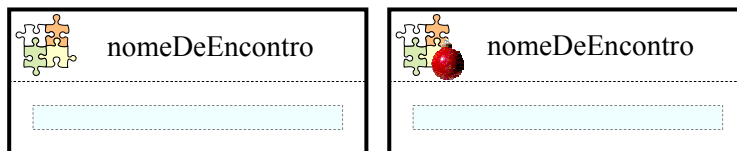
atorVazio



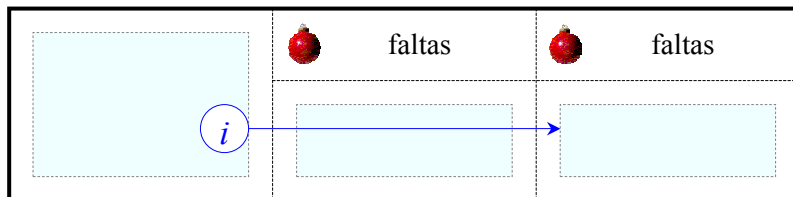
atorBifurca



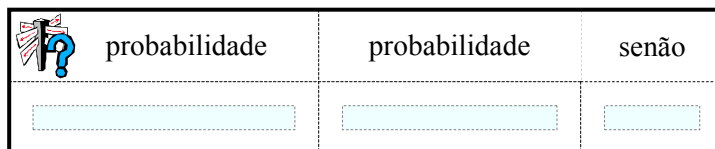
atorAceitaEncontro



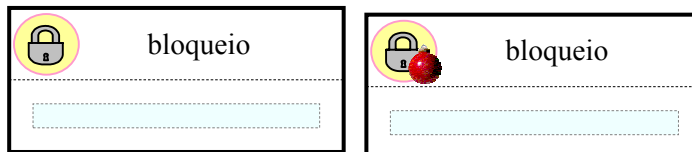
atorBloco



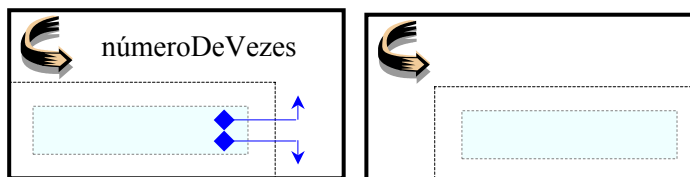
atorCaso



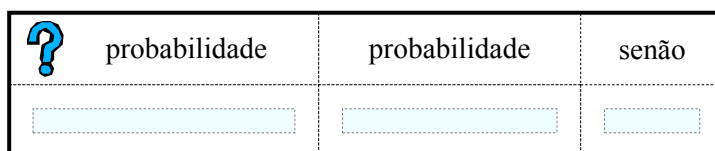
atorProtege






atorRepete



atorSe



atorSelecionaAssíncrono



		aborta
	nome DeEncontro	<input type="text"/>
<input type="text"/>		<input type="text"/>

		aborta
	intervalo DeTempo	<input type="text"/>
<input type="text"/>		<input type="text"/>

atorSelecionaCaso


	probabilidade			senão
	nome DeEncontro		intervalo DeTempo	
<input type="text"/>		<input type="text"/>		<input type="text"/>

atorSelecionaCondicional

		senão
	nome DeEncontro	<input type="text"/>
<input type="text"/>		<input type="text"/>

atorSelecionaTemporizado

		
	nome DeEncontro	
<input type="text"/>		intervalo DeTempo
<input type="text"/>		<input type="text"/>

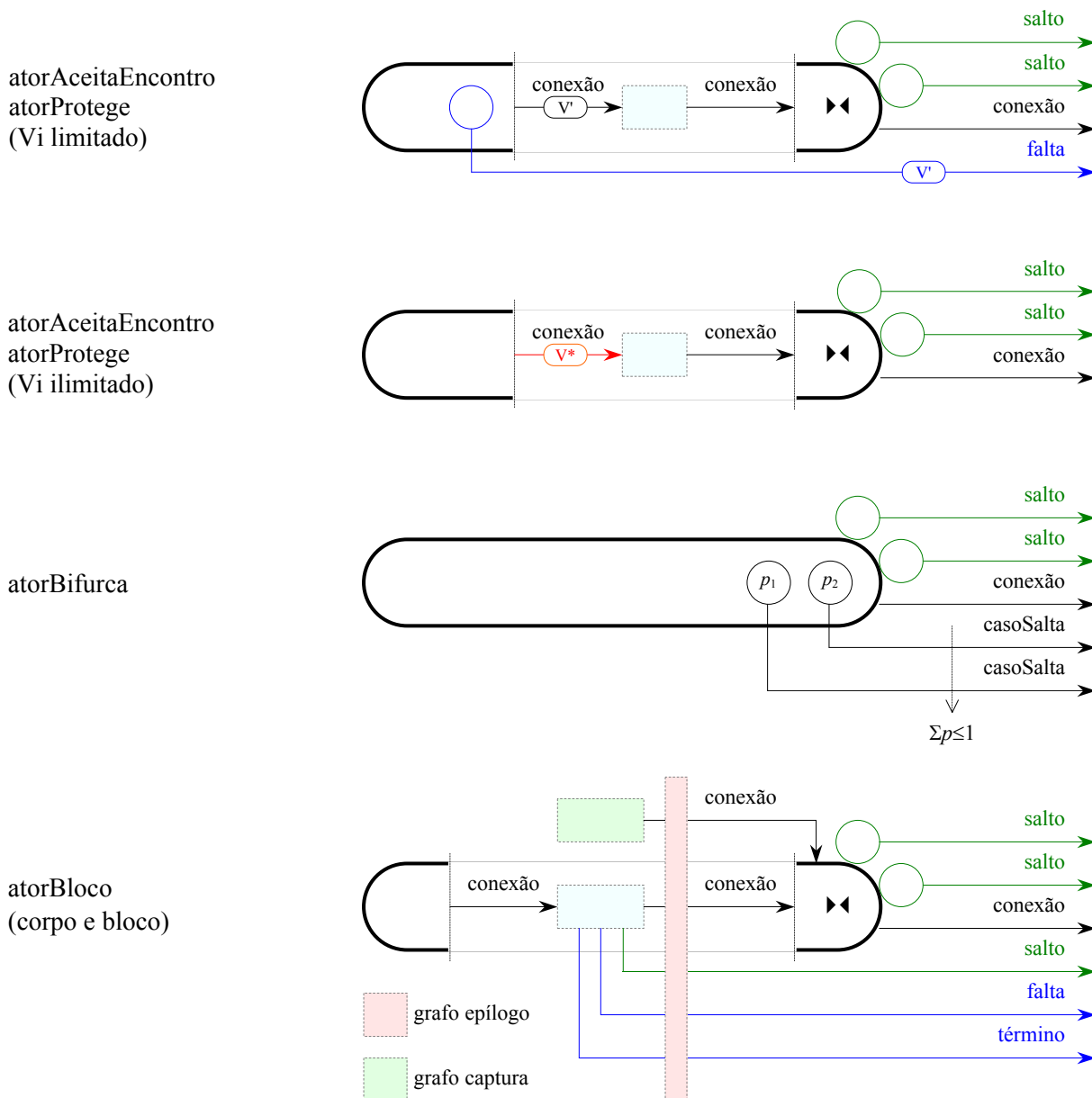
Alguns atores possuem inscrições com o símbolo  que significa que estão associados a guardas e devem executar somente em determinados casos. Tarefas e rotinas iniciam com o símbolo do ator atorBloco sem adornos que figura como corpo da tarefa ou rotina.

