UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Fernanda Fernandes Peronaglio

Simulação de Tempo Real: Implementação do método para escalonamento manual na ferramenta RTsim

São José do Rio Preto
2016

Fernanda Fernandes Peronaglio

Simulação de Tempo Real: Implementação do método para escalonamento manual na ferramenta RTsim

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

Orientador:

Prof. Dr. Aleardo Manacero Jr.

São José do Rio Preto
2016

Fernanda Fernandes Peronaglio

Simulação de Tempo Real: Implementação do método para escalonamento manual na ferramenta RTsim

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

Prof. Dr. Aleardo Manacero Jr. Fernanda Fernandes Peronaglio

Banca Avaliadora:

Prof.ª Dr.ª Renata Spolon Lobato

Prof. Dr. Adriano Mauro Cansian

São José do Rio Preto
2016

Resumo

O desempenho e estabilidade de sistemas de tempo-real estão diretamente relacionados ao algoritmo de escalonamento usado para alocar tarefas às CPUs. Esta dependência levou à proposição de diferentes políticas de escalonamento, que atendam a diferentes conjuntos de restrições e propriedades. Essa diversidade de propostas faz com que seja necessário identifcar qual escalonador é mais adequado ao conjunto de tarefas de uma dada aplicação ou sistema de tempo-real. Para avaliar essa adequação uma das abordagens mais interessantes é o uso de simulação. Nesse sentido o RTsim (*Real-Time Simulator*) é uma ferramenta com boas características para avaliação de escalonadores. Entretanto, o RTsim apresenta apenas um conjunto fixo de escalonadores nativos, o que é característica de todos simuladores em que não é necessário codificar o escalonador. Para aumentar a flexibilidade do RTsim acrescentou-se, com esse projeto, uma nova funcionalidade para que o seu usuário possa criar e simular um novo algoritmo de escalonamento através de uma interface gráfica simples. A interface aqui projetada é baseada na transformação das regras de escalonamento em equações matemáticas, que são então transformadas em bibliotecas Java e inseridas no RTsim, como pode ser visto nos resultados apresentados.

Palavras-chave: Simulação, Escalonamento, Sistemas de Tempo-Real

Abstract

Real-Time Systems performance and stability are directly related to the scheduling algorithm used to allocate tasks to the CPUs. This dependance caused the proposion of differents scheduling politics, which attend differents restrictions and properties groups. This diversity of proposals makes necessary to identify which scheduler is more appropriate to the tasks set of an assigned application or real-time system. To evaluate this adequacy, the most interesting approaches is the use of simulation. In this regard, RTsim (*Real-Time Simulator*) is a tool whith good characteristics to evaluate schedulers. However, RTsim shows only a fixed group of native schedulers, which is a characteristic of all simulators where is not necessary codify the scheduler. To increase RTsim's flexibility it was added, with this project, a new funcionality that allows the user to create and simulate a new scheduler algorithm trought a simple graphical interface. The projected interface is based on the transforming of the scheduling rules in mathematical equations, that are transformed in Java libraries and inserted on RTsim, like it can be seen in the showed results.

Keywords: Simulation, Scheduling, Real-Time Systems

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

GSPD Grupo de Sistemas Paralelos e Distribuídos

RTsim Real-Time simulator

Capítulo 1

Introdução

A computação se torna cada vez mais presente na vida e cotidiano das pessoas. Dentro da enorme diversidade de sistemas e atividades que fazem o uso de computação, encontram-se sistemas que precisam executar atividades com restrição crítica de tempo. Sistemas de tempo-real [Liu, 2000] são aplicações que atendem rigorosamente os prazos de suas tarefas, sendo utilizados principalmente em atividades consideradas perigosas, como por exemplo o controle de reatores em usinas.

A eficiência destes sistemas depende de algoritmos de escalonamento criados especificamente para este tipo de aplicação, buscando encontrar a melhor situação e ordem para execução de suas tarefas. Estes algoritmos são objeto de estudo de muitos programadores deste tipo de sistema,e sua compreensão e avaliação de desempenho para o sistema que será desenvolvido é fundamental para estes profissionais.

Entretanto, a avaliação destes sistemas usando o ambiente real é inviável. Isso porque falhas implicam riscos financeiros, ou de segurança física. Assim, seu desenvolvimento precisa ser testado por meio de simulações.

1.1 Objetivos

O objetivo deste trabalho foi o desenvolvimento de um novo módulo na ferramenta de simulação RTsim (Real-Time Simulator) [Manacero Jr. et al., 2001], visando torná-la flexível quanto a inserção de novos escalonadores. O RTsim possui algoritmos de escalonamento implementados de forma nativa, porém qualquer novo tipo de algoritmo que se desejasse simular levaria a novas implementações.

A nova proposta de módulo foi criar um gerador de algoritmos de escalonamento, em que os usuários escolheriam os parâmetros e gerariam o algoritmo para a simulação. Ou seja, será possível gerar e testar um algoritmo que não esteja atualmente implementado na ferramenta, de forma intuitiva e simples, em tempo de execução.

1.2 Justificativa

Como já indicado, a avaliação de sistemas de tempo-real é menos arriscada se feita por simulação. Diversos simuladores foram propostos, porém a maior dificuldade neles é a avaliação de diferentes algoritmos de escalonamento, com alguns apresentando um conjunto limitado de escalonadores nativos e outros a sua programação explícita em alguma linguagem específica. A proposta de permitir que esses escalonadores sejam gerados através de uma interface gráfica é, portanto, muito interessante para a área.

1.3 Motivação

A motivação para a realização deste trabalho é a ausência de uma ferramenta que permita a geração simplificada de escalonadores nos simuladores de tempo-real. Em particular, dado que o RTsim apresenta característica interessantes como ferramenta de apoio ao ensino de tempo-real, a inserção desta funcionalidade o tornará ainda mais vesátil.

Visando as vantagens que seriam alcançadas com a melhora da ferramenta, tanto para o público acadêmico quanto para os profissionais em geral, é que este trabalho foi proposto.

1.4 Metodologia

A metodologia adotada para realização deste projeto envolveu um estudo aprofundado de sistemas de tempo-real, com foco principal em elementos de um sistema e nos algoritmos de escalonamento mais importantes dentro da área.

Além do estudo, para atender aos objetivos do trabalho, foram propostas as seguintes modificações e inclusões na ferramenta RTsim:

- Reestruturação da interface gráfica atual para receber o novo módulo.
- Adição de uma classe de interpretação para o novo módulo.
- Adição de métodos para gerar e compilar classes Java criadas em tempo de execução.
- Reestruturação das interfaces de saída para exibir os resultados gerados pelos novos algoritmos.
- Adição das classes necessárias para simulação do novo método.

O desenvolvimento do módulo envolve a implementação completa de uma interface gráfica intuitiva, e também a criação de classes de interpretação dos dados de entrada e métodos para simulação dos dados obtidos.

1.5 Exequibilidade

Para a realização deste trabalho foram necessários conhecimentos em programação orientada a objetos, concorrência, sistemas de tempo-real, escalonamento de tarefas e construção de compiladores, que são temas abordados durante o curso de graduação.

O desenvolvimento ocorreu dentro do Grupo de Sistemas Paralelos e Distribuídos (GSPD) [GSPD, 2016] que possui projetos abordando simulação e escalonamento, além de ser o grupo responsável por manter a ferramenta RTsim.

Sendo assim, o projeto possui toda a estrutura para ser realizado e atender o que foi proposto.

1.6 Organização do texto

Este texto está dividido em cinco capítulos. Além da introdução realizada neste primeiro, o segundo capítulo tráz a revisão bibliográfica para este projeto, trazendo a fundamentação teórica necessária e a análise sobre o estado da arte. O terceiro capítulo tratará das etapas do desenvolvimento e atividades realizadas, já o quarto apresentará os testes e os resultados obtidos. Por fim o quinto capítulo apresentará as conclusões e sugestões de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Neste capítulo se apresenta a revisão bibliográfica necessária para realização e entendimento deste trabalho. Isso inclui a definição de sistemas de tempo-real, na seção 2.1, a descrição de escalonadores para tempo-real, nas seções 2.2 e 2.3, e por fim a descrição do RTsim, que é a ferramenta alvo deste projeto, na seção 2.4.

2.1 Sistemas de Tempo-Real

Com o aumento do número de atividades que necessitam de computação, gerou-se a necessidade de utilizar diversos tipos de sistemas, cada um ajustado para a função que irá executar. Para atividades com restrição de execução no tempo percebeu-se a necessidade de uso de sistemas que tenham um controle rígido sobre o instante de execução de suas tarefas e sejam capazes de administrar o atendimento a todas elas.

Um sistema de tempo-real envolve aplicações que possuem restrições em seu tempo de execução, tendo prazos de atendimento para cada tarefa. Esses sistemas são normalmente usados para aplicações críticas (com tempo rígido para atendimento), que podem ter consequências graves em caso de falha [Shin and Ramanathan, 1994], como por exemplo sistemas para controle de pacientes ou para controle de aeronaves [Farines et al., 2000]. Estes sistemas podem ser classificados de acordo com sua necessidade de produzir resultados corretos atendendo seu prazo, sendo [Nissanke, 1997] :

- Soft Real-Time: São chamados de sistemas não críticos de tempo-real, pois impactos de ocorrências de falhas.
- Hard Real-Time: São chamados de sistemas críticos de tempo-real, em que não são admitidas falhas, sendo que qualquer funcionamento incorreto pode gerar consequências catastróficas.

2.2 Escalonamento de Tempo-Real

O termo escalonamento se refere a ordenar tarefas que devem ser executadas. Uma escala de execução é uma lista que indica a ordem em que deve ser ocupado o processador do sistema. O elemento responsável por realizar o escalonamento e coordenar essa ocupação é chamado de escalonador, [Farines et al., 2000].

Dada a importância do atendimento aos prazos neste tipo de sistema, é possível entender a necessidade de um bom funcionamento do escalonador para controlar a execução das tarefas. Para que esse controle seja possível existem as políticas de escalonamento. Estas políticas são conjuntos de regras que devem ser seguidas na escolha de qual tarefa ocupará o processador e sobre quais condições uma ação deve ser tomada, de modo que o escalonador produza resultados corretos e em tempo adequado para serem utilizados.

Quando se trabalha com escalonamentos de tempo-real é necessário que se entenda o funcionamento dos algoritmos de escalonamento, que podem ser classificados, de acordo com seu comportamento [Liu, 2000], em:

- Algoritmos Estáticos: São aqueles algoritmos que ordenam seu conjunto de tarefas por um determinado parâmetro ao iniciar a execução e não o alteram durante o processo.

- Algoritmos Dinâmicos: Estes ordenam seu conjunto de tarefas no início da execução, mas podem mudar essa ordem de acordo com as ocorrências do sistema.

Também é importante para este trabalho que se conheça conceitos relacionados ao escalonamento, como:

- Tempo de Folga: Este parâmetro está relacionado à ocupação do processador, sendo que esse tempo é o período em que o processador fica ocioso, ou seja não há tarefas aguardando para executar e também não há tarefas em execução.

- Preempção: Uma tarefa sofre preempção quando está realizando sua execução e é removida do processador em benefício de outra, que irá assumir o mesmo. A tarefa interrompida volta para a fila e poderá terminar sua execução posteriormente do ponto em que havia parado. O escalonador é o responsável pela preempção.

- Factibilidade: Se todas as tarefas de um escalonamento podem executar e também ter seus deadlines atendidos, o escalonamento é dito factível, ou seja existe escalonamento possível para o conjunto de tarefas.

- Fila de Pronto: É a estrutura em que são armazenados os dados para execução de tarefas no sistema, desde que essas tarefas estejam prontas para executar.

- Tempo de chaveamento: É o tempo consumido na troca de tarefas no processador.

2.2.1 Tarefas

Pode ser observado que quando se faz referência a processos/atividades que devem ser executadas, estas são chamadas de tarefas. Em sistemas de tempo-real, uma tarefa é uma ação que deve ser executada respeitando seus parâmetros e características de execução [Farines et al., 2000]. Durante este texto, será trabalhado o termo Tarefa, que algumas vezes pode estar associado com a sua frequência de execução, como segue:

- Tarefas Periódicas: Este é o tipo de tarefa que ocorre sempre respeitando o mesmo intervalo pré-determinado de tempo entre suas ocorrências.
- Tarefas Aperiódicas: Tarefas deste tipo não possuem seu tempo de ocorrência conhecido, o sistema não sabe quando ela pode chegar na fila de pronto.

A caracterização de tarefas é feita a partir de alguns parâmetros específicos, que são:

- Carga (Tempo de Execução): É o tempo necessário para que a tarefa complete a sua execução. Somente após a execução de toda a carga é que será considerada como completa.
- Tempo de Chegada (Instante de Ocorrência): Instante de tempo em que a tarefa se torna pronta para executar.
- Período: Intervalo de tempo em que a tarefa ocorrerá novamente, sendo relevante apenas para tarefas periódicas.
- Deadline: Um dos parâmetros mais importantes para tarefas de tempo-real. É o prazo em que a tarefa deve completar sua execução antes de ser considerada como falha. Existem dois tipos de deadline, o deadline relativo (intervalo de tempo entre a chegada da tarefa e seu prazo máximo para terminar) e o deadline absoluto (instante exato em que a tarefa deve terminar sua execução, é a soma do tempo de chegada com o deadline relativo).
- Independência: Uma tarefa é dita independente se pode executar sem que uma outra tarefa tenha sido executada previamente.

