Simulador *Web* de Algoritmos para Escalonamento de Processos em um Sistema Operacional

Emerson L. de Moraes[1], Henrique Y. O. Asakura[1], Ricardo P. Bonfiglioli[1] Murilo da S. Dantas[1, 2]

[1]Tecnologia em Banco de Dados – Faculdade de Tecnologia de São José dos Campos (FATEC-SJC)
12247-004 – São José dos Campos – SP – Brasil

[2]Instituto Nacional de Pesquisas Espaciais (INPE)
12227-010 – São José dos Campos – SP – Brasil

emerson.moraes@fatec.sp.gov.br, rique.hasakura@gmail.com,
ricardo.bonfiglioli@gmail.com, murilodantas06@gmail.com

Abstract. *An operating system has several complex software concepts and their learning can be compromised. In general, this is due to abstraction required, its complexity and its excessive number of features such as process management, synchronization, deadlock handling, and in particular, process scheduling, focus of this work. The simulation algorithm increases the speed of learning, in practice it shows the operation of the system itself. By developing systems simulators of the concepts covered in class, it has been a substantial increase in the absorption of the contents of computing by students.*

Resumo. *Um sistema operacional possui diversos conceitos complexos de software e o seu aprendizado pode ser comprometido. Em geral, isso ocorre devido à abstração necessária, à sua complexidade e ao seu número excessivo de funcionalidades, como gerenciamento de processos, sincronismo, tratamento de impasse e, em especial, no escalonamento de processos, foco deste trabalho. A simulação de algoritmos eleva a velocidade de aprendizado, pois demonstra na prática o funcionamento do sistema em si. Ao desenvolver sistemas simuladores dos conceitos abordados em sala de aula, tem-se um incremento substancial na absorção de conteúdos da computação por parte dos alunos.*

1. Introdução

A disciplina de Sistemas Operacionais faz parte dos fundamentos da computação e, sendo assim, é obrigatoriamente integrante da grade curricular de todos os cursos de graduação em computação e informática [SBC 2003]. As principais funções de um sistema operacional são o gerenciamento e a disponibilização dos recursos de hardware e software para o usuário. "O sistema operacional constitui um conjunto de software extenso, complexo e de difícil compreensão global, por executar tarefas bastante diversas, mas complementares e interdependentes" [Maziero 2002]. De acordo com Machado e Maia (2004) "a experiência de professores e alunos tem mostrado como é grande a dificuldade em ensinar e compreender os conceitos teóricos, aplicação prática e técnicas apresentadas".

Atualmente, o modelo clássico de ensino segue um referencial teórico e recursos visuais estáticos, como figuras e esquemas. Esta metodologia está alicerçada na metodologia em que o professor é o único agente ativo no processo de ensino. Porém, para que o aprendizado de conceitos complexos, como os de sistemas operacionais, seja completo, é necessário, em conjunto com a abordagem teórica, utilizar recursos práticos em que o aluno participe ativamente na construção do próprio conhecimento.

Com o advento da Internet, a disseminação dos conhecimentos das mais diversas áreas tem se tornado mais fácil. A infraestrutura da Internet possibilita a publicação gratuita de conteúdos, vídeos, chats, fóruns e de software online capaz de simular conceitos de difícil compreensão.

O objetivo principal deste trabalho foi elaborar uma ferramenta de auxílio aos alunos e professores interessados no uso de uma ferramenta educacional com uma abordagem didática dos conceitos de escalonamento de processos da disciplina de Sistemas Operacionais. A ferramenta pretende ser de fácil acesso, interativa e intuitiva, por meio de imagens e simulações animadas, para complementar o aprendizado. Criada para ser multiplataforma, essa ferramenta foi desenvolvida apenas com Software Livre.

Este artigo aborda primeiramente, como é feito o ensino da matéria de Sistemas Operacionais, e as teorias instrutivista e construtivista. Em seguida, como a disciplina é tratada de uma forma prática, e como se encaixa a proposta desse projeto nesse contexto; posteriormente são apresentados os resultados obtidos. E por ultimo, as considerações finais e propostas futuras.

2. Ensino-aprendizagem de Sistemas Operacionais

A disciplina de Sistemas Operacionais atualmente, na maioria das universidades é abordada de uma forma de ensino instrutivista. Essa metodologia consiste em um mestre ativo que detém o conhecimento sobre o assunto tendo a função de simplificar o conteúdo e repassar aos aprendizes que, por sua vez, geralmente buscam absorver as informações passivamente. Contudo, esse método tem suas falhas, pois as aulas são muito teóricas, e muitas vezes os conceitos não são plenamente absorvidos [MAZIERO 2002].

2.1. Construtivismo e Sistemas Operacionais

Os atuais jovens estudantes nos cursos superiores são descritos como a Geração Y. Uma das características desses jovens é a dificuldade de concentração por muito tempo numa aula expositiva sem que haja interatividade e fácil compreensão. Além disso, esses indivíduos acompanham o mundo da tecnologia avançar rapidamente, e por isso necessitam de recursos rápidos, audiovisuais e de fácil acesso para obter eficaz aprendizado e retenção de informações [LOIOLA 2009].