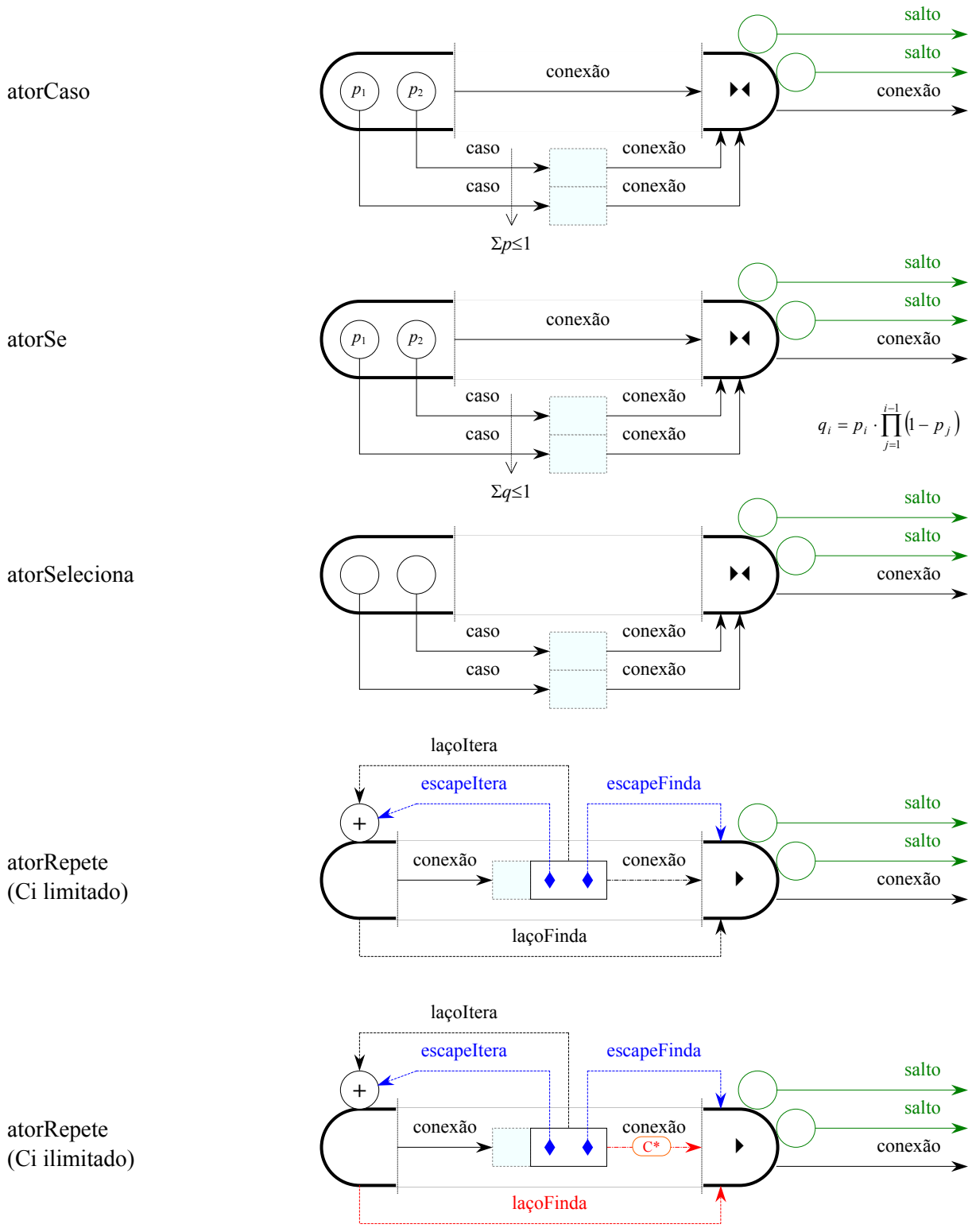




## APÊNDICE E ESQUEMAS DE NODOS PARA GRAFOS DE SEGMENTOS

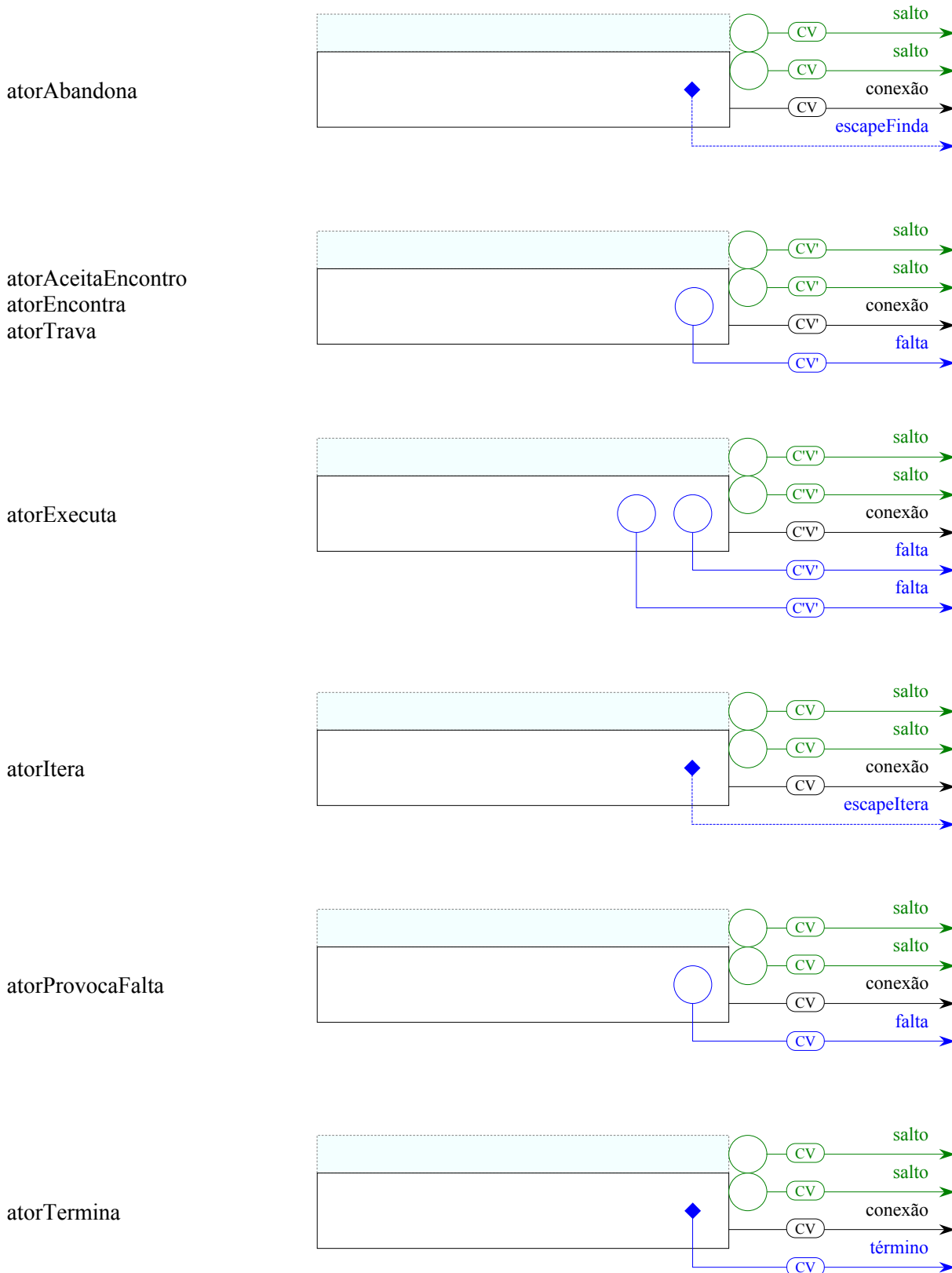
Neste apêndice descreve-se os nodos que compõem os grafos de segmentos. Os atores complexos, por exemplo, atorCaso e atorBloco, são decompostos em dois nodos: a) o nodo de entrada, que direciona o fluxo para os segmentos internos; e b) o nodo de término que recolhe o fluxo novamente. Segmentos internos podem direcionar o fluxo para outros nodos, no caso de saltos, faltas e términos. Os atores atorSalta que seguem os atores complexos são incorporados ao nodo de término. O ator atorBifurca não é decomposto em dois nodos. A lista a seguir relaciona os esquemas de nodos dos atores complexos.

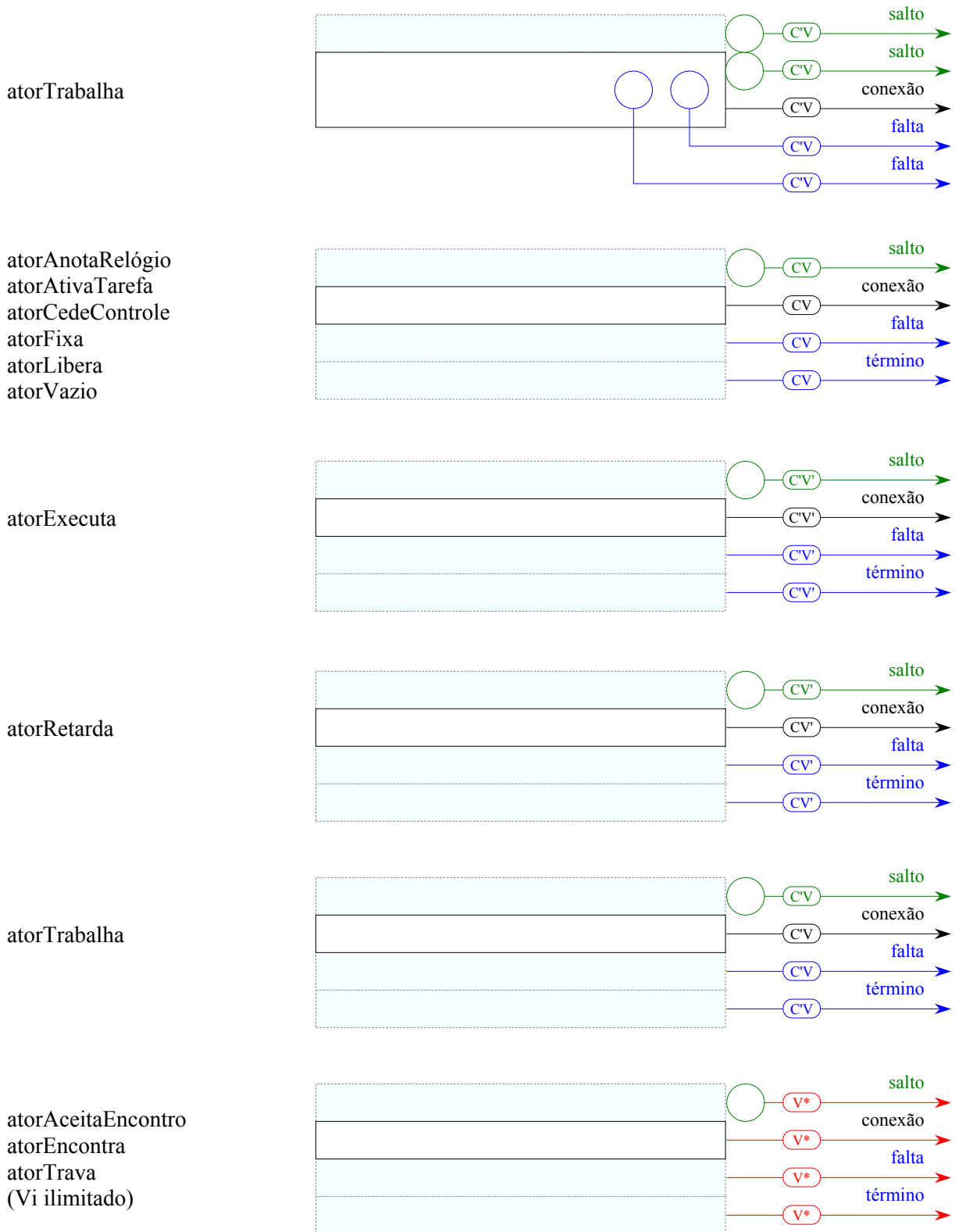




Os atores simples são agrupados para formar segmentos. Estes segmentos são nodos. A características básicas destes nodos são: a) arcos de entrada são direcionados para o primeiro ator do segmento; e b) somente o último ator do segmento pode motivar arcos de saída. Os atores atorSalta que seguem os atores simples que formam o segmento são incorporados ao nodo do segmento. A lista a seguir relaciona os

esquemas de nodos dos segmentos de atores simples. Os primeiros 7 nodos tratam os casos em que o último ator do segmento causa escapes, faltas ou término. Nos demais casos estes atores estão incorporados em qualquer posição no segmento.