2.2.2 Algoritmos de Escalonamento

O estudo dos algoritmos de escalonamento e seus parâmetros é a base para a realização deste projeto. Para entender o funcionamento de um algoritmo e como são compostos o seu conjunto de regras e tarefas, serão apresentados como exemplo três algoritmos que são o Taxa Monotônica, o Least Slack Time First e o Míope.

É importante lembrar que para este trabalho foram estudados diferentes tipos de algoritmos, principalmente com características de escalonamento diversificadas, para que

fosse possível encontrar os parâmetros fundamentais de escalonamento. Como por exemplo:

- Taxa Monotônica [Liu and Layland, 1973]
- Least Slack Time First (LST) [Leung, 1989]
- Deadline Monotônico [Leung and Whitehead, 1982]
- Earliest Deadline First (EDF) [Liu and Layland, 1973]
- Servidor por Deferência [Strosnider et al., 1995]
- Míope [Ramamritham et al., 1990]

O Taxa Monotônica e o Least Slack Time First servirão apenas como exemplo de escalonamento monoprocessado para uma melhor compreensão do que ocorre em um escalonador. Isso também vale para o míope nos sistemas multiprocessados.

Taxa Monotônica [Liu and Layland, 1973]

É um algoritmo estático desenvolvido para tarefas com prioridade fixa, ou seja as tarefas recebem a atribuição de prioridade no início da execução e não alteram sua ordem durante o processo. Por ser um algoritmo simples e de fácil entendimento se tornou fundamental em todos os estudos de tempo-real.

O Taxa Monotônica ordena suas tarefas de acordo com seus períodos, sendo que tarefas com menor período terão maior prioridade. Para melhor funcionamento deste algortimo, suas tarefas devem ser periódicas e independentes e sua execução deve ser feita em ambientes monoprocessados. Também é importante considerar que o deadline de cada tarefa é equivalente ao seu próprio período e que as cargas devem ser conhecidas. É um algoritmo preemptivo, ou seja pode remover uma tarefa do processador se receber uma de maior prioridade na fila de pronto.

O algoritmo está implementado no RTsim e a Figura 2.1 mostra o resultado de um escalonamento.

No exemplo da Figura 2.1 as tarefas receberão suas prioridades de acordo com seus períodos, sendo T1 com prioridade = 1, T2 com prioridade = 2, T3 com prioridade = 3 e T4 com prioridade = 4, sendo que 1 é a maior prioridade.

O escalonador aloca a CPU à tarefa com maior prioridade, sendo que tarefas sofrem preempção sempre que uma tarefa mais prioritária tem uma nova ocorrência. Isso aconteceu em T=100(T1), T=180(T2), T=200(T1), T=250(T3), T=300(T1) e T=360(T2).

	Carga	Período	Chegada
T1	20	100	0
T2	40	180	0
T3	60	250	0
T4	80	450	0

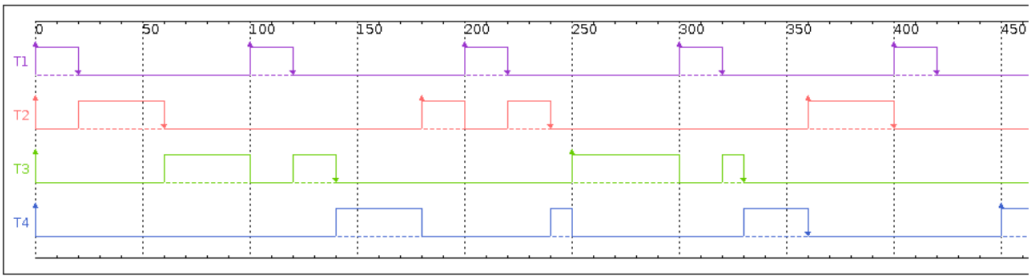

Figura 2.1: Escalonamento produzido pelo Taxa Monotônica

Least Slack Time First (LST) [Leung, 1989]

Este é um algoritmo com prioridade dinâmica, sendo que a cada chegada de uma tarefa na Fila de Pronto, o algoritmo refaz a ordem de execução de acordo com a situação. A prioridade é calculada de acordo com o tempo de folga para execução da tarefa, ou seja o período de tempo que a tarefa tem para executar a sua carga restante antes que ocorra seu deadline.

Também é um algoritmo preemptivo e idealizado para tarefas independentes, e que precisa ter o conhecimento das cargas que serão executadas. Utilizado em ambientes monoprocessados.

O LST não está implementado de modo nativo na atual versão do RTsim, embora seja um algoritmo importante entre os dinâmicos. Como os algoritmos dinâmicos muitas vezes apresentam um desempenho melhor no escalonamento, principalmente por terem a habilidade de mudar seu comportamento e ajustar melhor o escalonamento a situação atual, é importante entender como funcionam.

A Figura 4.10 mostra um escalonamento com o algoritmo utilizando o mesmo conjunto de tarefas que pode ser visto na Figura 2.1. Em um primeiro momento, as tarefas possuem a mesma prioridade que foi estabelecida no exemplo anterior (P1 = 1, P2 = 2, P3 = 3, P4 = 4), uma vez que seus deadlines são iguais aos períodos e nenhuma teve carga executada.

O escalonamento segue igual ao do Taxa Monotônica até T=250, quando P4 está executando e ocorre nova instância de P3. No LST a folga de P3, nesse instante, é de 190, enquanto a de P4 é de 170, tornando P4 mais prioritária que P3.

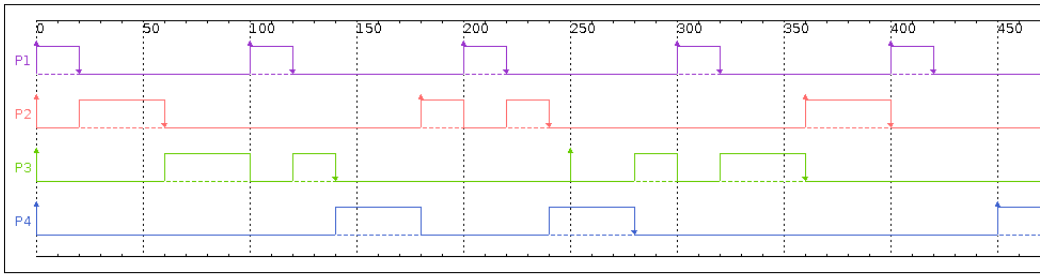

Figura 2.2: Escalonamento produzido pelo Least Slack Time First

Míope [Ramamritham et al., 1990]

Os algoritmos anteriores são para ambientes monoprocessados, ou seja ambientes que tem apenas uma unidade de processamento. O algoritmo míope foi desenvolvido a partir de algoritmos heurísticos , e é destinado a sistemas multiprocessados. Recebe este nome por "enxergar" apenas as tarefas que estão em sua janela de escalonamento. A janela de escalonamento, K, é o número de tarefas que o algoritmo irá considerar a cada passo. Caso o deadline das tarefas dentro da janela não possam ser atendidos, ocorre o backtracking, que é um refinamento do algoritmo de busca. Algumas características importantes são:

- As tarefas do conjunto são aperiódicas, independentes, tem carga e deadline conhecidos, além de possivelmente demandarem recursos específicos.
- A janela de escalonamento é quem define o número de tarefas que serão analisadas a cada passo
- O sistema pode ter N processadores, possivelmente disponibilizando recursos distintos.
- O escalonamento é local, ou seja não há migração de tarefas entre processadores
- O algoritmo não é preemptivo, ou seja, quando uma tarefa assume o processador não pode ser removida antes de executar toda a carga.

O funcionamento do algoritmo pode ser descrito pelos seguintes passos:

1. Ordenar a lista de tarefas prontas em ordem crescente de deadline relativo
2. Aplicar uma função heurística para as K primeiras tarefas da lista
3. Criar uma lista com as tarefas ordenadas pelo valor da função heurística
4. Se o escalonamento não pode atender todos os deadlines, aplicar a função de backtracking. O objetivo é encontrar uma solução possível ou então parar o algoritmo.

5. Se é possível atender aos deadlines, escalonar a primeira e repetir os passos de 1 a 3, desconsiderando a tarefa que já foi escalonada.

As figuras 2.3 e 2.4 apresentam o escalonamento parcial do conjunto de tarefas apresentado na Tabela 4.1. Esses escalonamentos foram produzidos pelo RTsim, com sorteio das ocorrências das tarefas. Deles é interessante observar na Figura 2.4 que a ocorrência da tarefa A5, no processador P2, não causa preempção da tarefa A6, como previsto pelo algoritmo.

Tabela 2.1: Entrada do algoritmo míope

Tarefas	Processador	Carga	Deadline	Recursos
A1	P1	10	20	A B D
A2	P1	50	100	E
A3	P1	60	150	T
A4	P1	10	120	C
A5	P2	30	100	E
A6	P2	25	60	W
A7	P2	5	25	T

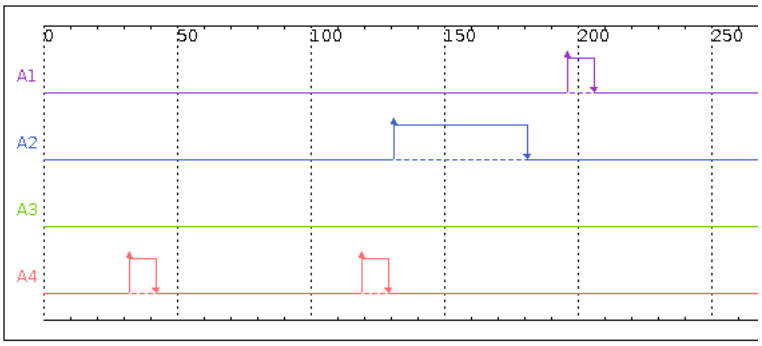

Figura 2.3: Gráfico para processador P1 com algoritmo míope

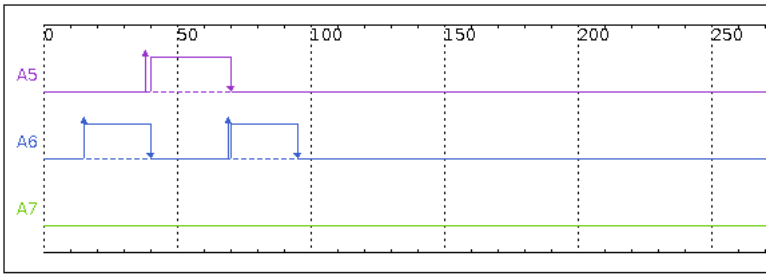

Figura 2.4: Gráfico para processador P2 com algoritmo míope

2.3 Simuladores de Sistemas de Tempo-Real

Simulação é uma área fundamental dentro da computação. Por meio dela é possível prever o funcionamento de sistemas e realizar testes sem ter que efetuar sua construção real e sem ter gastos desnecessários. A situação não é diferente para sistemas de tempo-real, em que simular é indispensável principalmente na avaliação de sistemas críticos.

Antes de se apresentar o RTsim, que é a ferramenta de estudo deste trabalho, nesta seção são apresentados alguns dos simuladores existentes e suas funcionalidades, para que seja possível a comparação entre as funcionalidades implementadas na ferramenta deste trabalho com as existentes.

2.3.1 STRESS [Audsley et al., 1994]

O STRESS foi desenvolvido na Universidade de York por um grupo de pesquisa especializado em sistemas de tempo-real. Este simulador é orientado para a simulaçao de sistemas de tempo crítico, possuindo uma linguagem específica desenvolvida para que fosse possível trabalhar com esta ferramenta. O software é utilizado no meio industrial e acadêmico.

Possui uma interface gráfica que permite a entrada de dados e características do sistema a ser simulado. Esta interface recebe primeiramente o nome do arquivo desejado e aguarda a escolha de uma das suas três principais funções:

- Análise: A escolha desta função resulta no surgimento de uma nova janela de opções, em que é possível escolher qual tipo de análise será feita pelo simulador.
- Simular: Esta escolha inicia a simulação e todos os seus resultados são salvos em um arquivo.
- Mostrar: Função que mostra o comportamento das tarefas e o resultado da simulação.

Cada uma dessas opções mostrada acima possui uma interface de saída diferente, que permite mais escolhas de comportamento ou exibição.

2.3.2 SPARTS [Nikolic et al., 2011]

O SPARTS foi desenvolvido em Portugal e é uma ferramenta flexível para simulação. Seu funcionamento é dividido em quatro módulos principais, como visto na Figura 2.5.

- Task Set Generator: A finalidade básica desse componente é permitir que o usuário trabalhe com diferentes conjuntos de tarefas. Portanto ele permite que o comportamento e as características do conjunto de tarefas sejam indicadas como parâmetros.

- Job Generator: Tem como objetivo especificar os parâmetros individuais de cada tarefa do conjunto que foi introduzido na etapa anterior.
- Job Sequencer: Sua função é receber as tarefas do Job Generator e organizá-las para execução de acordo com o tempo de cada uma. O resultado é passado para a próxima etapa.
- Execution Environment: Essa etapa executará a simulação. Por meio de dois módulos que interagem um com o outro, as tarefas são escalonadas de acordo com o algoritmo escolhido.

A estrutura modular do simulador permite seu uso para diferentes propósitos, sendo que seus módulos podem ser usados separadamente.