Em decorrência disso, o ensino de uma disciplina como a de Sistemas Operacionais, repleta de conceitos complexos e abstratos, pode se beneficiar com o uso de recursos baseados nas teorias construtivistas de Piaget (1932), segundo as quais mestre e aluno devem ter uma relação interativa no processo ensino-aprendizagem. É possível descrever algumas tentativas nesse sentido baseadas nessas ideias, como pode ser visto a seguir.

2.1.1. Laboratórios Supervisionados

Com o intuito de complementar e facilitar o estudo do conteúdo da matéria de Sistemas Operacionais, alguns pesquisadores, como Downey (1999) propõem laboratórios supervisionados – "closed labs" – onde os alunos possam estudar de forma prática os conceitos e algoritmos da disciplina de Sistemas Operacionais. Nesses ambientes computacionais, são instalados alguns sistemas que auxiliam no entendimento da matéria, fornecendo simulações e desenvolvimento de algoritmos [MACHADO e MAIA 2004].

O uso de pequenos projetos práticos é basicamente a utilização de recursos do sistema para o entendimento do funcionamento e dos conceitos do sistema operacional. Para isso os alunos utilizam a interação com o sistema via linha de comandos e a programação de chamadas de sistemas. Este é considerado o estudo prático mais fácil de implementar nos laboratórios supervisionados e fornece um grande entendimento do sistema aos alunos [MAIA e PACHECO 2003].

A maioria dos laboratórios requer que os alunos tenham uma boa noção de algumas linguagens de programação, como o C, Java e Pascal [Machado e Maia 2004]. Outra desvantagem nesse enfoque é o fato de que os alunos só usam o sistema e entendem o que ele faz e pra que serve, mas não estudam o desenvolvimento do sistema e a sua estrutura interna [MAZIERO 2002].

Em contrapartida, a exploração de código em sistemas reais compreende-se da depuração e modificação dos códigos do sistema, realizando testes em alguns sistemas como o próprio Linux. Esse tipo de abordagem prática do sistema é considerado difícil de implementar e os alunos necessitam de um grande conhecimento em arquitetura e programação de sistemas operacionais. Por esses motivos, somente alguns cursos têm o tempo e condições necessárias para disponibilizar esse tipo de laboratório supervisionado aos alunos [MACHADO e MAIA 2004].

2.1.2. Simulação de algoritmos

Nesta abordagem são criados programas que simulam algoritmos usados em sistemas operacionais tais como de escalonamento de CPU, substituição de páginas de memória, escalonamento de acesso a disco, entre outros. Estes programas geram resultados estatísticos ou animações gráficas para ajudar na compreensão desses algoritmos [MAZIERO 2002].

Também conhecida como *visualização de algoritmos*, esta abordagem favorece o aprendizado, pois propõe a observação e a interação com a simulação. Simulações assim são frequentemente encontradas na Internet na forma de *applets* Java [MAZIERO apud THOMSON LEARNING, Inc].

Entretanto, vale ressaltar que segundo Stasko (1997), a simples observação e interação com as simulações podem ser insuficientes para compreender as diferenças sutis entre o algoritmo simulado e o real. Ocorre, além disso, a simplificação e isolamento de aspectos do sistema de tal maneira a não propiciar uma visão global e integrada do mesmo. E ainda, "a implementação da simulação poderá se mostrar excessivamente trabalhosa, dependendo da sofisticação gráfica exigida" [MAZIERO 2002].

Apesar dos inconvenientes acima, esta abordagem fornece um entendimento efetivo de cada funcionalidade do Sistema Operacional proporcionando um complemento nos estudos desta disciplina. No contexto das disciplinas de SO, um simulador muito conhecido é o SOSim [Machado e Maia 2004], que simula o escalonamento e a gerência de processos e a gerência de memória virtual por paginação. Além do SOSim, outros simuladores relacionados à simulação da gerência de processos são o wxProc [Rocha et al 2004], que simula políticas de escalonamento, e apresenta os algoritmos relacionados a gerência de processos; e a ferramenta proposta por [Isotani et al 2009], que permite capturar os dados sobre os processos em execução no computador.

3. Desenvolvimento do Simulador

Neste trabalho, os algoritmos de sistemas operacionais escolhidos para serem simulados, no sentido de testar a viabilidade da tecnologia utilizada, foram os de escalonamento de CPU. Segundo Silberschatz, Galvin e Gagne (2008) "o escalonamento de CPU é a base dos sistemas operacionais multiprogramados. Alternando a CPU entre os processos, o sistema operacional pode tornar o computador mais produtivo".