Os arcos de saída de nodos representam dois fluxos distintos: a) fluxo de controle do simulador de tarefas e rotina (traço cheio e tracejado); e b) fluxo de controle na análise de tempos (traço cheio e traço-ponto-traço). Além disso, os arcos podem ser de diversos tipos: a) conexão, que indica fluxos inerentes à seqüência dos atores; b) caso,

que indica fluxos decorrentes da seleção de casos, por exemplo, no ator AtorSe; c) casoSalta, que indica fluxos decorrentes dos casos no ator atorBifurca; d) falta, que indica fluxos decorrentes de faltas, por exemplo, na cláusula "com" do ator atorTrabalha; e) término, que indica fluxos decorrentes do ator atorTermina; f) salto, que indica fluxos decorrentes do ator atorSalta; e g) laçoItera, laçoFinda, escapeItera e escapeFinda, que indicam fluxos internos ao ator atorRepete.

As anotações  $\textcircled{cv}$  sobre os arcos representam o modo de atualização dos tempos  $C_i$  e  $V_i$  pelos atores em destaque: para C e V preservam o valor e para C' e V' atualizam o valor. C\* e V\* indicam que os tempos são ilimitados. Por exemplo, o ator atorRepete pode permanecer indefinidamente em seu laço. Neste caso, a análise de tempos não consegue limitar o tempo de execução  $C_i$ . Arcos em vermelho são inviáveis.



## APÊNDICE F FERRAMENTAS PARA ANÁLISE E SIMULAÇÃO

Neste apêndice apresenta-se ferramentas para análise e simulação de escalonamento em sistemas em tempo real. Os dados foram compilados a partir de artigos publicados em revistas, dissertações de mestrado, teses de doutorado, propaganda encontrada em páginas de empresas comerciais e de instituições de ensino.

O QUADRO F.1 mostra a lista de ferramentas examinadas juntamente com a identificação de algumas de suas características.

QUADRO F.1: Características das ferramentas pesquisadas

Ferramentas	Analítico	Simulação	Real	Abstrato	Livre	Acessível	SO	Ano	Acadêmico Comercial Outros
							Linux, Unix Windows		
1 AFTER	X	--	X	--	--	--	W, U	2003	--
2 ARTISST	--	X	--	X	X	X	L	2003	O
3 ASSERTS	X	X	--	X	--	--	X	1999	C
4 CAISARTS	X	--	--	X	--	--	--	1996	A
5 CarbonKernel	--	X	X	--	X	X	L	2002	C
6 Cheddar	X	--	X	X	X	X	W, L	2006	A
7 DET/SAT/SIM	X	X	--	X	--	--	--	1998	O
8 DRTSS / PERTSSim	--	X	--	X	--	--	U	1996	A
9 MatLab TrueTime	--	X	--	--	X	X	U, L, W	2006	C
10 Perf	X	X	X	--	X	X	W	2002	A
11 PerfoRMAx	X	X	--	X	--	X	W	2006	C
12 RapidRMA	X	X	--	X	--	X	U, L, W	2006	C
13 Scheduler 1-2-3	X	X	--	--	--	--	U	1988	A
14 ScheduLite	X	--	--	X	--	--	W	1996	A
15 SEW	X	--	X	--	X	X	L, W	1999	A
16 STRESS	X	X	--	X	--	--	U	1994	A
17 TEV	X	X	X	--	X	X	W	2006	A
18 TimeWiz	X	X	X	--	--	X	W	2006	C
19 UTSA	--	X	X	--	X	X	U, L, W (Java)	2005	A
20 YASA	--	X	--	X	X	X	L, U, W	2006	A

A coluna **Analítico** indica se a ferramenta analisa os aplicativos e determina suas propriedades de escalonabilidade baseado na teoria de escalonamento. A coluna

**Simulação** indica se a ferramenta experimenta os aplicativos com cargas artificiais para estabelecer valores estatísticos para suas propriedades de escalabilidade sem garantias de exequibilidade. Por um lado, a simulação entra em campo para suprir a falta de cobertura teórica, por outro lado pode ser utilizada para visualizar a evolução temporal dos aplicativos. As colunas **Real** e **Abstrato** indicam se os aplicativos são representados através de modelos reais ou abstratos, respectivamente, por exemplo, em linguagem de programação tipo C, Java, ou através grafos de tarefas, recursos, etc. As colunas **Livre**, **Acessível**, **SO** e **Ano** indicam se a ferramenta é de livre acesso, se está de fato acessível no momento, sob que sistema operacional pode ser utilizada e até que ano foi possível detectar sua presença. A ferramenta 19 da lista está disponível através de *browsers*. A última coluna indica a origem das ferramentas. Em geral resultam de trabalhos acadêmicos, como monografias, dissertações de mestrado e teses de doutorado. Algumas evoluíram a partir de outras e passaram a ser comercializadas associadas a ambientes de desenvolvimento proprietários, tais como RapidRMA e perfoRMAx.

As próximas seções apresentam os principais aspectos das ferramentas para análise e simulação de sistemas em tempo real. As ferramentas ARTISST, Cheddar e YASA estão em destaque por serem livres, recentes e cobrirem aspectos relevantes ao presente trabalho. As demais ferramentas estão mencionadas com alguns detalhes.

A bibliografia utilizada na pesquisa das diversas ferramentas está relacionada nas Referências Bibliográficas do trabalho.

## F.1 ARTISST

A proposta do ferramental ARTISST<sup>6</sup> (DECOTIGNY; PUAUT, 2002) é de analisar e avaliar sistemas em tempo real através de simulação. Trata de sistemas cujos ambientes e/ou comportamentos no domínio do tempo não estejam totalmente

---

<sup>6</sup> ARTISST: *A Real Time System Simulation Tool*, Ferramenta para Simulação de Sistemas em Tempo Real.



caracterizados, e em que a abordagem analítica seria difícil quando não impossível. Espera que os resultados obtidos pela execução real ou simulada de sistemas sejam atingíveis e menos pessimistas, ainda que menos seguros.

Outro aspecto importante do ferramental é o de modelar o sistema em conjunto com o ambiente o mais próximo possível de seus componentes reais.

A FIGURA F.1 mostra a arquitetura básica do ARTISST composto por três módulos: a) simulador do ambiente externo; b) simulador do sistema; e c) módulo de análise, avaliação e apresentação de dados. Os módulos comunicam-se via mensagens.

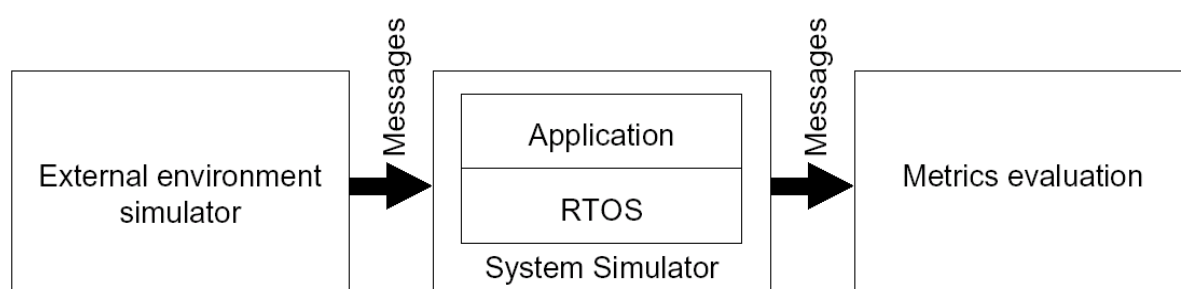


FIGURA F.1: Arquitetura básica do ferramental ARTISST

O simulador do sistema simula um aplicativo que opera sobre um sistema operacional em tempo real – RTOS. O aplicativo é formado por um conjunto de tarefas com padrões de ativação e lançamento arbitrariamente complexos e com restrições de tempo de execução e sincronização. O conjunto de tarefas interage com um ambiente arbitrário que pode mudar dinamicamente.

Em relação ao sistema operacional, possibilita configurar e adequar diversos escalonadores e serviços, e leva em consideração seus custos computacionais.

A FIGURA F.2 mostra o modelo de uma tarefa e ilustra o estado de determinado serviço com diversas marcações, tais como instante da ativação, instante da liberação, e diversas preempções ou bloqueios.

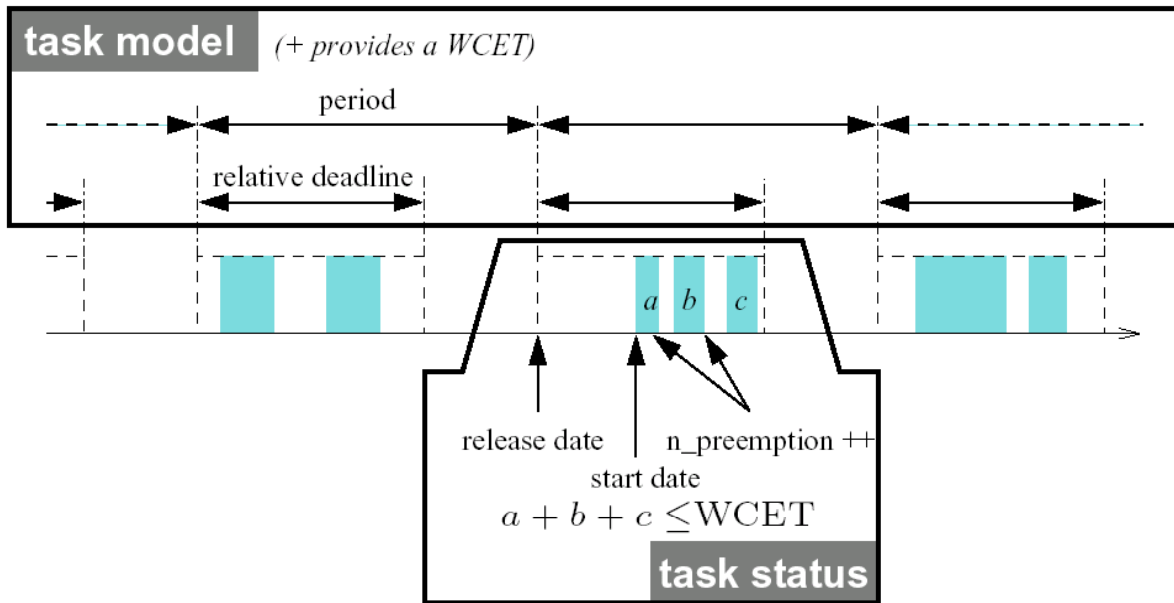


FIGURA F.2: Modelo de tarefa com estado da tarefa

Uma característica importante é a extensibilidade e capacidade de personalização dos modelos de tarefas e dos estados. Desta forma pode-se adequar os modelos às necessidades, por exemplo, do sistema operacional e escalonador simulado.