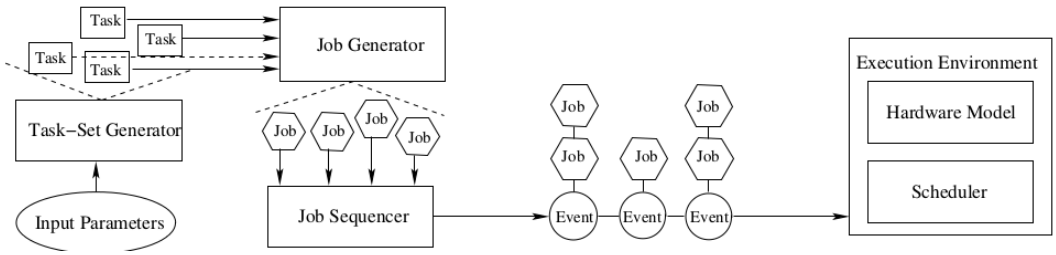

Figura 2.5: Funcionamento do escalonador SPARTS

2.3.3 Realtss [Diaz et al., 2007]

O Realtss foi desenvolvido no Instituto de Tecnologia do México, totalmente voltado para ser uma ferramenta aberta, que pode ser usada por qualquer pessoa que deseje. Sua estrutura modular permite a inclusão de mais algoritmos na ferramenta, trabalhando com linguagens como Tcl, C e C++.

O objetivo deste simulador é testar as políticas de escalonamento existentes, sendo uma ferramenta importante de ensino e pesquisa. Pode ser utilizada em sistemas operacionais Linux ou Windows.

A simulação com o Realtss é feita em três etapas:

- Definição dos Parâmetros: A interface desta etapa é bem clara e possui diversas opções, permitindo que o usuário indique características das tarefas (por exemplo, o período) e escolha o algoritmo que irá utilizar na simulação.
- Simulação: Após a conclusão da etapa anterior, o simulador exibe o comportamento das tarefas por meio de gráficos.
- Análise dos Resultados: A análise fornecida mostra as ocorrências mais importantes durante a simulação e também os dados de utilização, como utilização do processador.

A interface gráfica deste software é simples e muito explicativa, não sendo de difícil utilização e permitindo uma boa interação com o usuário.

2.3.4 AURTSS [Yaashuwanth and Ramesh, 2010]

Diferente dos simuladores vistos até aqui, o AURTSS (AU Real Time Scheduler Simulator) é uma ferramenta para plataforma *Web*. Desenvolvido na India, em um laboratório chamado LabView, iniciou com o objetivo de ensinar sistemas de tempo-real e também permitir a prática de profissionais.

A ferramenta possui uma interface bem intuitiva, construída de modo que qualquer usuário possa entender o que precisa ser inserido nos campos. O simulador recebe os dados das tarefas como entrada, e as escolhas de escalonamento, como alocação de recursos ou ativar/desativar preempção, e fornece como sua saída o gráfico de ocorrências no tempo.

O principal objetivo de desenvolver para plataforma *Web* foi a facilidade do acesso. Qualquer máquina conectada a internet pode executar a ferramenta, sendo interessante para ministrar aulas e para uso geral em diversos lugares. Também foi considerado pela equipe que este tipo de desenvolvimento facilita a questão de atualização, pois todos os usuários estariam atualizados ao mesmo tempo.

2.3.5 SimSo [Ch'eramy et al., 2013]

O SimSo (Simulation of Multiprocessor Scheduling with Overheads) é um simulador para ambiente multiprocessado desenvolvido em uma universidade francesa. Esta ferramenta é open source, sendo todo o seu código e documentação disponibilizado no website do projeto.

A linguagem Python foi adotada para o desenvolvimento por ser considerada pelos autores como fácil de ser aprendida e utilizada. Para que a simulação possa ocorrer, é necessário a programação manual, ou seja, é preciso que o usuário descreva o simulador por meio de uma classe programada em Python.

A ferramenta possui interface gráfica, sendo que esta foi construída com o objetivo de ser intuitiva para os usuários. A interface foi pensada de modo que o simulador fosse adequado para ser utilizado como ferramenta educacional.

2.4 RTsim [Manacero Jr. et al., 2001]

O RTsim (Real-Time simulator) é um simulador para escalonamento de tarefas de tempo-real. Seu desenvolvimento teve início em 1996, com o intuito de ser uma ferramenta de simulação que permitiria aos desenvolvedores de sistemas de tempo real testar seus

algoritmos e verificar se eram adequados para a aplicação. Após o desenvolvimento de seu protótipo foi observado uma nova aplicação para a ferramenta, o ensino de sistemas de tempo-real, o que resultou em um novo foco para desenvolvimento.

2.4.1 Breve Histórico

A versão inicial implementada em C, possuia cinco algoritmos implementados para ambientes monoprocessados, e após alguns anos recebeu mais cinco para multiprocessados [Kehdy, 1999]. Possuia também uma interface para interação usuário, implementada por meio da biblioteca chamada Xforms.

A biblioteca atendia a maioria das necessidades, mas tornava a portabilidade um grande problema para o uso da ferramenta. Isto devido a dependências de outras aplicações e de gerenciamento de janelas. O principal problema, é que ela dependia de gerenciadores de janelas que em sua maioria eram encontrados apenas em sistemas Linux.

A falta de portabilidade gerava um problema com o número de usuários, e consequentemente na difusão da ferramenta entre os simuladores existentes. Para resolver este problema e tornar o RTsim uma ferramenta de maior alcance, a biblioteca Xforms foi removida e foram integradas interfaces com o mesmo funcionamento da anterior, mas agora implementadas em linguagem Java. Para que a integração fosse possível foi aplicada uma técnica de comunicação entre linguagem C e Java [Miola, 2001], que permitiu invocar uma classe Java a partir de C. O grande problema era que Java dependia do carregamento da JVM (Máquina Virtual Java) e esta tinha que ser carregada a cada vez que se abria uma janela.

Portanto, foi proposto um novo projeto para reimplementar toda a ferramenta em Java. A reimplementação eliminou os problemas de carregamento e de portabilidade, tornando a ferramenta apta para ser executada em diversos sistemas operacionais [Goncalves, 2005]. Até hoje a ferramenta se mantém nesta linguagem.

2.4.2 Estado Atual

Atualmente o RTsim é mantido pelo GSPD (Grupo de Sistemas Paralelos e Distribuídos - UNESP) e está disponível para carregamento na página do grupo [GSPD, 2016]. É utilizado principalmente para os alunos de Sistemas de Tempo-Real do curso de graduação em Ciência da Computação.

A versão atualmente disponível possui cinco algoritmos monoprocessados.

- Taxa Monotônica
- Servidor por Deferência
- Servidor Esporádico

- Topo de Prioridade
- Troca de Prioridade

E possui três algoritmos para sistemas multiprocessados.

- Míope
- Leilão
- Leilão Focado

O RTsim pode ser executado de três formas distintas, sendo que o usuário escolhe uma delas após a entrada de dados sobre as tarefas a serem escalonadas. As formas de execução são:

- Plotar Gráfico: A função gera o gráfico detalhado do escalonamento, para que o usuário possa acompanhar as ocorrências. A interface é completa tanto no ambiente monoprocessado, como no multiprocessado, separando os gráficos por processador e fornecendo diversas opções para o usuário.
- Análise: Interface de saída que apresenta apenas estatísticas sobre as ocorrências, como por exemplo tarefas que perderam deadline.
- Modo Aprendizado e Modo Teste: São as funções mais importantes para o uso em ensino. Com elas é possível que o usuário entre manualmente com o escalonamento e receba a correção do mesmo. É possível exibir os erros que o usuário cometeu ao tentar escalonar e também fornecer ajuda durante a entrada do escalonamento no modo aprendizado.

O uso durante as aulas ministradas com o apoio do simulador tem se mostrado de grande auxílio para os estudantes. O conteúdo visto em aula pode ser praticado e testado na ferramenta, o que torna a compreensão do comportamento dos escalonadores mais fácil, principalmente por possibilitar a visualização gráfica das tarefas sendo executadas.

A Figura 2.6 exemplifica a entrada de dados do Algoritmo Taxa Monotônica (a esquerda) e a saída gráfica resultante da opção “Plotar Gráfico” (a direita). A esquerda da Figura 2.6 está a entrada de dados, onde é possível observar os parâmetros de entrada que são solicitados, as opções que podem ser escolhidas e também é possível ver um registro de todas as tarefas que já foram fornecidas. A direita está o gráfico do escalonamento que exibe as execuções de cada tarefa e possui opções de visualização e estatísticas para auxiliar o usuário.

A entrada de dados no RTsim pode ser feita pela inserção de cada tarefa manualmente, usando janelas diferentes, ajustadas para os diversos tipos de tarefa, ou pela leitura de

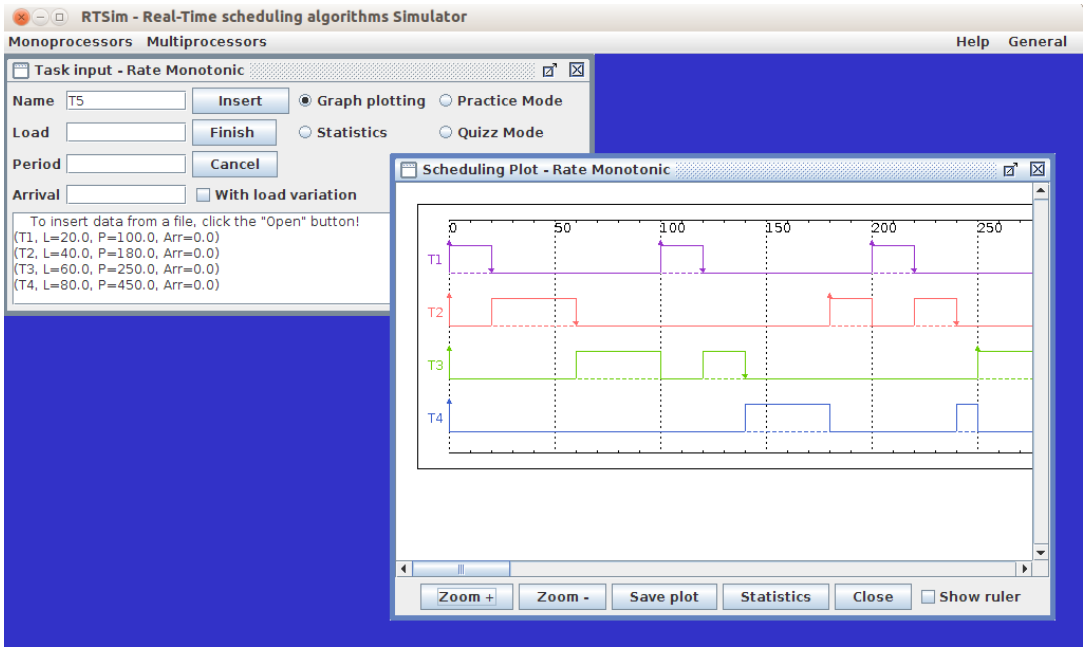

Figura 2.6: Interfaces do RTsim para o algoritmo Taxa Monotônica

arquivos contendo as informações necessárias. Já a saída de resultados é feita tanto de forma gráfica como em arquivos. Dois arquivos são produzidos automaticamente pelo simulador, sendo um com o rastro da simulação e outro com dados das tarefas de entrada. Este último arquivo pode ser usado futuramente como dado de entrada de uma simulação.

2.5 Considerações Finais

Neste capítulo foram apresentados os conceitos gerais sobre sistemas de tempo-real, seus escalonadores e algumas das ferramentas para sua simulação. Esse conhecimento é transportado para o desenvolvimento de um módulo para geração de escalonadores no RTsim, como descrito no próximo capítulo.

Capítulo 3

Desenvolvimento

Esse capítulo aborda as atividades desenvolvidas neste trabalho. Na seção 3.1 se apresentam os principais componentes da implementação do RTsim. O trabalho desenvolvido é apresentado nas seções seguintes, sendo que na seção 3.2 estão descritas as modificações necessárias nas interfaces do RTsim para que a inserção da geração de escalonadores fosse eficiente. Já na seção 3.3 são apresentados os detalhes da implementação deste gerador. Por fim, em 3.4 é apresentado um breve resumo das atividades ainda em andamento no projeto.

3.1 Estudo da ferramenta RTsim

Para implementar as alterações necessárias no RTsim é preciso compreender como a hierarquia de classes atual é organizada. Sua estrutura básica cria classes separadas para cada algoritmo de escalonamento implementado de forma nativa. Além disso ocorre uma separação entre classes de interface de entrada, interface de saída e de simulação do escalonamento.

A classe principal da ferramenta é a *GUI_DesktopView*, que é responsável por gerar a interface inicial e realizar a chamada de novas classes de acordo com escolhas do usuário. Existem classes específicas para a exibição de gráficos monoprocessados e multiprocessados e geração de estatísticas, como a *GUI_GraficoMono*, que contém os métodos necessários para efetuar o desenho do gráfico na tela.

Também são implementados separadamente os métodos para "Modo Teste" e "Modo Aprendizado", sendo que cada um deles tem as suas respectivas classes de interface e tratamento. Independente da escolha do tipo de saída que o usuário selecionar, a classe responsável pelo algoritmo é a mesma e é sempre executada, gerando os dados que serão usados pelas outras.

Para cada algoritmo existem duas classes de execução principais. A Figura 3.1 ilustra de maneira simplificada o relacionamento entre as classes por meio de um diagrama.

Neste é possível observar as relações entre interfaces e implementação dos algoritmos, e também a herança entre classes. Por exemplo, a execução do Algoritmo Taxa Monotônica segue, levando em consideração que o usuário não irá carregar um conjunto de tarefas a partir de um arquivo, mas sim inserir os dados diretamente na interface (Figura 3.2), a seguinte ordem de chamada de classes.