O texto à esquerda na FIGURA F.3 dá uma idéia de como os algoritmos de execução da tarefa (melhor, dos serviços da tarefa) são codificados.

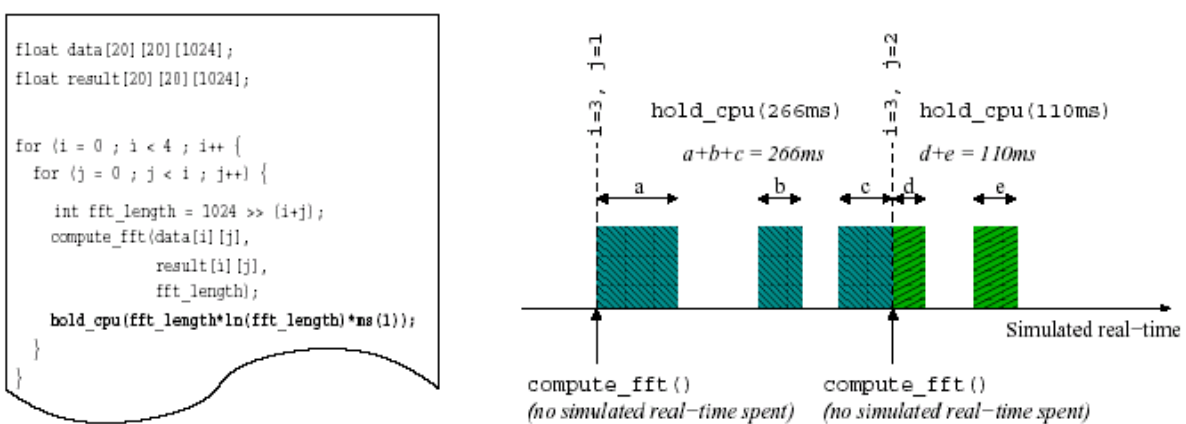


FIGURA F.3: Simulação de tempo de execução – hold\_cpu (duração)

O código fonte pode ser escrito em linguagens tradicionais, tais como C e C++. As computações descritas não têm efeito sobre o tempo de execução que é

determinado por um artifício: a chamada da função `hold_cpu` (duração) inserida em qualquer ponto nos algoritmos. O gráfico de Gantt à direita na FIGURA F.3 mostra a chamada `hold_cpu` (226ms) distribuída nos intervalos de tempo *a*, *b* e *c*, seguida pela chamada `hold_cpu` (110ms) distribuída nos intervalos de tempo *d* e *e*.

A FIGURA F.4 mostra um exemplo de estrutura modular do ARTISST baseada em módulos que geram, processam e reagem a eventos propagados de um módulo a outro. Basicamente é um simulador a eventos discretos.

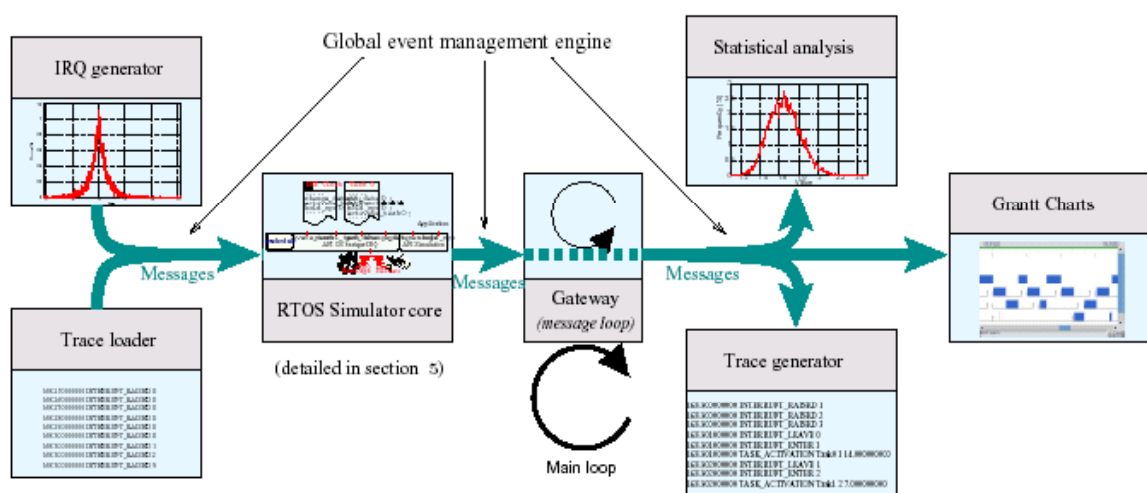


FIGURA F.4: Exemplo da estrutura modular do ARTISST

A cadeia de módulos é característica de cada aplicativo e de responsabilidade do projetista. Os eventos ou mensagens geradas nos módulos ou que fluem através dos módulos são definidos pela tripla  $\langle \text{data}, \text{tipo}, \text{parâmetros} \rangle$ . São classificados em eventos: a) de gestão de interrupções simuladas; b) de gestão de tarefas; c) de mudança de estados; e d) definidos pelo usuário.

A lista abaixo relaciona os principais módulos oferecidos pelo ARTISST que por sua vez pode ser estendida com módulos construídos pelos usuários:

- gerador de eventos aleatórios (entrada);
- gerador de eventos por histórico (entrada);
- simulador do sistema;
- gerador de históricos (saída);
- analisador estatístico (saída);

- elaborador de diagramas Gantt (saída);
- bombeador de mensagens.

A FIGURA F.5 mostra uma sessão típica de simulação de um sistema. O diagrama de Gantt mapeia os eventos que ocorrem no sistema operacional entre a preempção da tarefa 0 e a transferência do controle do processador para a tarefa 2.

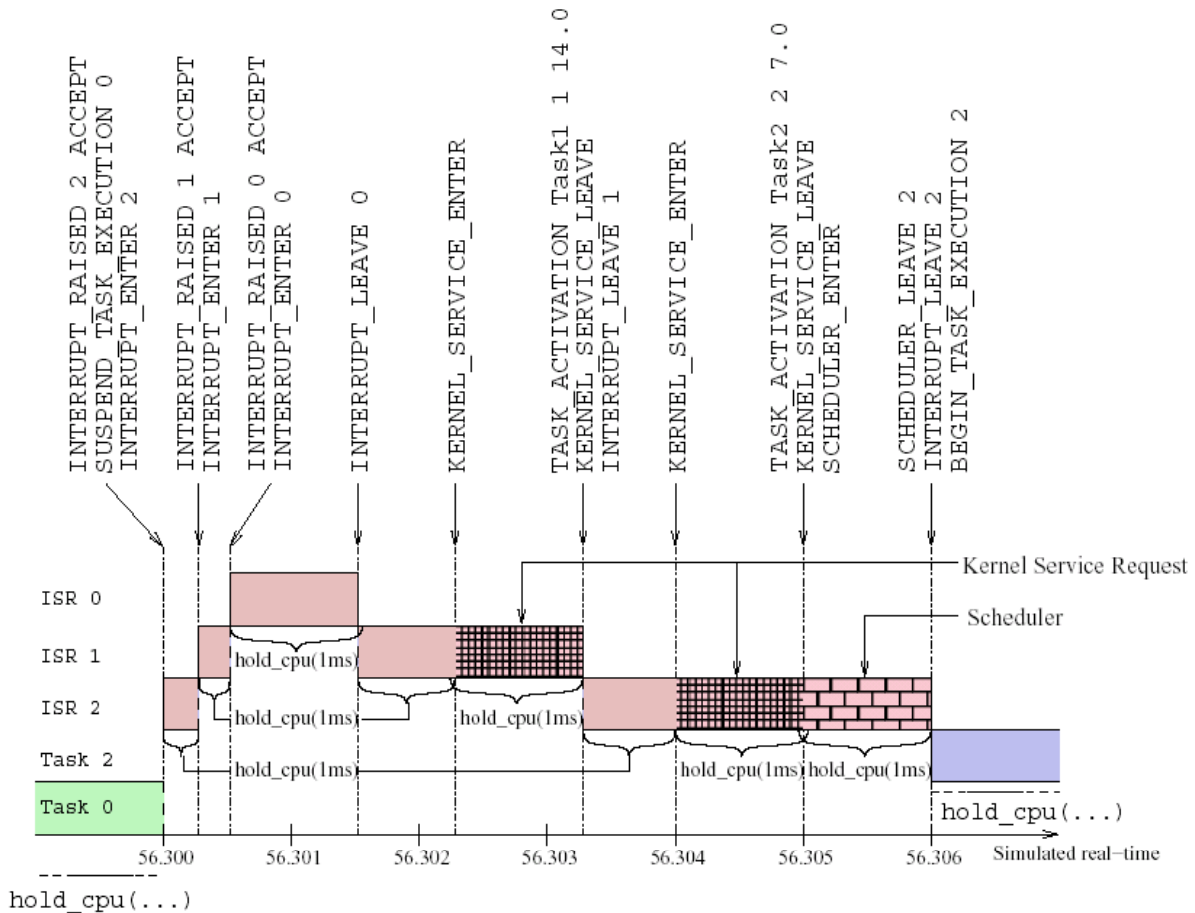


FIGURA F.5: Eventos gerados por uma sessão típica de um sistema

A ferramenta ARTISST não se enquadra na categoria de ferramentas acadêmicas. Preocupa-se com a acurácia da medição de tempos em relação aos objetos reais de estudo.