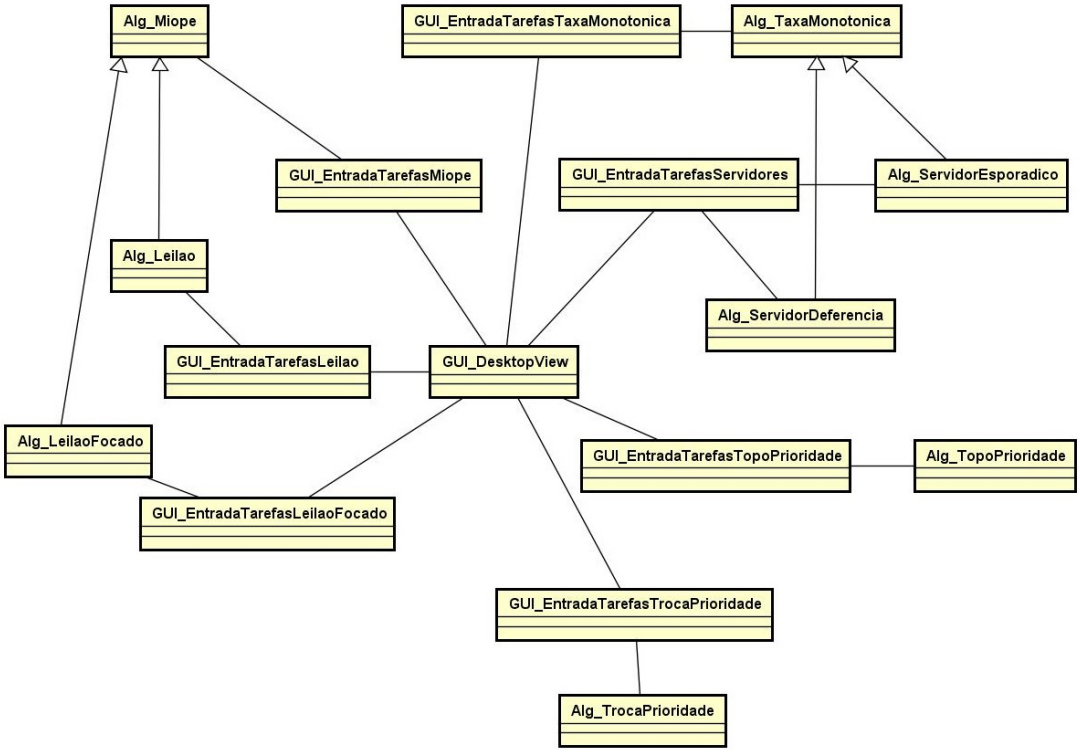

Figura 3.1: Diagrama de classe exibindo as classes de interface e simulação dos algoritmos

1. A execução da tela inicial é feita pela classe *GUI_DesktopView*, que realiza uma ação a partir da escolha do usuário por meio do menu de opções.

2. O usuário seleciona no menu de monoprocessados o algoritmo Taxa Monotônica, e o programa faz a chamada da classe *GUI_EntradaTarefasTaxaMonotonica*.

3. Esta classe é responsável por exibir a janela de entrada do conjunto de tarefas. Quando o usuário entra com os dados e solicita o início do escalonamento, é feita a chamada da classe principal, a *Alg_TaxaMonotonica*, que realiza o escalonamento e gera os dados necessários para as próximas classes:

   i) Se o usuário escolheu “Plotar Gráfico”, será chamada a classe responsável por desenhar o diagrama do escalonamento (*GUI_GraficoMono*).

   ii) Se o usuário escolher “Modo Teste” ou “Modo Aprendizado”, serão chamadas a *GUI_ModoTeste* ou *GUI_ModoAprendizado* respectivamente.

iii) Se a opção for por apenas apresentar estatísticas do escalonamento, será chamada a classe *GUI_EstatisticaMono*.

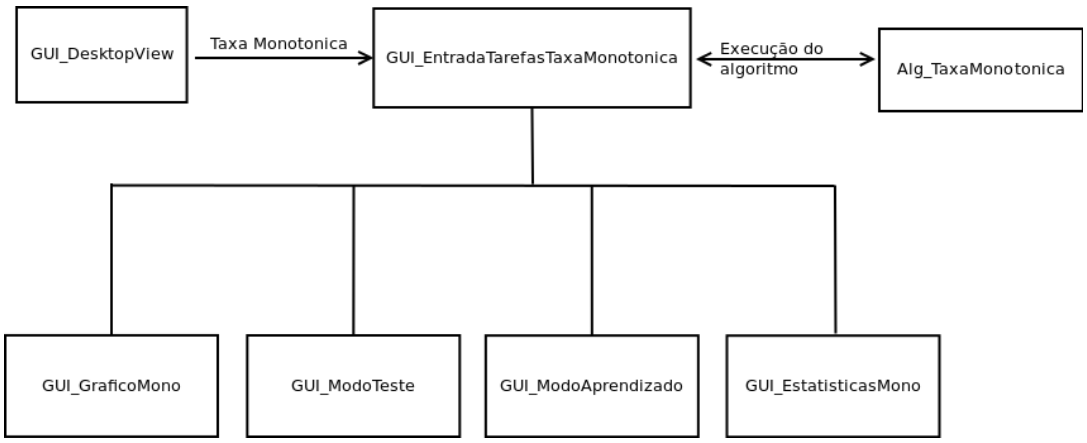

Figura 3.2: Chamadas de classes de execução para gerar escalonamento

As classes que implementam os algoritmos possuem métodos semelhantes, porém implementados de forma diferente por pertencerem a diferentes algoritmos. Os métodos são executados de modo que simulem o escalonamento e produzam resultados para as outras classes.

O método *escalonar()* é o principal para o escalonamento, sendo implementado de maneira recursiva para facilitar a codificação. Este método simula todo o comportamento do escalonador, verificando ordens de execução e realizando preempção quando necessário. Possui a capacidade de verificar se poderá executar a tarefa e calcular os tempos de folga, entre outras funcionalidades.

Outros métodos fazem tratamento dos dados que passarão pelo escalonador, como por exemplo o método que ordena as tarefas por prioridade, ou o que determina os tempos de folga do processador.

Classes de interface são mais complexas pela quantidade de itens que elas manipulam e pela quantidade de ações que eles possuem para cada evento. Cada simples item que será mostrado na tela, como por exemplo uma caixa de texto, é tratado e criado separadamente, sendo unido aos outros e organizado na tela por um outro método da classe.

Do ponto de vista funcional, as interfaces de entrada são responsáveis pela manipulação de arquivos de entrada e de saída (com dados das tarefas), além de toda interação com o usuário. Toda a validação dos dados de entrada também é feita por essas interfaces, permitindo um tratamento de erros dessa entrada.

3.2 Modificação das Interfaces do RTsim

As interfaces de entrada originais do RTsim, como a exemplificada na Figura 3.3 , copiada aqui da Figura 2.6, não eram adequadas para o módulo aqui desenvolvido. Em parte

existia uma confusão entre os componentes da interface e sua funcionalidade, e em parte faltava flexibilidade nos elementos que as compunham. Assim, aquelas interfaces foram modificadas para um novo padrão.

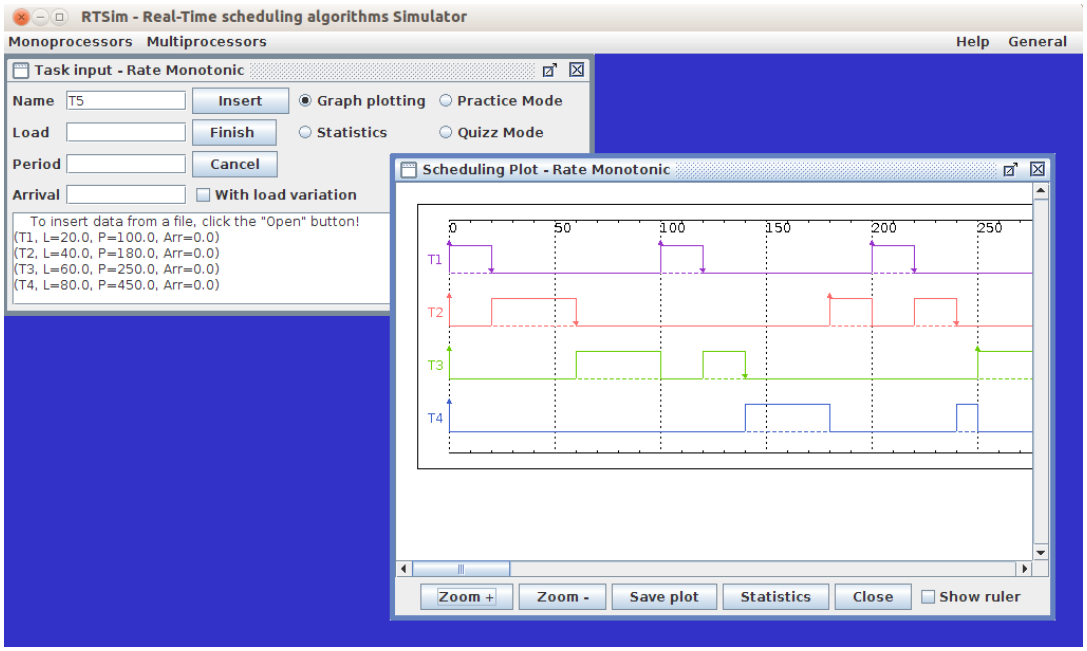

Figura 3.3: Interfaces do RTsim para o algoritmo Taxa Monotônica

A nova interface trabalha com um tamanho maior de janela para facilitar a visualização e organização de seus componentes. Composta por três painéis de dados, cada painel contém os itens que mais se adequam a sua classificação funcional (Figura 3.4), sendo:

1. Propriedades das Tarefas: É o painel principal, responsável por conter todos os parâmetros de tarefas que serão inseridos pelo usuário do sistema. Na Figura 3.4 pode ser visto no canto superior esquerdo, neste caso aguardando o nome da tarefa, carga, período e instante de chegada.

2. Tarefas: Esse painel contem a área de texto que exibe as tarefas já inseridas pelo usuário, sendo um resumo simples, mas que permite o acompanhamento da inserção. Contem também o botão que permite carregar um arquivo contendo um conjunto de tarefas usado anteriormente.

3. Propriedades da Simulação: Neste painel o usuário pode escolher o que deseja executar com o conjunto de tarefas inserido, como plotar gráfico ou iniciar o Modo Teste. Também está disponível a opção de inserir variação de carga [1] durante a simulação.

[1] Variação de carga é um mecanismo em que o simulador altera a carga original da tarefa pela inserção de desvio gerado aleatoriamente

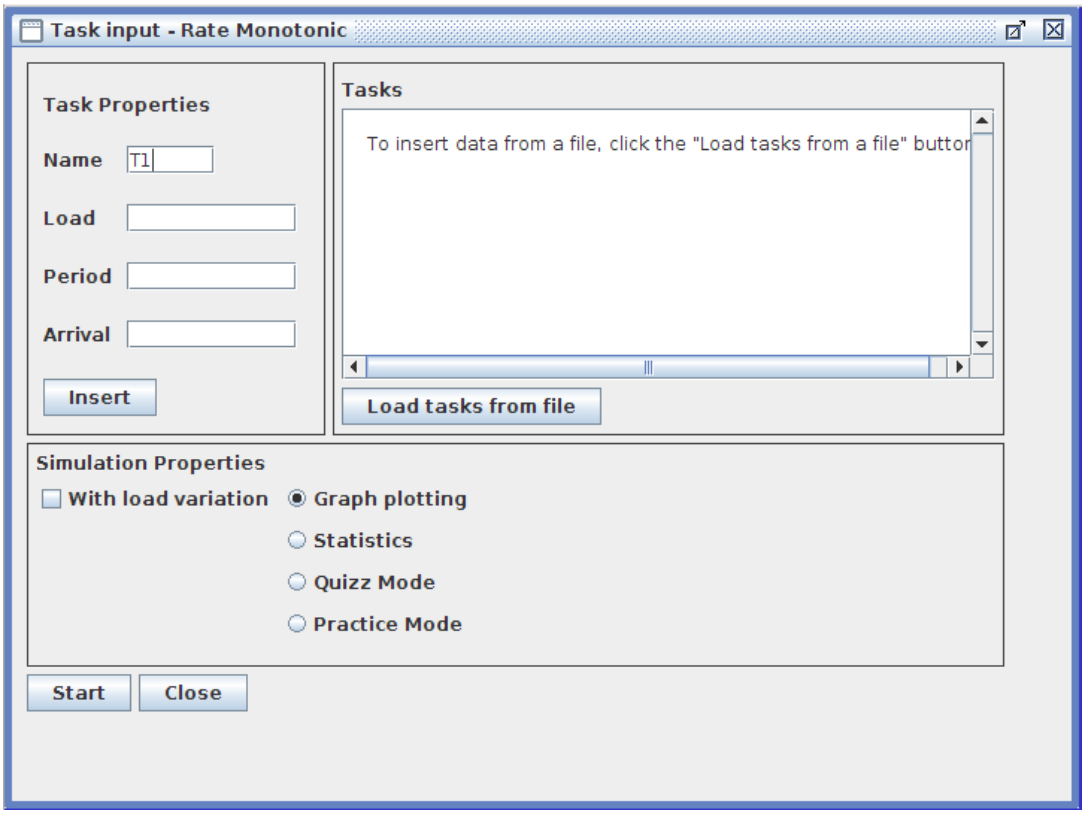

Figura 3.4: Novo layout das interfaces de entrada dos algoritmos

A interface mostrada na Figura 3.4 foi inserida em todos os algoritmos multiprocessados e em três dos monoprocessados. As exceções foram os algoritmos de Topo e Troca de Prioridade, que já possuiam uma interface mais atual e com um padrão semelhante ao novo modelo. Além disso, por envolver semáforos como parâmetros, estes não se adequariam à nova interface, o que poderia resultar em uma disposição confusa de seus elementos.

Na Tabela 4.3 são apresentados os parâmetros que caracterizam tarefas em ambientes monoprocessados. Como se pode observar, existe uma diferença de denominação de dois campos (período e instante de chegada contra *deadline* e intervalo mínimo entre chegadas) e um campo adicional no caso de tarefas aperiódicas, que é o número de ocorrências. Destas diferenças tem-se que período equivale a *deadline* para as periódicas, não implicando diferença portanto.

Tabela 3.1: Diferenças entre parâmetros

Tarefas Periódicas	Tarefas Aperiódicas
Nome Tarefa	Nome Tarefa
Carga	Carga
Período	*Deadline*
Instante de Chegada	Intervalo Mínimo entre Chegadas
	Número de Ocorrências

Para ambientes multiprocessados a janela de configuração foi mantida, apenas com a introdução da possibilidade de configuração dos nós que executarão a tarefa.

As interfaces de saída, como a geração de gráfico não foram alteradas. Estas já estavam com uma boa disposição dos elementos, sendo bem agradáveis visualmente. Portanto, o novo módulo foi implementado para gerar as mesmas interfaces de saída e não afetar a padronização das interfaces.

3.3 Geração de escalonadores de ambientes monoprocessados

Para a realização deste trabalho foi necessário um estudo sobre os algoritmos de escalonamento existentes e mais utilizados. O estudo levou em consideração principalmente os parâmetros das tarefas, pois o objetivo era entender a relação entre eles e quais seriam os mais interessantes para compor a interface gráfica no novo módulo.

As diferenças de comportamento entre os escalonadores foram analisadas e estudadas, visando como objetivo final encontrar um padrão, em suas características, que pudesse ser aproveitado na nova interface. Este padrão deve ser capaz de abranger a maior quantidade de requisitos dos algoritmos, de modo que o usuário fique livre para gerar algoritmos variados na mesma interface.

A inserção genérica de escalonamento proposta necessita da inserção de características das tarefas e também do escalonamento. Além disso, o módulo deve ser capaz de trabalhar com atributos de tarefas periódicas ou aperiódicas, tratando os parâmetros de forma diferente para cada tipo.

Os parâmetros de entrada escolhidos são os mesmos já utilizados nas interfaces mostradas anteriormente, porém agora estão todos juntos no mesmo painel. Sendo eles:

- Carga
- Período
- *Deadline*
- Chegada
- Intervalo mínimo entre chegadas
- Ocorrências

Estes parâmetros são utilizados por uma grande gama de algoritmos, por isso foram selecionados para formar a entrada do módulo. É possível observar que a inserção dos

valores de entrada de acordo com o escalonamento que é pensado, está agora sob responsabilidade do usuário. É necessário que se conheça o funcionamento do algoritmo antes de especificá-lo no simulador.

Outro conjunto de parâmetros identificado se refere ao comportamento do escalonador em relação a ordenação de tarefas, ou seja, como um algoritmo trata a priorização das tarefas. Nesse caso se identificou as seguintes possibilidades:

- Prioridade Fixa
- Prioridade Dinâmica: Sendo possível escolher em que momento o escalonador irá fazer o recálculo, que pode ser na ocorrência de:
  - Chegada de Tarefa
  - Conclusão de Tarefa

Os parâmetros mostrados anteriormente tratam de características de execução da tarefa individualmente ou da ordenação do conjunto de tarefas. Para que não se diferenciasse tanto das outras interfaces e se evitasse a colocação de muitos elementos em uma única janela, a entrada da especificação do escalonador a ser gerado foi dividida em cinco janelas, com exibição ordenada sequencialmente. A ordem destas janelas estabelece uma sequência natural para a especificação e geração do escalonador, sendo elas:

1. Definição do nome para o algoritmo: Interface inicial que recebe o nome que terá o escalonador gerado.
2. Definição de características básicas: Nesta etapa é determinado o tipo de prioridade entre tarefas que será utilizada pelo simulador.
3. Tipo de Geração: Aqui o usuário tem a opção por selecionar algum algoritmo previamente especificado ou fazer a especificação dele a partir de uma expressão algébrica.
4. Escalonador de Tarefas: Etapa principal deste novo módulo, em que é feita a inserção da equação que define a política de escalonamento, usando os parâmetros descritos a seguir e relacionando-os por meio de operadores aritméticos(soma, subtração, entre outros).
5. Final: Esta etapa fornece um resumo das características de escalonamento que foram inseridas, para que seja possível conferir se correspondem ao desejado e, em caso positivo, iniciar a geração do escalonador.

O levantamento dos algoritmos mais importantes da área mostrou quais são os parâmetros mais utilizados em escalonamentos, tanto de forma direta como associados em expressões algébricas. Os parâmetros atualmente considerados são:

- Carga
- Período
- *Deadline*
- *Deadline* Relativo
- Carga Restante
- Carga Executada
- Tempo Atual

Estes parâmetros carregam informações fundamentais sobre o escalonamento de tarefas e funcionamento do algoritmo. Além disso, o conjunto escolhido abrange tarefas periódicas e aperiódicas, o que aumenta as possibilidades de geração. Pode se tomar como exemplo o algoritmo Least Slack Time First apresentado na seção 2.2. Seu escalonador relaciona três parâmetros (*Deadline*, Tempo Atual do Escalonamento e Carga Restante) por meio de uma equação simples e efetua a ordenação de execução.

Na Figura 3.5 se exibe a interface desenvolvida para a especificação dos parâmetros para o escalonamento. É possível observar na interface a existência de um conjunto de botões (mais a esquerda) para os parâmetros a serem usados na definição da política de escalonamento. Também se vê ao centro botões para os operadores aritméticos que podem ser usados. Por fim, do lado direito aparecem as opções de ordenação e na parte superior a expressão algébrica construída.

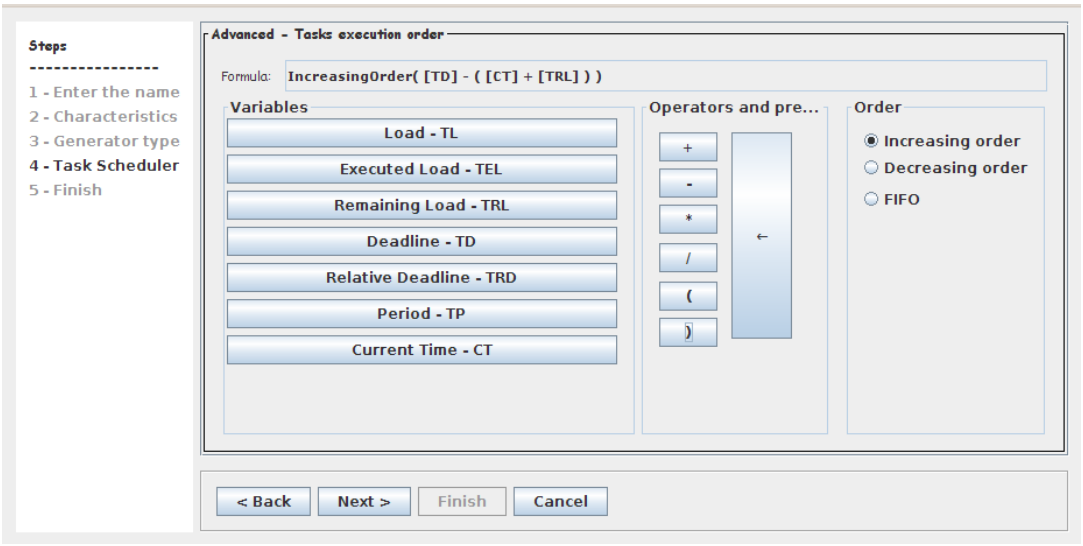

Figura 3.5: Painel para inserção de fórmula no escalonador

3.3.1 Implementação

Para implementar o módulo de geração de escalonadores foi preciso resolver vários problemas relacionados à interação entre o usuário e o módulo e entre o módulo e o RTsim. A solução adotada para cada problema é apresentada a seguir.

Interpretação da fórmula

Para que fosse possível receber a fórmula do usuário por meio da interface mostrada na Figura 3.5, foi necessário implementar um interpretador específico. A expressão que aparece no campo "Fórmula" na Figura 3.5 é armazenada e usada como entrada desse interpretador.

O processo de interpretação segue uma estrutura pré-definida, na qual cada objeto da interface é mapeado para objetos e métodos específicos da biblioteca de interpretação. Essa biblioteca está implementada pela classe *interpretador.jj*, sendo que para cada parâmetro inserido na expressão se associa um objeto da classe e os métodos que o manipulam. Por exemplo, ao adicionar o *token* correspondente ao parâmetro *deadline*, "TD", o mesmo é associado ao objeto *tDeadline*. Na construção da classe que simulará o algoritmo, todas as referências a "TD" serão associadas ao método *getDeadline()*, que retorna o seu valor.

Geração da nova classe

Como o objetivo é gerar um novo algoritmo de escalonamento na ferramenta e isso envolve a codificação de uma classe, o sistema gera automaticamente uma classe com o nome do escalonador e seus dados necessários.

Como em Java não é possível a inserção ou a atualização de classes em seu pacote em tempo de execução, a solução é fazer uso do carregamento de classes externas. Portanto, a nova classe é gerada em um diretório externo chamado de "rtsim/externo". Deste modo é feita a compilação e execução da classe sem a necessidade de reiniciar o software.

O Algoritmo 1, mostrado a seguir, é responsável por gerar e executar a nova classe. É possível observar toda a ordem de execução, iniciando com a interpretação (linha 1) e escrita do código (linha 3), a sua compilação (linha 4) e finalizando com o carregamento (linha 11) e a execução da classe gerada (linha 12).

```
1  Realiza interpretação dos dados de entrada;
2  if Não houver erros na interpretação da fórmula then
3      Escreve o código em um arquivo com o nome escolhido pelo usuário;
4      Compila o código gerado como uma classe Java;
5      if Não realizou a compilação then
6          Informar Erro;
7      end
8      else
9          if Compilação correta then
10             Salva os dados de entrada em um arquivo de texto;
11             Carrega a nova classe para o conjunto em execução;
12             Executa a classe gerada;
13         end
14     end
15 end
```

Algoritmo 1: Algoritmo para geração da nova classe

O Algoritmo 2 a seguir ilustra um método da classe gerada pela ferramenta. Nele é possível observar como é feita a ordenação da lista de tarefas por prioridade. As declarações são feitas de acordo com as variáveis presentes na fórmula inserida (linhas 2 e 3). Neste trecho ocorre uma primeira ordenação das tarefas de acordo com a fórmula, para que seja posssível definir as prioridades (linha 5).

```
1  if Há tarefas para escalonar then
2      Declara as variáveis que estavam na fórmula;
3      Estabelece a relação que o usuário inseriu;
4      while Lista não estiver completamente ordenada do
5          Ordena lista de tarefas por prioridade estabelecida na entrada
6      end
7  end
```

Algoritmo 2: Algoritmo para ordenar as tarefas

Desenvolvimento dos métodos para simulação

Para que a classe gerada não ficasse muito extensa devido aos métodos de simulação do RTsim, como por exemplo o *escalonar()*, que é uma funcionalidade comum a todos os tipos de escalonamento, foi criada uma classe de apoio. Esta classe é nativa da ferramenta e é chamada pelas classes geradas. Nela são implementados métodos comuns da ferramenta, como a preparação de tarefas para o gráfico.

Esta nova classe recebe a lista de tarefas e os comandos de ordenação da classe gerada. Após receber os parâmetros fica responsável por executar o escalonamento e gerar as saídas necessárias.

Em caso de prioridade dinâmica, quando é necessário o recálculo durante a execução, esta classe faz a chamada do método *setPrioridade()* gerado nas classes automatizadas. Este método faz a reordenação de acordo com os parâmetros de entrada e retorna a lista de tarefas para continuar a simulação.

Tratamento de Tarefas Periódicas e Aperiódicas

Cada tarefa é tratada como um objeto composto por suas características. O tratamento de tarefas periódicas foi o primeiro a ser implementado devido a sua simplicidade de tratamento.

Tarefas Aperiódicas tem comportamento mais complexo, envolvendo diferentes métodos de implementação. Estão disponíveis os tratamentos por Interrupção, Background e servidor de Polling, sendo:

- Interrupção: Tarefas aperiódicas são tratadas como tarefas de maior prioridade no escalonamento. Todas as periódicas param a execução quando uma aperiódica ocorre e esta executa até completar.
- Background: As tarefas recebem a prioridade mais baixa, sendo executadas apenas nos tempos de folga do processador, quando não existem periódicas na fila.
- Polling: Neste método é criada uma tarefa periódica com carga e período escolhidos pelo usuário. As tarefas aperiódicas só poderão executar quando a tarefa servidora tiver carga disponível e estiver em seu período de execução.