## F.2 Cheddar

O projeto Cheddar (SINGHOFF *et.alii*, 2004) propõe três requisitos de desenvolvimento de ferramental para análise de exeqüibilidade e simulação de escalonamento:

- implementação da maioria dos métodos da teoria clássica de escalonamento, incluindo testes de exeqüibilidade para sistemas com um processador e sistemas distribuídos, com a maioria de escalonadores usuais em tempo real e padrões de ativação de tarefas;
- promoção de ferramentas de uso simplificado por pessoas e estudantes sem grandes conhecimentos da teoria de escalonamento e aberto para conexão com outros programas, tais como simuladores e ferramentas CASE através de intercâmbio de dados formatados em XML;
- extensibilidade da máquina de simulação para acomodar escalonadores e padrões de escalonamento além daqueles já conhecidos, expressos através de uma linguagem tipo Ada.

Com as funcionalidades adicionadas às ferramentas, usuários e estudantes podem experimentar e entender os fundamentos da teoria de escalonamento em tempo real.

Os programas estão implementados em Ada e executam em Solaris, Linux, Windows, e em qualquer plataforma que suporta Gantt/GtkAda. São distribuídos sob a Licença Pública Geral GNU.

A FIGURA F.6 mostra a composição de um aplicativo: processadores, *buffers*, recursos compartilhados, mensagens e tarefas. Sobre o conjunto de tarefas realiza dois tipos de análises: simulação de escalonamento<sup>7</sup> e teste de exeqüibilidade.

---

<sup>7</sup> A “simulação de escalonamento” baseia-se nos parâmetros das tarefas, ao contrário da “simulação de tarefas”, que se baseia em algoritmos que descrevem o comportamento temporal e casual das tarefas.

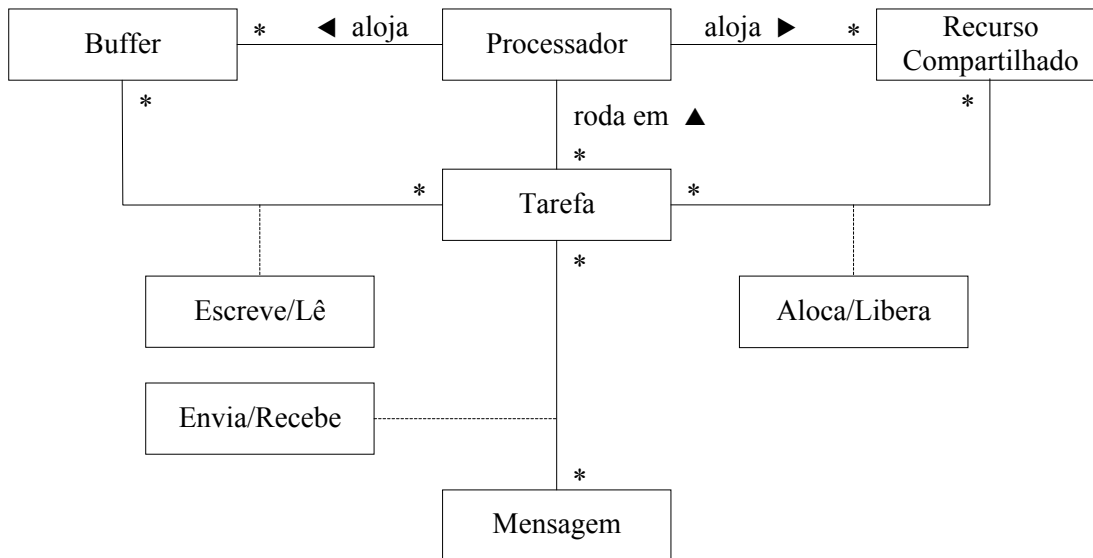


FIGURA F.6: Diagrama UML de um aplicativo modelado com Cheddar

A FIGURA F.7 mostra um conjunto de três tarefas:  $T_1\langle 10,3,5\rangle$ ,  $T_2\langle 20,8,20\rangle$  e  $T_3\langle 35,7,30\rangle$  com os parâmetros: período, tempo de computação e prazo, nesta ordem.

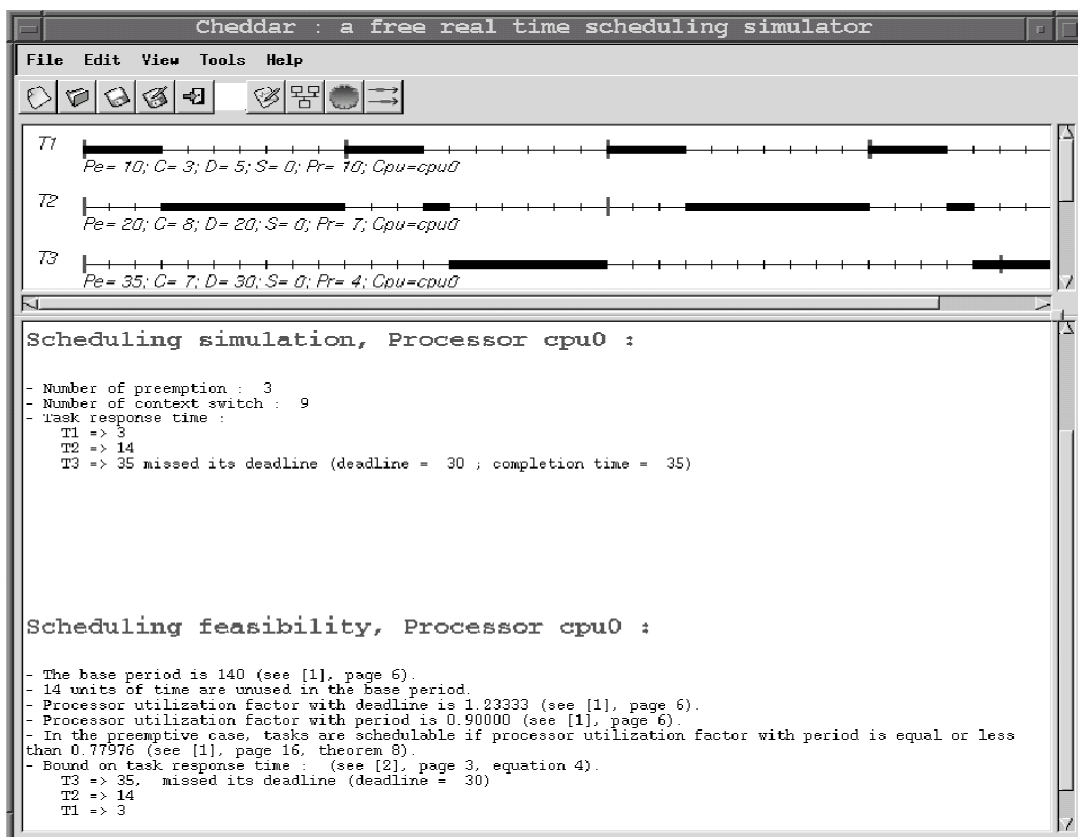


FIGURA F.7: Janela principal do editor do Cheddar

O quadro superior da figura mostra o diagrama de Gantt da simulação de escalonamento, e o quadro inferior mostra o resultado tanto da simulação como do teste de exequibilidade.

A FIGURA F.8 dá uma idéia da implementação da máquina de simulação. O processo de simulação é composto por três etapas:

- computação do escalonamento para cada unidade de tempo, e a determinação dos eventos que devem ocorrer, entre eles disparo de tarefas, alocação e/ou liberação de recursos compartilhados, escrita e leitura de *buffers* e envio e recebimento de mensagens;
- análise dos eventos para determinar as propriedades do aplicativo em estudo através de analisadores predefinidos e outros acrescentados pelo usuário;
- exibição dos resultados na janela principal do Cheddar.

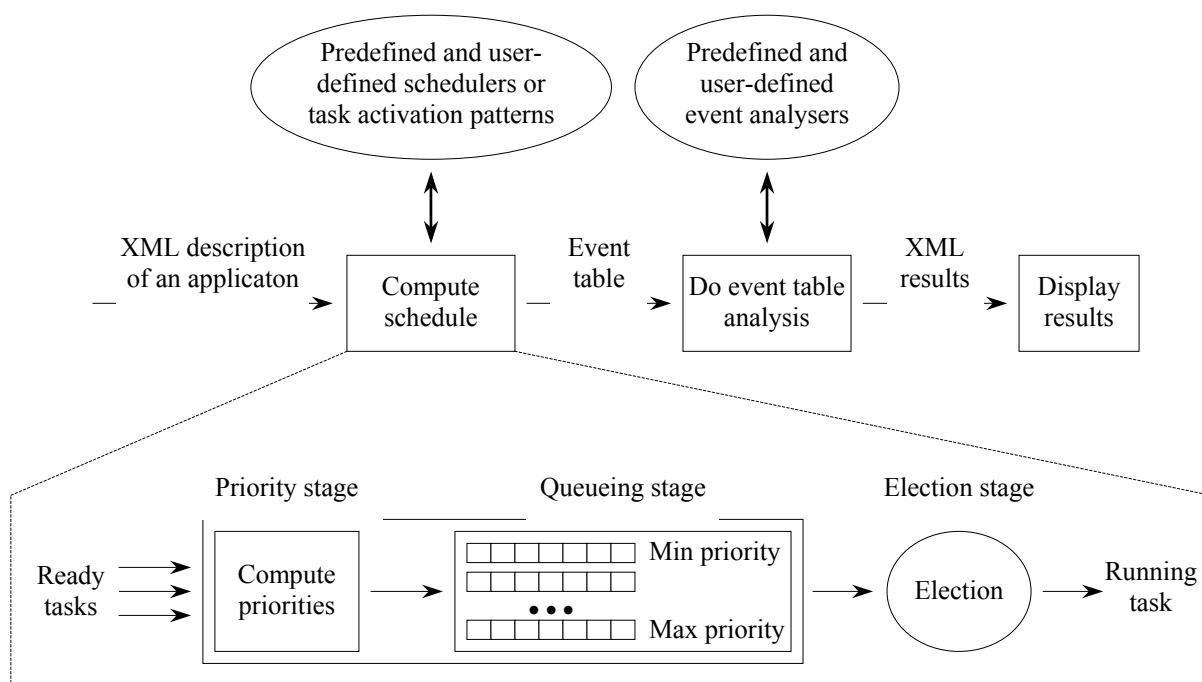


FIGURA F.8: Modelo da máquina do simulador

A etapa de computação do escalonamento é implementada por um escalonador que recebe uma lista de tarefas no estado PRONTO e escolhe uma destas para entrar no estado EXECUTANDO. Escalonadores operam em três estágios:

- estágio de priorização em que cada tarefa recebe uma prioridade;
- estágio de enfileiramento em que as tarefas são enfileiradas por prioridade, usando a política POSIX;
- estágio de escolha em que a primeira tarefa da fila com maior prioridade recebe o processador por uma unidade de tempo.

Os escalonadores usuais já vêm predefinidos. No entanto, qualquer um dos estágios pode ser redefinido pelo usuário através de uma linguagem própria tipo Ada, como mostra o ALGORITMO F.1.

### ALGORITMO F.1: Exemplo de definição de um escalonador

---

```

start_section:
  partition_duration : array (tasks_range) of integer;
  dynamic_priority : array (tasks_range) of integer;
  number_of_partition : integer := 2;
  current : integer := 0;
  time_partition : integer := 0;
  - the partition scheduling table
  partition_duration(0) := 2;
  partition_duration(1) := 4;
  time_partition := partition_duration(current);
priority_section:
  if time_partition=0 then
    current := (current+1) mod number_of_partition;
    time_partition := partition_duration(current);
  end if;
  - choose the task with the highest priority
  - owned by the active partition
  for i in tasks_range loop
    if tasks.task_partition(i)=current then
      dynamic_priority(i) := tasks.priority(i);
    else
      dynamic_priority(i) := 0;
      tasks.ready(i) := false;
    end if;
  end loop;
  time_partition := time_partition-1;
election_section:
  return max_to_index(dynamic_priority);

```

---



### F.3 YASA

O ferramental YASA<sup>8</sup> (BLUMENTHAL, 2002) (BLUMENTHAL *et alii*, 2002) opera por simulação virtual e execução em ambientes reais. Propõe-se a avaliar um conjunto de tarefas com respeito a escalonabilidade e garantia de restrições em tempo de execução sob diferentes algoritmos de escalonamento.

Em especial trata de métodos de escalonamento com prioridade dinâmica: EDF, LLF e ELLF. Para absorver o impacto computacional destes algoritmos, propõe-se o uso de escalonamento suportado por co-processador. Segundo os autores do YASA, estas tecnologias alcançam bom desempenho, requerendo, no entanto, análises cuidadosas para garantir a pontualidade de um conjunto de tarefas.

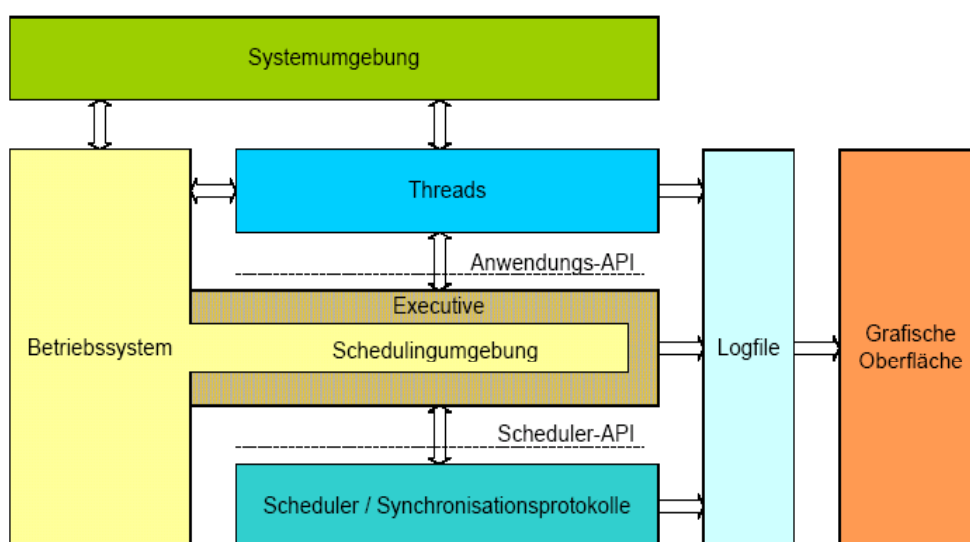


FIGURA F.9: Arquitetura do ferramental YASA

A FIGURA F.9 mostra como YASA amplia as propriedades de escalonamento do sistema operacional através de executivos que resulta na

- definição de interfaces com programas (APIs);
- capacidade de troca de componentes durante a execução;
- registro dos eventos do sistema;

<sup>8</sup> YASA: *Yet Another Scheduling Analyzer*, Mais um Analisador de Escalonamento.

- avaliação e otimização através de programas externos;
- capacidade de comparação de diversas configurações.

A ligação dos programas com o executivo dá-se basicamente por anotações de pré-processamento através de macros que são resolvidas em tempo de compilação.

A FIGURA F.10 mostra a estrutura de um executivo em que se ressaltam os blocos: a) adaptação ao sistema operacional; b) iniciação do projeto e do escalonador; e c) os componentes do YASA. À direita estão os componentes externos: conjunto de tarefas, o escalonador e o registro de eventos.

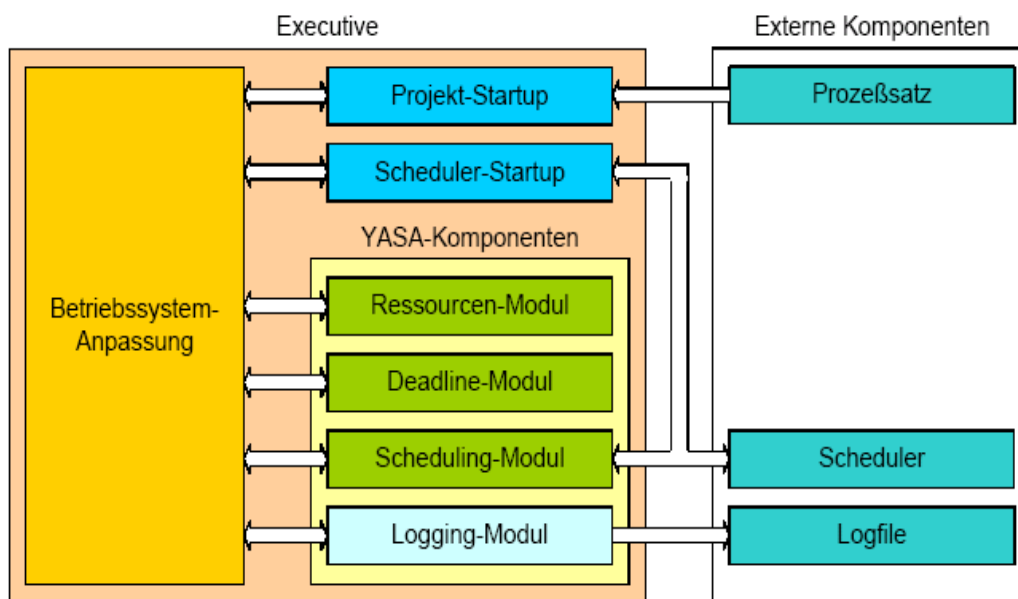


FIGURA F.10: Estrutura de um executivo

O YASA oferece, até então, 12 escalonadores do tipo estático, prioridade fixa e prioridade dinâmica, entre eles DMA, EDF, LLF, ELLF, FCFS, RMA, RR, SJF e Linux. Ainda, oferece os protocolos de sincronização definidos no padrão POSIX (1003.1c), entre eles FIFO, PIP, PCP, e DPCP, SRP não relacionados no Capítulo 3 e relacionados com EDF.

A FIGURA F.11 mostra a interface gráfica do usuário com YASA. Através desta é possível configurar e acompanhar os resultados de um projeto em tempo real. A interface é multilíngüe e independente de plataforma. Mostra dados em forma de diagramas e tabelas.