A seleção do método é feita por uma interface, em que após ser identificado que foi inserida alguma tarefa aperiódica, será exibida uma caixa com botões para escolha do tratamento. Essa escolha é passada para a classe de tratamento e a ferramenta segue sua execução normalmente.

Reutilização dos algoritmos criados

Uma parte importante na avaliação de um novo algoritmo é a possibilidade de simular novamente o mesmo ou poder modificar algum de seus parâmetros de entrada. Para tanto se incluiu no RTsim funcionalidades para utilizar o escalonador criado a partir das especificações salvas.

Como nos outros módulos, todas as características de entrada são salvas em um arquivo de texto, que é recuperado com a opção “Use an existing scheduler”. Esta funcionalidade permite que o usuário defina um conjunto de tarefas e suas características, e estas são escalonadas de acordo com as regras do escalonador.

3.4 Geração de escalonadores de ambientes multiprocessados

Para este ambiente foi realizado um estudo que verificou a melhor maneira de adaptar os métodos e objetivos do ambiente monoprocessado. Deste modo, o objetivo foi construir uma interface semelhante e com a mesma função da que foi apresentada na Seção 3.3.

A interface desenvolvida possui o mesmo padrão de janelas adotado para o ambiente monoprocessado. As diferenças estão apenas nas opções oferecidas.

Todos os escalonadores que serão gerados devem adotar o padrão de escalonamento de tarefas em "janela", o mesmo que os algoritmos nativos oferecem. O usuário é livre para determinar a quantidade de processadores e o tamanho da janela que será adotada.

Os parâmetros que podem ser especificados para cada tarefa inserida são:

- Processador que deverá executar
- Nome
- Carga
- Deadline

Todas as tarefas são do tipo aperiódica e suas ocorrências são geradas aleatoriamente pelo programa.

3.4.1 Implementação

Adaptação das interfaces gráficas para ambiente multiprocessado

Como foi dito anteriormente, para este módulo foi mantido o mesmo padrão de interface, ou seja as características são inseridas nas janelas sequenciais.

A única alteração na estrutura das janelas foi a inserção de uma nova interface com o painel de fórmula, sendo que esta é responsável por receber a fórmula que ordena as tarefas dentro da janela de execução. Portanto, agora o usuário deve inserir duas fórmulas, sendo a primeira responsável pela ordenação geral das tarefas e a segunda pela ordenação das tarefas dentro da janela.

Desenvolvimento dos métodos para execução

Os métodos de interpretação e geração da nova classe são exatamente os mesmos explicados na seção anterior para ambiente monoprocessado. Estes foram mantidos pois o objetivo é o mesmo, consistindo na geração de uma nova classe e de sua execução.

A alteração se dá no conteúdo da nova classe, em que foi adicionado um novo método chamado *ordenaRecurso()*, sendo este responsável pela forma de ordenação das tarefas na janela.

Assim como foi criada uma classe de apoio para o ambiente monoprocessado, criou-se uma classe responsável por controlar detalhes do escalonamento, sendo que esta é a responsável pela chamada do método citado acima e pelo controle das estruturas que são comuns ao escalonamento.

Adaptação da exibição gráfica dos resultados e da reutilização dos algortimos criados

Os algoritmos nativos do ambiente multiprocessado possuem como saída gráfica a exibição de uma interface com um quadro estatístico do escalonamento e o gráfico de execução. Estes foram adaptados para que o novo módulo fosse capaz de exibir os resultados no mesmo modelo.

O método utilizado para leitura de arquivo com dados de um escalonador já criado, foi adaptado para ler as informações de um ambiente multiprocessado. Este funciona exatamente da mesma maneira, sendo que o usuário escolhe o arquivo e o método executa a interpretação e carregamento na interface.

3.5 Considerações Finais

A o longo deste capítulo foi apresentado o projeto do módulo de geração de escalonadores do RTsim. A apresentação procurou detalhar os procedimentos adicionados ao simulador e também as alterações em componentes já existentes. Do aqui exposto é possível perceber a relevância e qualidade de projeto.

Capítulo 4

Testes e Resultados

Os testes realizados neste trabalho tiveram como objetivo verificar o comportamento da nova interface e os resultados que eram exibidos. A interação da interface de acordo com as ações do usuário e o seu comportamento na simulação foram analisados com o objetivo de conferir se a interface é intuitiva e de fácil utilização. Os resultados da simulação foram conferidos de acordo com o que era esperado sobre seu comportamento.

Este capítulo fará uma explicação sobre os testes e ilustrará como funcionam os novos métodos inseridos.

4.1 Testes de interfaces de entrada e modos prática

Testando a interface geral de inserção dos dados

Este teste exibe todos os componentes e opções da interface de entrada de dados. Para isso, será relatado um passo a passo com todas as janelas do método.

Passo 1 Definição do nome do escalonador:

A Figura 4.1 exibe o primeiro painel que o usuário deve utilizar, no qual deve ser inserido um nome para o novo escalonador.

Passo 2 Escolha do tipo de prioridade:

O painel exibido na Figura 4.2 é responsável por definir qual o tipo de prioridade que será adotada pelo escalonador. Sendo escolhida a prioridade dinâmica para este exemplo.

Passo 3 Escolha do tipo de inserção de fórmula:

No painel da Figura 4.3 é possível observar duas opções. A primeira opção leva o usuário a uma tela na qual ele pode escolher por uma fórmula de escalonador pré-estabelecida para ser utilizada (Figura 4.4). A segunda, leva o usuário ao

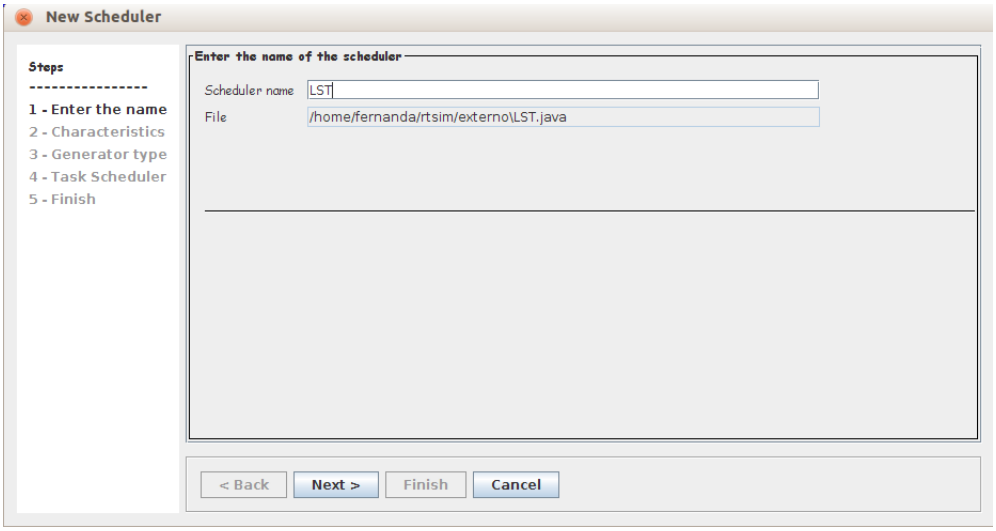

Figura 4.1: Janela para identificação do algoritmo e classe de tarefas

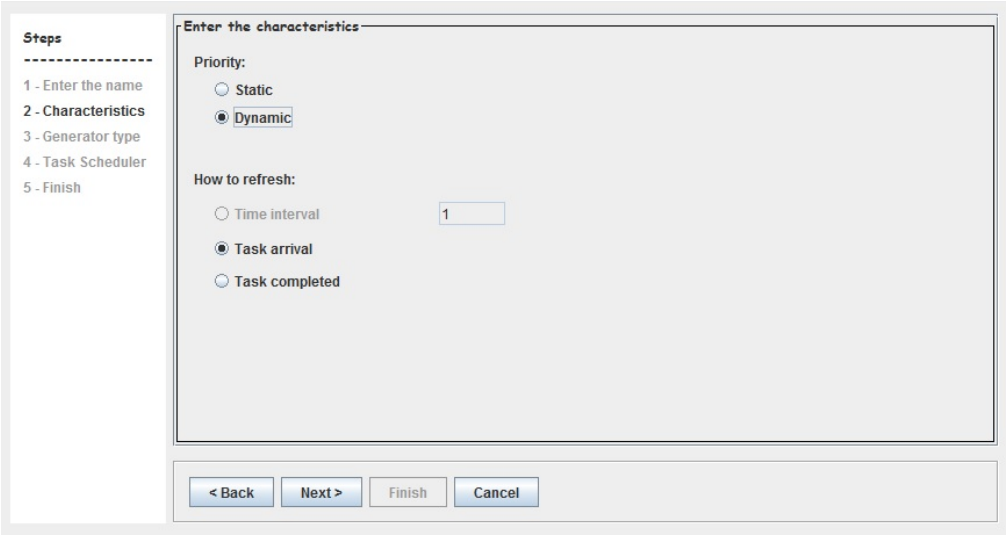

Figura 4.2: Janela de definição do modelo de escalonamento

painel de fórmula em que ele deve inserir o comportamento do escalonador na forma de uma equação matemática (Figura 4.5)

Passo 4 Painel de entrada da equação:

No painel da Figura 4.5 o usuário deve inserir a equação matemática que define o comportamento do escalonador, sendo que esta deve ser compostas pelos parâmetros disponíveis na interface.

Passo 5 Resumo das entradas:

Por fim, a Figura 4.6 exibe um resumo para conferência de tudo que usuário escolheu para o novo escalonador.

Passo 6 Utilização do algoritmo:

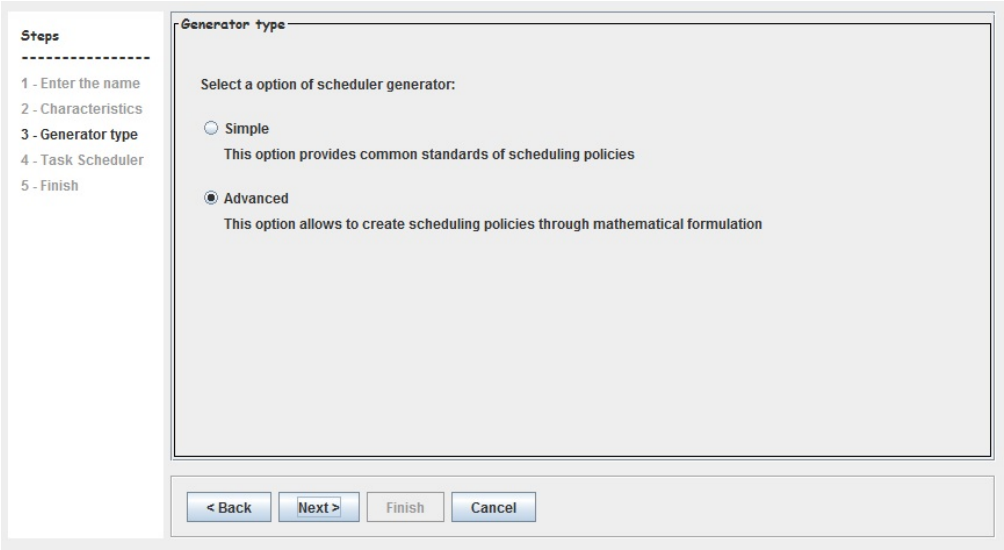

Figura 4.3: Janela de escolha do modo de especificação do algoritmo

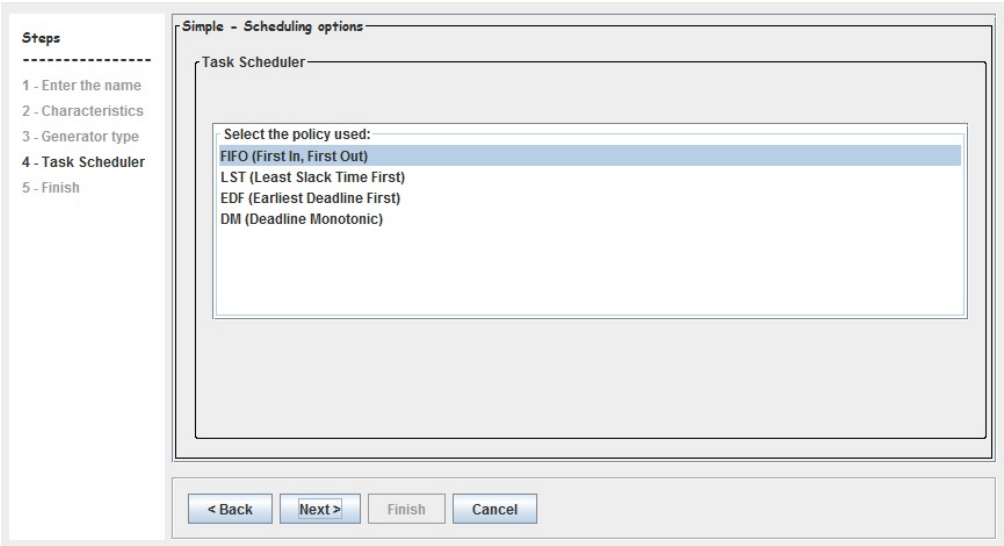

Figura 4.4: Janela para escolha de algoritmo pré-definido

A Figura mostra a interface para entrada de tarefas. Nela o usuário seleciona o escalonador que quer utilizar e insere o conjunto de tarefas que será escalonado.

Modo Aprendizado

O objetivo foi adaptar o modo aprendizado ja existente para que o novo módulo conseguisse usá-lo. Sendo assim, a entrada de dados no modo aprendizado é feita pelo mesmo conjunto de interfaces mostrado anteriormente.