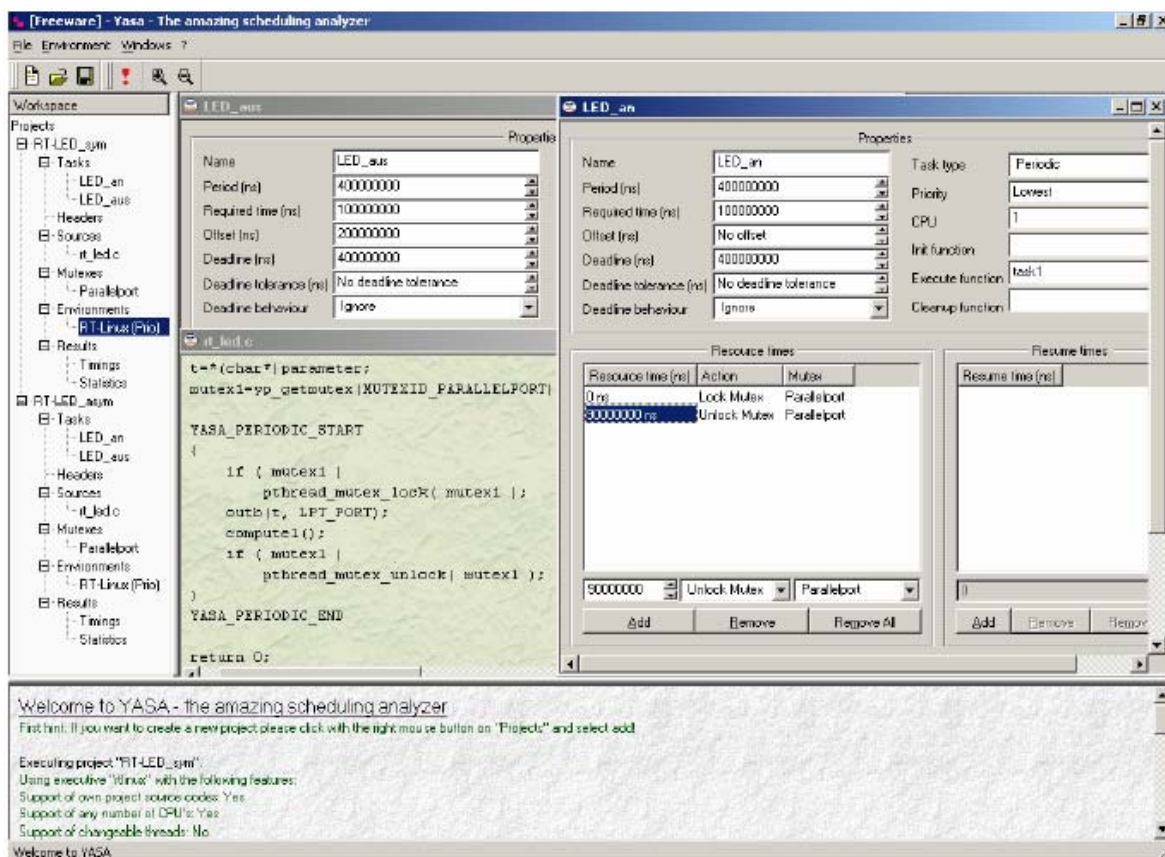


FIGURA F.11: Interface operacional com o usuário do YASA

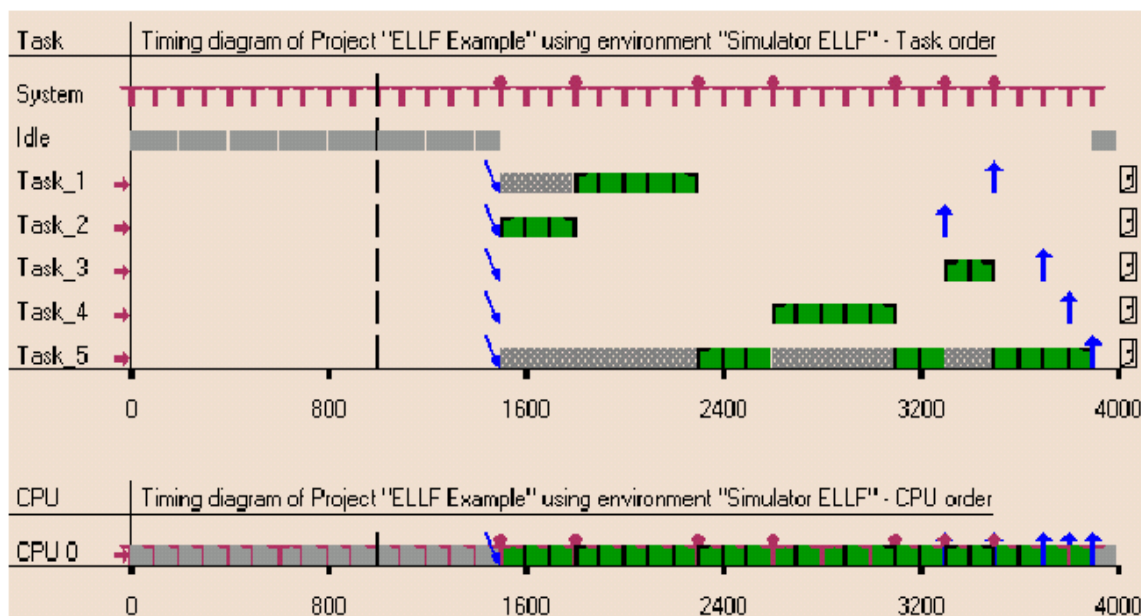


FIGURA F.12: Diagrama de tempos (ELLF)

A FIGURA F.12 mostra um gráfico de tempos com anotações de diversas características das tarefas, tais como uso de CPU, tempo de espera, período, prazo, deslocamento, prioridade e eventos de ativação.

Durante a execução do aplicativo, todos os eventos internos são registrados para posterior análise. YASA provê mais de 100 tipos de informações, tais como tempo médio de computação e sub-utilização dos processadores. Dados de diversas configurações podem ser relacionados.

#### F.4 AFTER

AFTER<sup>9</sup> (STEWART; ARORA, 2003) é uma ferramenta para analisar traços de programas em tempo real e sugerir melhorias. A entrada para a ferramenta são cenários de tempo extraídos de um sistema existente. Estes cenários são analisados para detecção de possíveis problemas. Desta forma pode-se efetuar e avaliar diversas modificações, tais como mudança do período de tarefas, mudança do tempo de execução de tarefas, alternância de prioridade fixa e dinâmica, e alternância de tratamento de eventos em tarefas aperiódicas através de interrupções ou servidores aperiódicos. Inicialmente, seu uso estava limitado a sistemas baseados no sistema operacional em tempo real VRTX. Seus autores apontam como trabalhos futuros, portá-lo para outras plataformas, tais como VxWorks e microcontroladores.

#### F.5 ASSERTS

ASSERTS<sup>10</sup> (GHOSE *et alii*, 1997) está orientado a sistemas distribuídos e heterogêneos. Permite modelar sistemas pela combinação de parâmetros declarativos e especificação de tarefas através de uma linguagem de pseudo-código (diferente de

---

<sup>9</sup> AFTER: *Assist in Fine Tuning for Embedded Real Time Systems*, Assistente para ajuste fino em Sistemas em Tempo Real.

<sup>10</sup> ASSERTS: *A Software Simulation Environment for Real-Time Systems*, Ambiente de Simulação de Aplicativos para Sistemas em Tempo Real. Lockheed Martin Federal Systems.

C++), que provê controle de fluxo e primitivas de comunicação. Permite múltiplos processadores, e cada processador pode ter escalonador e protocolos de acesso a recursos distintos.

O sistema é especificado através de arquivos de parâmetros. ASSERTS oferece diversos componentes, tais como Futurebus e Ethernet, bem como escalonadores abrangendo RMS, EDF e CE. Os algoritmos de escalonamento podem ser configurados. Novos algoritmos de escalonamento podem ser adicionados. Suporta alguns testes de escalonabilidade baseados em RMA.

## F.6 CAISARTS

CAISARTS<sup>11</sup> (HUMPHREY; STANKOVIC, 1996) é uma ferramenta de apoio ao projetista de sistemas em tempo real no que diz respeito ao escalonamento. Está fundamentado na observação de que as três fases de gerenciamento de tarefas e recursos em sistema em tempo real, alocação, escalonamento e liberação não podem ser tratadas isoladamente, uma vez que os mecanismos usados em uma fase afetam diretamente o desempenho das outras. CAISARTS é composto por duas partes: a) um sistema de modelagem de ambiente; e b) uma máquina de inferência. O modelo de um aplicativo é formado por um conjunto de objetos simples, tais como tarefas, dispositivos eletro-eletrônicos e recursos compartilhados. As relações entre os objetos são expressas através de regras tipo SE-ENTÃO que formam a base de conhecimento para escalonamento de sistema em tempo real. As regras são organizadas em repositórios de regras hierárquicos de modo a permitir o uso de partes diferentes de regras em diferentes pontos do projeto. Assim o usuário pode selecionar as orientações pertinentes.

---

<sup>11</sup> CAISARTS: *Conceptual, Analytical, and Implementation Scheduling Advice for Real-Time Systems*, Orientador de Escalonamento Conceitual, Analítico, e de Implementação para Sistemas em Tempo Real.

A base de conhecimento pode ser estendida com a adição de novas regras. Além disso admite regras em conflito. Neste caso, usuário é advertido, podendo tomar decisões bem embasadas.

## F.7 CarbonKernel

CarbonKernel<sup>12</sup> (CARBONKERNEL, 2001, 2006) é um simulador de sistema operacional baseado em técnicas de simulação por eventos. Possibilita avaliar o comportamento de diversos sistemas operacionais em tempo real. Para isso implementa um sistema operacional virtual com um conjunto completo de serviços genéricos que servem de suporte para criar RTOS simulados específicos. Cada RTOS simulado apresenta a mesma API de seu RTOS real correspondente. Suporta modelos de RTOS para: eCos, RTLinux, pSOS+ e VxWorks.

CarbonKernel simula o código fonte (escrito em C ou C++) diretamente sem considerar as instruções de máquina subjacentes. Desta forma não apresenta resultados precisos, e sim, uma primeira aproximação que pode ser refinada posteriormente, considerando-se processadores específicos. Os resultados da simulação podem ser apresentados através de gráficos de Gantt.

## F.8 DET/SAT/SIM

DET/SAT/SIM<sup>13</sup> (ANCILOTTI *et alii*, 1998) são partes de uma ferramenta adequada para aplicativos tempo-críticos com estimador de WCET e gerador de traço de execução. O sistema em tempo real é descrito em duas partes: o sistema e os nodos. O sistema é estruturado em nodos e elos de comunicação. As tarefas são distribuídas entre nodos.

---

<sup>12</sup> CarbonKernel foi inicialmente financiado por Axlog Ingénierie.

<sup>13</sup> DET/SAT/SIM: *Design Tool, Schedulability Analyzer and Scheduling Simulator*, Ferramenta de Projeto, Análise de Escalonabilidade e Simulador de Escalonamento.

Tarefas podem ser classificadas (pelo grau de pontualidade) em severas e fracas. Permite modelar tarefas periódicas e aperiódicas fracas. Os atributos das tarefas são: prazo, período, tempo de computação, importância e tolerância à perda de prazos. O conjunto de tarefas é especificado através de uma linguagem baseada em blocos cujo tempo de execução segue certas distribuições. Aparentemente não permite expressar controle de fluxo não-determinístico ou dependente de dados.