A Figura 4.8 exibe o resultado da utilização do modo prática com o mesmo conjunto de tarefas. É possível observar que ele faz o acompanhamento das inserções do usuário e exibe informações ao longo da execução.

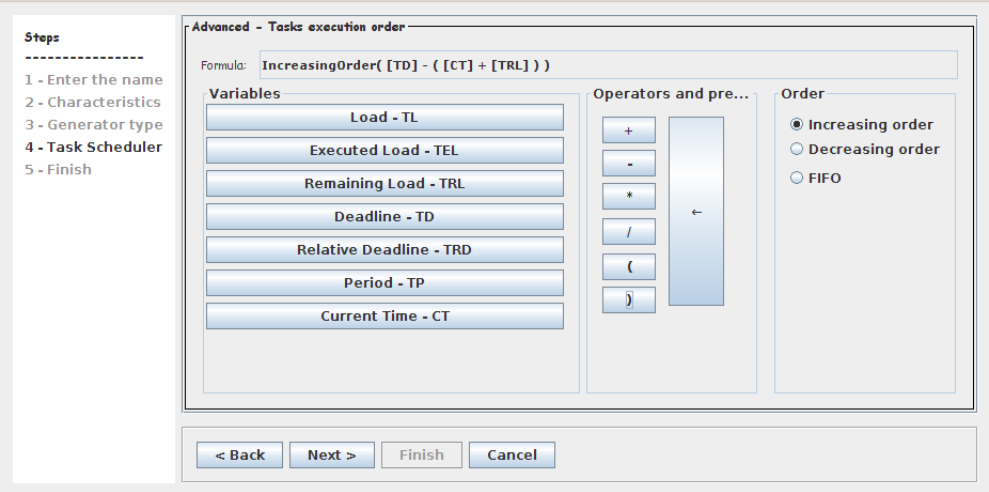

Figura 4.5: Janela para inserção de equação

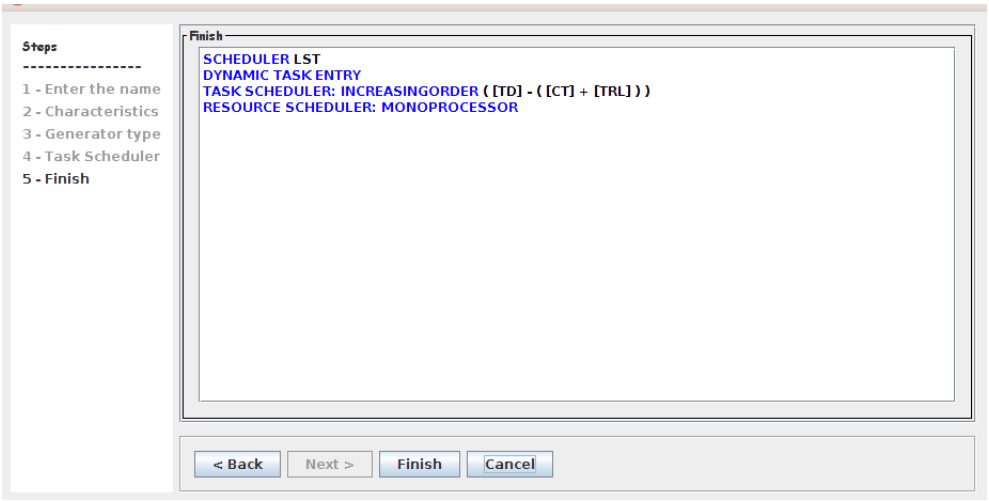

Figura 4.6: Janela de verificação da especificação

O objetivo foi alcançado, pois o modo foi adaptado para o novo módulo e funciona exatamente como o implementado para os outros algoritmos.

Modo Teste

O modo teste teve o mesmo objetivo do modo aprendizado, e o resultado também foi bem sucedido, pois se comporta da mesma maneira que o esperado.

A Figura 4.9 exibe a saída do módulo. Este funciona como um teste real, em que o usuário só conhece seus erros ou acertos ao final da inserção.

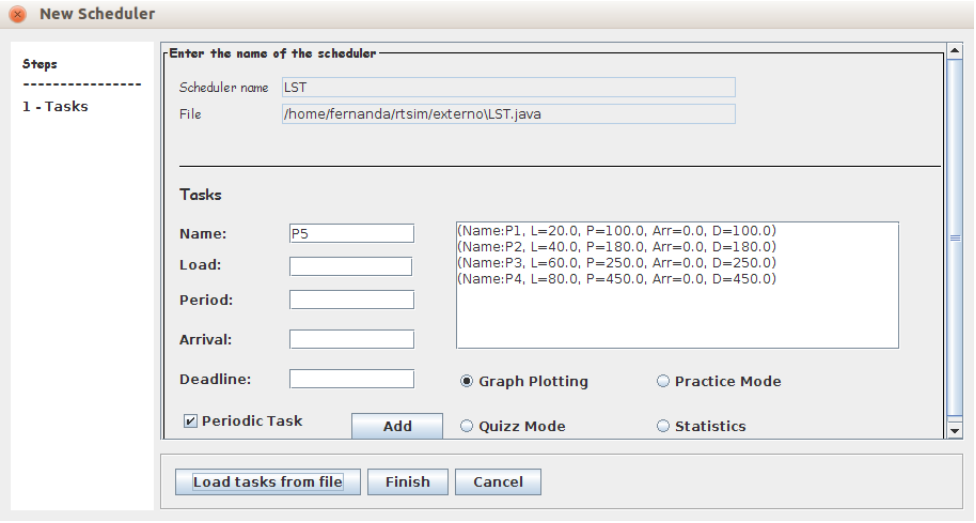

Figura 4.7: Janela para entrada de tarefas

Figura 4.8: Modo Aprendizado

4.2 Resultados dos testes de escalonamento

Além de uma interface intuitiva, a ferramenta precisa produzir o escalonamento correto das tarefas inseridas. Portanto são exibidos a seguir os testes com tipos de tarefas e escalonadores diferentes.

Executando o escalonamento Least Slack Time First - Algoritmo para ambiente monoprocessado e tarefas periódicas com prioridade dinâmica

Como foi dito ao longo deste trabalho, o algoritmo LST é fundamental para os estudos, mas não estava implementado de forma nativa. Aqui será descrito o resultado de uma implementação automática deste escalonador.

O conjunto de tarefas exibido na Tabela 4.1 foi simulado na ferramenta e o re-

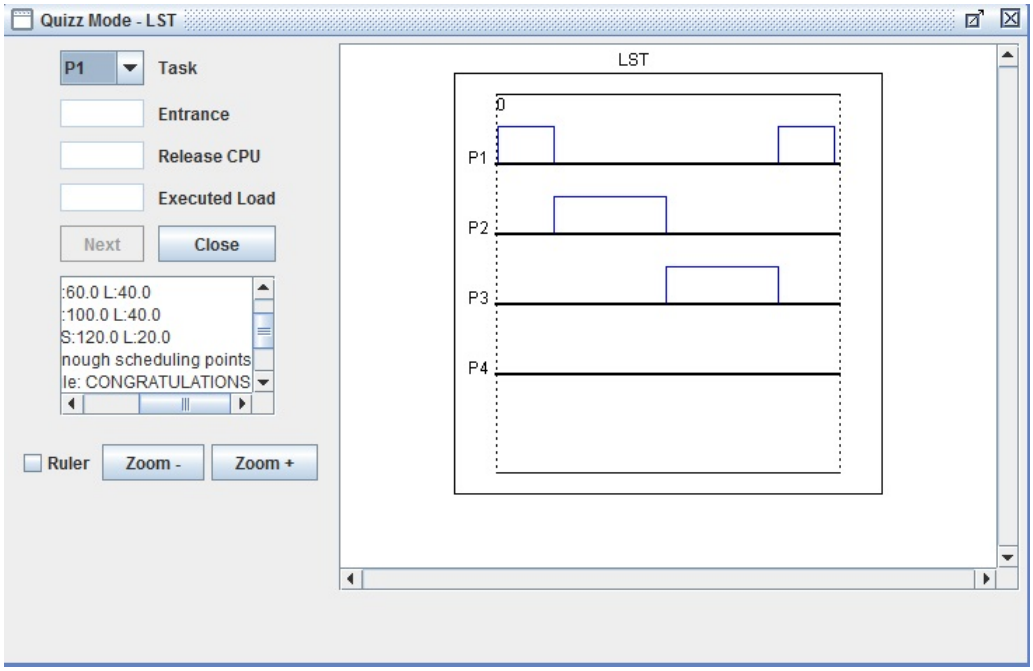

Figura 4.9: Modo Teste

sultado pode ser visto na Figura 4.10.

Tabela 4.1: Conjunto de tarefas utilizado no LST

Tarefa	Carga	Período	Deadline
P1	20	100	100
P2	40	180	180
P3	60	250	250
P4	80	450	450

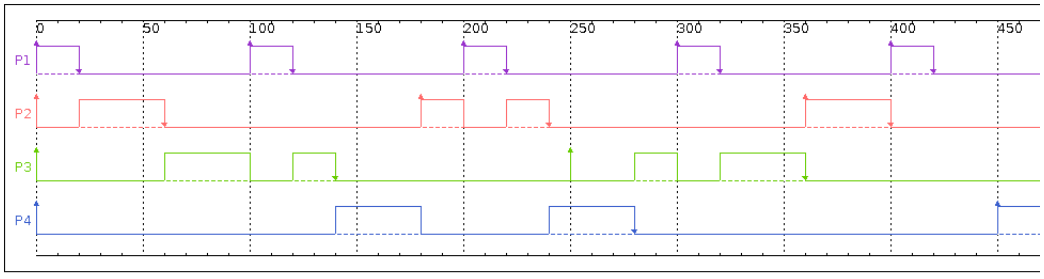

Figura 4.10: Saída gráfica do algoritmo LST

Na Figura 4.10 é possível destacar dois pontos importantes em que ocorre a preempção de uma tarefa em favor de outra com maior prioridade. A seguir é exibida uma descrição detalhada do cálculo que é feito pela ferramenta:

- Tempo=100: A tarefa P3 está executando e P1 chega na fila de pronto. A prioridade é recalculada para todas as tarefas, sendo que nesse instante a folga de P1 é 80 unidades e a folga de P3 é de 130 unidades, ocorrendo

então preempção de P3 em favor de P1.

- Tempo=200: Neste instante a tarefa P2 está executando com folga igual a 140 unidades, sendo que a tarefa P1 chega com folga de apenas 80 unidades, e mais uma vez ganha o direito de executar primeiro.

Diante do que foi exposto, é possível observar que o resultado da simulação foi correto, pois a simulação se comporta exatamente como o esperado de acordo com as regras do algoritmo especificado.

Executando o escalonamento Deadline Monotônico - Algoritmo para ambiente monoprocessado e tarefas periódicas com prioridade fixa

O mesmo conjunto de tarefas visto anteriormente foi executado para o deadline monotônico. A Figura 4.11 exibe o resultado do escalonamento, este possui definição estática de prioridades, ordenando de acordo com os valores de deadline relativo.

Nesta execução é possível observar que as prioridades que foram estabelecidas no início da execução são mantidas, independente do que ocorra durante a execução.

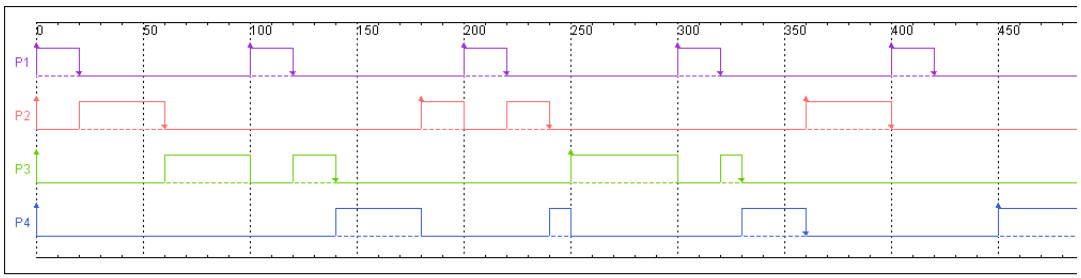

Figura 4.11: Escalonamento produzido pelo Deadline Monotônico

Executando algoritmos monoprocessados com tarefas aperiodicas no conjunto

Para testar as diferentes formas de tratamento do ambiente monoprocessado, foi adotado o conjunto de tarefas exibido na Tabela 4.3.

É importante lembrar que neste módulo as tarefas periódicas são escalonadas de acordo com a fórmula inserida e as aperiódicas podem ser escalonadas de três modos previamente definidos. Para os testes foi adotado o escalonamento em ordem crescente de carga para tarefas periódicas e foi variado o tratamento de aperiódicas para o conjunto.

Tabela 4.2: Conjunto de tarefas utilizado no teste de aperiódicas

Tarefa	Carga	Período	Deadline	Ocorrências
P1	30	100	100	-
P2	15	120	200	-
A1	10	-	200	10
A2	10	-	400	15
A3	40	-	200	40 80 120

i) Tratamento de aperiódicas em Background:
Neste tratamento as aperiódicas só podem executar nos tempos de folga do processador, sendo entendidas como as de menor prioridade. A Figura 4.12 exibe o resultado obtido, no qual é possível observar a execução das tarefas A1,A2 e A3 apenas nos tempos de folga do processador. Também é exibido que P2 possui maior prioridade do que P1, e isto se deve ao fato de P2 ter a menor carga de execução.