Suporta as políticas de escalonamento do processador RMS, DMS e EDF, e políticas de acesso a recursos PIP, PCP, DPCP e SRP. Os módulos de escalonamento são escritos em C, e novos algoritmos podem ser adicionados.

O simulador gera cargas de tarefas aperiódicas de acordo com as especificações de distribuição aleatória. Apresenta os resultados através de gráficos de Gantt.

## **F.9 DRTSS / PERTSSim**

DRTSS<sup>14</sup> (STORCH; LIU, 1996) capacita o projetista a modelar grande variedade de sistemas em tempo real a diferentes níveis de detalhes. Portanto, dá suporte a diferentes modelos de escalonamento, incluindo escalonamento de mono e multiprocessadores, tarefas periódicas e servidores para tarefas aperiódicas, algoritmos de escalonamento para computação imprecisa com compromissos de QoS e algoritmos definidos pelo usuário.

Os dispositivos mais importantes do DRTSS são: a unidade de análise de saídas, máquina de variação de parâmetros e o modelo de tarefas vivas. A unidade de análise de saídas pode ser programada através de uma linguagem proprietária para disparar e processar eventos de simulação (liberação de tarefas, preempção de tarefas, etc) e extrair informações de interesse ao modelo em observação. As saídas processadas podem ser orientadas pela variação de parâmetros.

---

<sup>14</sup> DRTSS: *Distributed Real-Time System Simulation*, Simulador de sistemas em tempo real distribuídos.

Nos modelos de tarefas vivas, tarefas e recursos são representados através de classes C++ contendo código real. Por um lado, isto permite implementar novos escalonadores e protocolos de acesso a recursos, e por outro lado permite refinar a funcionalidade de tarefas até a obtenção do código final.

DRTSS evoluiu para o produto PERTS<sup>15</sup>Sim e mais tarde para RapidRMA comercializado pela Tri-Pacific.

## F.10 Matlab TrueTime

TrueTime é um simulador para sistemas de controle em tempo real baseado no Matlab/Simulink. Sistemas de controle em tempo real envolvem tradicionalmente dois tipos de engenheiros (EKER; CERVIN, 1999): o engenheiro de controle que elabora o modelo da planta a ser controlada e o engenheiro de sistemas em tempo real que implementa o algoritmo de controle, considerando tarefas, prazos, prioridades, etc. Neste contexto o uso do Matlab/Simulink traz algumas vantagens, por exemplo, engenheiros de controle geralmente utilizam esta ferramenta para simular seus projetos. Por outro lado há pouco esforço em combinar as duas áreas em ferramentas para análise e simulação de sistemas em tempo real.

## F.11 Perf

Perf (BREGANT, 2002) (GÓES, 2001) é um ambiente de desenvolvimento integrado para sistemas em tempo real, composto por diversos módulos. Um destes módulos incorpora a capacidade de análise de escalonamento. Apresenta os seguintes resultados: seqüência de escalonamento, tempos de execução de cada serviço, tempos consumidos por interrupções e preempções, tempos de relaxamento, tempos

---

<sup>15</sup> PERTS: *Prototyping Environment for Real-Time Systems*, Ambiente para Prototipagem de Sistemas em Tempo Real.



consumidos pelo escalonador, tempos de espera devidos a bloqueios, taxa de utilização de CPU pelas tarefas e pelo escalonador, detecção e indicação de perdas de prazos.

A ferramenta obtém os dados pelo método de simulação. Portanto, não objetiva garantir a escalonabilidade do sistema e seu conjunto de tarefas. Outrossim, gera indicadores o mais próximo possível da execução real para determinado cenário.

Uma característica interessante é sua flexibilidade, permitindo experimentar diversas políticas de escalonamento. A simulação pode considerar um sistema operacional real incorporado ao sistema em análise ou um modelo genérico.

## **F.12 PerfoRMAx**

PerfoRMAx é uma ferramenta para análise do escalonamento em sistemas em tempo real utilizando RMA. A ferramenta: garante matematicamente o desempenho do sistema e sua escalonabilidade em situações de pior caso; aponta gargalos e recomenda estratégias de reprojeto; suporta as principais políticas de escalonamento; e independe da linguagem e da plataforma alvo. PerfoRMAx é fornecido comercialmente em conjunto com Aonix ObjectAda (AONIX, 2006).

A ferramenta mostra através de gráficos a carga do processador gerada por simulação estática do escalonamento das tarefas. Assim, as regiões críticas podem ser observadas.

## **F.13 RapidRMA, RapidRMA para UML**

RapidRMA (TRI-PACIFIC, 2006) é uma ferramenta comercial para análise de escalonamento baseada na ferramenta acadêmica PERTS. Recentemente incorpora interfaces para UML em tempo real. RapidRMA presta-se à análise de sistemas em tempo real severo, tais como controle de vôo e automotivo, dispositivos médicos, telemetria, e à análise de sistemas em tempo real rígido e suave, tais como redes e sistemas de telecomunicações. A ferramenta é baseada nas políticas de escalonamento RMS, DMS, CE e simulação aperiódica.

### F.14 Scheduler 1-2-3

Scheduler 1-2-3 (TOKUDA; KOTERA, 1998) é um analisador e simulador independente desenvolvido na CMU<sup>16</sup> como parte do projeto ARTS. É citado freqüentemente em artigos para sistemas em tempo real relevantes. Trata o escalonamento de tarefas periódicas e aperiódicas. As tarefas periódicas são tratadas por RMA e com o protocolo de acesso a recursos PIP. As tarefas aperiódicas são escalonadas através de um servidor de diferimento, com capacidade para simulação da liberação dos serviços a intervalos de tempo aleatórios segundo alguma distribuição em particular.

### F.15 ScheduLite

ScheduLite (LARSON, 1996) é uma ferramenta acadêmica para análise de sistemas em tempo real utilizando a política de escalonamento por prioridade fixa. Aplica-se a sistemas distribuídos ou monoprocessados. Grupos de tarefas podem ser atribuídos a nodos de processador. Os resultados são apresentados por tabelas e diagramas, que mostram tempo de resposta e outros parâmetros de análise.

### F.16 SEW

SEW<sup>17</sup> (CHATTERJEE *et alii*, 1997) é uma ferramenta de análise sem componente de simulação para avaliação de aplicativos de larga escala. Determina se o aplicativo atende aos requisitos de tempo. SEW modela o sistema a ser analisado em grandes blocos, tais como computadores, barramentos e discos. Estes blocos encapsulam os algoritmos dos respectivos recursos. A função de análise calcula limites de atraso, flutuação e utilização de recursos.

---

<sup>16</sup> CMU: *Carnegie Mellon University*.

<sup>17</sup> SEW: *Systems Engineering Workbench*, Bancada para Engenharia de Sistemas.

## F.17 STRESS

STRESS (AUDSLEY *et alii*, 1994) é um analisador e simulador de escalonamento baseado numa linguagem textual para descrição de recursos, estrutura de tarefas, algoritmos de escalonamento e protocolos de acesso a recursos. A análise de escalonamento considera as abordagens RMS, DMS com o protocolo de acesso a recursos PCP, e ainda o escalonamento EDF e LLF. A simulação permite adicionar outros algoritmos utilizando a linguagem de STRESS. Os resultados da simulação são apresentados em gráficos de Gantt.

A intenção básica de STRESS, segundo os autores, é experimentar novas abordagens de escalonamento ao invés de analisar e simular aplicativos reais. Seu desenvolvimento está descontinuado.

## F.18 TEV

TEV<sup>18</sup> (MORON; RIBEIRO; SILVA, 1998) é um ambiente de ensino integrado com diversas ferramentas para apoio ao processo de desenvolvimento de programas paralelos em tempo real. O ambiente inclui um gerador de programas paralelos, um analisador de WCET a partir do código fonte, um analisador de escalonamento e um depurador paralelo. Os programas gerados utilizam o núcleo operacional paralelo Virtuoso e são expressos em C. Os modelos elaborados com o TEV são baseados nos elementos oferecidos pelo Virtuoso: tarefas, semáforos, mensagens, relógios, recursos e memória.

## F.19 TimeWiz

TimeWiz (TIMESYS, 2006) é um sistema integrado para modelar, analisar e simular o desempenho e comportamento temporal de sistemas em tempo real. No

---

<sup>18</sup> TEV: *Teaching Environment for Virtuoso*, Ambiente de Ensino para Virtuoso.

modo analítico opera com diversos tipos de controle de bloqueios de recursos e com dois modelos de prioridade: estática e definida por RMA. Suporta múltiplos processadores. No modo simulação leva em conta o ambiente operacional do sistema e permite determinar os tempos de resposta mínimo, médios e máximos das tarefas.

## F.20 UTSA

O Simulador de Escalonamento de Processos da UTSA<sup>19</sup> (UTSA, 2006) permite simular um conjunto de processos com diversos algoritmos de escalonamento, e em seguida pode-se comparar os resultados estatísticos, tais como vazão e tempo de espera. Suporta os algoritmos de escalonamento RR, FCFS, SJF, PSJF e SJFA. Rajadas de E/S e de CPU são descritas por distribuições de probabilidade.

O simulador é uma aplicação Java e pode ser executado em qualquer plataforma JRE 1.1 ou mais recente. Produz arquivos diário no formato HTML, incluindo tabelas de dados e gráficos Gantt.

As simulações são submetidas através de três tipos de arquivos: a) arquivo de configuração, que descreve o simulador; b) arquivo de experimento, que consiste em várias execuções e modificações de parâmetros para cada execução; e c) arquivo de execução, que contém os parâmetros iniciais, tais como algoritmo de escalonamento, números de processos, distribuição de duração, de interchagada e de rajada de E/S.

---

<sup>19</sup> UTSA (*University of Texa at San Antonio*) *Process Scheduling Simulator*, Simulador do Escalonamento de Processos.