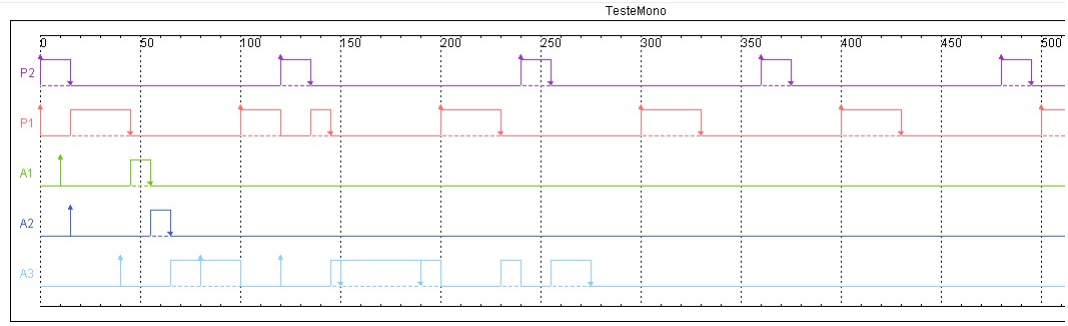

Figura 4.12: Tratamento em Background

ii) Tratamento de aperiódicas por Interrupção:
Para este tipo de tratamento, as tarefas aperiódicas devem ter a maior prioridade no tratamento, ou seja executar mesmo que tenha que realizar preempção de uma periódica. A Figura 4.13 exemplifica o que foi dito acima, exibindo as trocas de tarefas sempre favorecendo as aperiódicas.

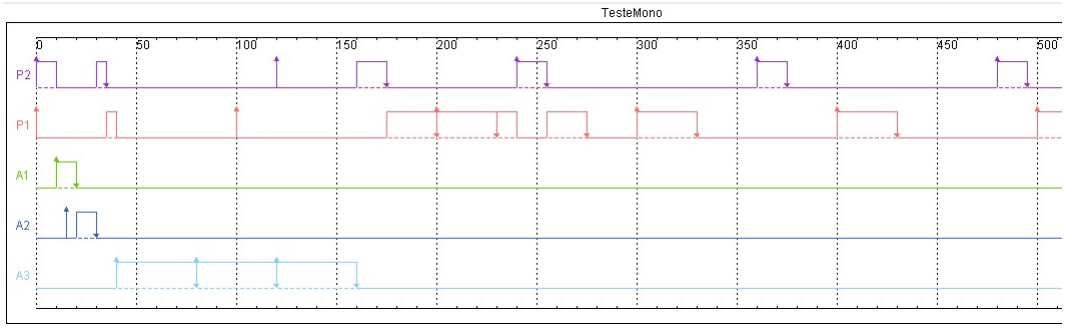

Figura 4.13: Tratamento por Interrupção

iii) Tratamento de aperiódicas por Servidor de Polling:

Para este teste foi criada um tarefa periódica servidora com carga de 30 unidades e período de 120 unidades que servirá para indicar quando as aperiódicas poderão executar. A Figura 4.14 torna possível visualizar que as tarefas A1,A2 e A3 tem agora tempos específicos para execução, sendo que este é o tempo em que a tarefa servidora ganha direito de executar.

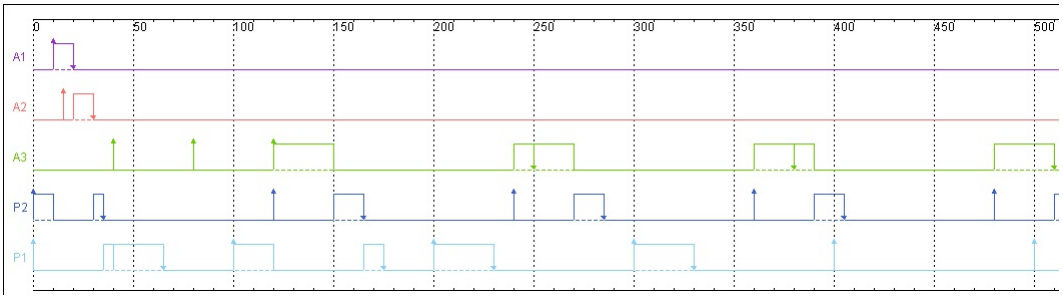

Figura 4.14: Tratamento por Servidor de Polling

Executando algoritmos em ambiente multiprocessado

Como foi dito anteriormente, os algoritmos multiprocessados tem duas etapas de ordenação. A ordenação das tarefas de forma geral é dada pela primeira fórmula inserida pelo usuário, e a segunda fórmula trata da ordenação das tarefas na janela de execução. Também é possível que se escolha a mesma ordenação nas duas fórmulas.

A interação e inserção de elementos nas janelas é exatamente a mesma, isto porque o módulo multiprocessado foi criado com base no mesmo padrão de monoprocessado, tendo apenas algumas adaptações.

A Tabela 4.3 exibe o conjunto de tarefas que foi usado para testar o funcionamento do novo módulo. É importante destacar que o novo método, assim como os algoritmos nativos do ambiente, aceita a alocação de recursos para as tarefas.

Tabela 4.3: Entrada do algoritmo multiprocessado

Tarefas	Processador	Carga	Deadline	Recursos
A1	P1	10	20	A B D
A2	P1	50	100	E
A3	P1	60	150	T
A4	P1	10	120	C
A5	P2	30	100	E
A6	P2	25	60	W
A7	P2	5	25	T

O resultado gráfico é exibido na Figura 4.15, na qual é possível observar as tarefas definidas para o processador P1, e na Figura 4.16 definidas para P2, ordenadas

de acordo com o parâmetro definido pelo usuário, sendo neste exemplo a carga.

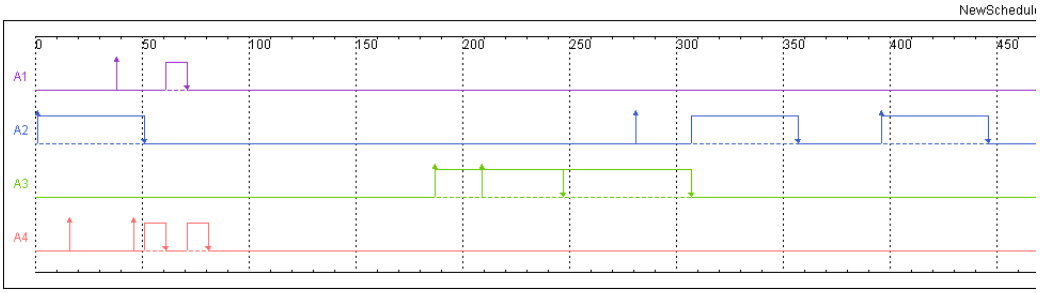

Figura 4.15: Escalonamento em ambiente multiprocessado

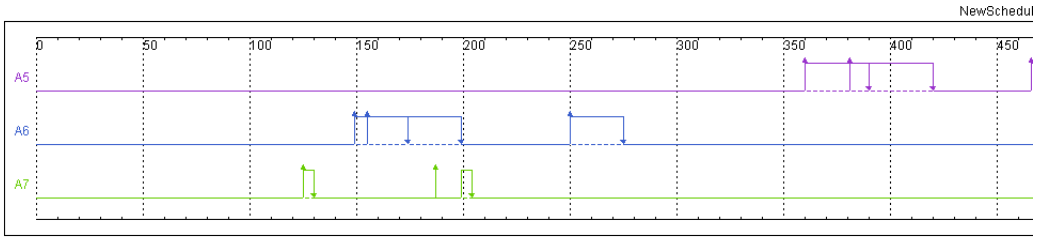

Figura 4.16: Escalonamento em ambiente multiprocessado (Processador P2)

É importante destacar que estes algoritmos que serão gerados tem a característica de não admitir preempção. A ordem de prioridade definida é respeitada para a definição de qual executará, mas não é suficiente para fazer a preempção de uma tarefa para execução de outra.

4.3 Considerações Finais

Neste capítulo foi exibido todo o funcionamento da ferramenta e as interações possíveis com a interface. Aqui foi possível perceber os objetivos alcançados e o funcionamento do novo módulo de maneira sólida e eficaz.

Capítulo 5

Conclusões e Trabalhos Futuros

5.1 Conclusões

Diante do que foi exposto nos capítulos iniciais deste trabalho foi possível observar a importância de atividades práticas no estudo de sistemas de tempo-real. A ferramenta que foi apresentada já realizava um bom trabalho no seu estado de implementação, sendo que este projeto foi proposto de modo a torná-la ainda mais completa e funcional.

Os novos métodos desenvolvidos receberam uma nova interface gráfica, composta em janelas e visando principalmente a facilidade de uso. Portanto, para que não ocoresse diferenças de padrão nas partes gráficas, foi feita uma reestruturação das interfaces já existentes, de modo que o resultado fosse o mais próximo possível da interface do método.

O projeto apresentado teve sucesso ao cumprir os objetivos que foram propostos. A ferramenta está agora em um nível superior de funcionalidade, principalmente porque agora atende uma abrangência maior de necessidades dos usuários. Esta versão atende melhor aos alunos e professores que a utilizam, mas principalmente torna a ferramenta mais atrativa aos profissionais.

Este trabalho foi fundamental para promover novas possibilidades de uso para o RTsim, e principalmente para atrair novos usuários e interessados na área. A nova versão foi desenvolvida focando sempre em torná-la funcional e intuitiva, para facilitar seu uso e aceitação entre os usuários.

5.2 Trabalhos Futuros

O trabalho futuro que pode ser proposto para este projeto é uma ampliação ainda maior das funcionalidades existentes. Um estudo dos algoritmos que não se encaixam no padrão da ferramenta e uma proposta de modificações pode ser realizada, e tornar a ferramenta ainda mais abrangente.

O RTsim está com a interface abrangente e reestruturada, o que permite que mais modificações sejam feitas sem realizar trabalho desnecessário. Além disso, seu código modularizado facilita implementações futuras, sendo possível inserir novos módulos a qualquer momento na ferramenta.

Referências Bibliográficas

[Audsley et al., 1994] Audsley, N., Burns, A., Richardson, M., and Wellings, A. (1994). Stress: A simulator for hard real-time systems. *Software-Practice and Experience*, pages 543–564.

[Ch'eramy et al., 2013] Ch'eramy, M., D'eplanche, A.-M., and Hladik, P.-E. (2013). Simulation of real-time multiprocessor scheduling with overheads. In *Intl Conf on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2013)*, pages 5–14.

[Diaz et al., 2007] Diaz, A., Batista, R., and Castro, O. (2007). Realtss: a real-time scheduling simulator. In *Electrical and Electronics Engineering, 2007. ICEEE 2007. 4th International Conference on*, pages 165–168. IEEE.

[Farines et al., 2000] Farines, J.-M., Fraga, J. d. S., and Oliveira, R. d. (2000). Sistemas de tempo real. *Escola de Computação*, 2000:201.

[Goncalves, 2005] Goncalves, T. (2005). Rtsim 2.0 - reengenharia e novas funcionalidades. Monografia de Projeto Final de curso, UNESP.

[GSPD, 2016] GSPD (2016). Gspd's homepage. Disponível em <http://www.dcce.ibilce.unesp.br/spd/>, acessado em 15 de junho de 2016.

[Kehdy, 1999] Kehdy, R. (1999). Simulação de algoritmos de escalonamento de tarefas de tempo-real em sistemas distribuídos. Monografia de Projeto Final de curso, UNESP.

[Leung, 1989] Leung, J. Y.-T. (1989). A new algorithm for scheduling periodic, real-time tasks. *Algorithmica*, 4(1-4):209–219.

[Leung and Whitehead, 1982] Leung, J. Y.-T. and Whitehead, J. (1982). On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance evaluation*, 2(4):237–250.

[Liu and Layland, 1973] Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61.

[Liu, 2000] Liu, J. W. S. (2000). *Real-Time Systems*. Prentice Hall.

[Manacero Jr. et al., 2001] Manacero Jr., A., Miola, M., and Nabuco, V. (2001). Teaching real-time with a scheduler simulator. In *Proc. of the 31st Frontiers in Education Conference*, pages T4D15–19. IEEE Press.

[Miola, 2001] Miola, M. (2001). Reengenharia aplicada ao rtsim. Monografia de Projeto Final de curso, UNESP.

[Nikolic et al., 2011] Nikolic, B., Awan, M. A., and Petters, S. M. (2011). Sparts: Simulator for power aware and real-time systems. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 999–1004. IEEE.

[Nissanke, 1997] Nissanke, N. (1997). *Realtime Systems*. Prentice Hall.

[Ramamritham et al., 1990] Ramamritham, K., Stankovic, J. A., and Shiah, P.-F. (1990). Efficient scheduling algorithms for real-time multiprocessor systems. *Parallel and Distributed Systems, IEEE Transactions on*, 1(2):184–194.

[Shin and Ramanathan, 1994] Shin, K. G. and Ramanathan, P. (1994). Real-time computing: A new discipline of computer science and engineering. *Proceedings of the IEEE*, 82(1):6–24.

[Strosnider et al., 1995] Strosnider, J. K., Lehoczky, J. P., and Sha, L. (1995). The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. *IEEE Transactions on Computers*, 44(1):73–91.

[Yaashuwanth and Ramesh, 2010] Yaashuwanth, C. and Ramesh, R. (2010). Web-enabled framework for real-time scheduler simulator (a teaching tool). In *Computer Research and Development, 2010 Second International Conference on*, pages 826–830. IEEE.