Foram abordadas as seguintes políticas de escalonamento: (i) *First Come, First Served* (FCFS), onde o primeiro processo a chegar será o primeiro a ser atendido; (ii) *Shortest Job First* (SJF), onde o menor processo é atendido preferencialmente; e o *Round-Robin (time-slicing),* onde é dada uma fatia de tempo para cada processo e é a base para compartilhamento de tempo.

3.1. Tecnologias Utilizadas

Atualmente, no desenvolvimento de aplicações para a Internet tem se utilizado um conjunto de tecnologias chamado AJAX (*Asynchronous Javascript And* XML), que trouxe muito mais interação para a *Web*. Porém, surgiram alguns problemas, sendo o mais evidente deles a incompatibilidade do *JavaScript* em navegadores diferentes, devido ao crescimento da quantidade de código. Para resolver esse problema, alguns *frameworks* foram criados prometendo simplificar o desenvolvimento de aplicações com AJAX. O *framework* Java da Google para esse fim é o *Google Web Toolkit* (GWT) [Junior 2007]. O GWT tem a capacidade para a criação de interfaces amigáveis com alto nível de interação com o usuário final, pois, atualmente, existem várias bibliotecas de componentes gráficos para GWT, algumas desenvolvidas pela empresa Google e outras pela comunidade, que possui muitos colaboradores.

O grande diferencial desse *framework* em relação aos demais é que o programa pode ser escrito inteiramente em Java como se fosse um sistema *desktop*. Ou seja, o GWT abstrai toda a parte de AJAX, compatibilidade com navegadores, entre outras coisas que os desenvolvedores teriam que dominar para desenvolver esse nível de interação na Internet. Isso é feito por via de um compilador nativo ao GWT que transforma o código Java em *JavaScript*.

A programação com base no GWT é feita em dois módulos: (i) lado do usuário (*client-side*), onde são programadas as classes que vão se tornar a interface com o usuário, ou seja, tudo que se tornará *JavaScript* e (ii) lado do servidor (*server-side*), onde são programadas as regras de negócio do sistema. Nesse lado, o desenvolvedor

pode utilizar todo o poder da linguagem Java para segurança, persistência em banco de dados, etc. [Soares, Filgueiras e Silva 2009].

3.2. Requisitos do Sistema

A simulação algoritmos é composta de três etapas: entrada de parâmetros, animação e resultado resumido.

Na entrada de parâmetros o usuário deve informar os seguintes tipos de valores:

- A quantidade de processos presentes na fila de prontos para serem atendidos pela CPU;
- O tempo de chegada de cada processo na fila de espera;
- O tempo de serviço necessário para cada processo ser executado por completo;
- A política de escalonamento a ser simulada.

Depois dos parâmetros inseridos pelo usuário e confirmados pelo mesmo a animação é apresentada em forma de gráfico de duas dimensões. A dimensão horizontal representa a escala de tempo de cada processo na CPU. A dimensão vertical identifica os processos. O gráfico é desenhado de acordo com o algoritmo escolhido.

O resultado mostra, além das informações de entrada, o tempo de término de cada processo e algumas das propriedades características usadas na comparação de algoritmos de escalonamento. Estas propriedades são:

- *Turnaround time* (tempo de retorno): tempo desde a chegada de cada processo na fila de prontos até seu término;
- Tempo de espera: tempo gasto por um processo na fila de prontos;
- Tempo de resposta: tempo que levou para um processo ser inicialmente executado pela CPU desde sua submissão.

3.3. Arquitetura do Sistema

O protótipo do sistema de simulação funciona no modelo Cliente-Servidor (veja Figura 1). No lado cliente, o simulador possui os seguintes artefatos: (i) a interface gráfica utilizada pelo usuário que será exibida no navegador; (ii) as classes que se utilizam do RPC (*Remote Procedure Calls*) do GWT para que seja possível chamar os serviços que são executados no lado servidor da aplicação; e (iii) o *Comet Client*, que é uma classe GWT com a função de receber as atualizações que o servidor mandar em tempo real.

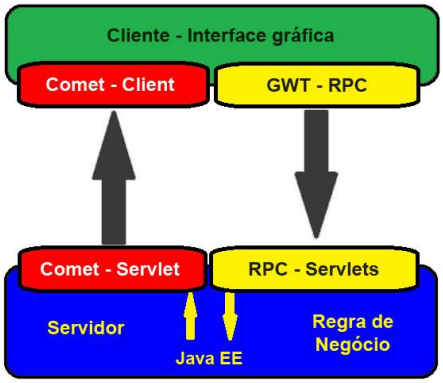

Figura 1: Arquitetura do Sistema.

Já no lado servidor da aplicação, estão localizadas: (i) as *servlets* RPC que atende as requisições dos usuários; e a (ii) *Comet Servlet*, que atualiza o cliente de acordo com a regra de negócio do sistema.

3.4. Modelo Conceitual do Sistema

Na Figura 2 é possível visualizar as responsabilidades do sistema, conforme os requisitos supracitados.

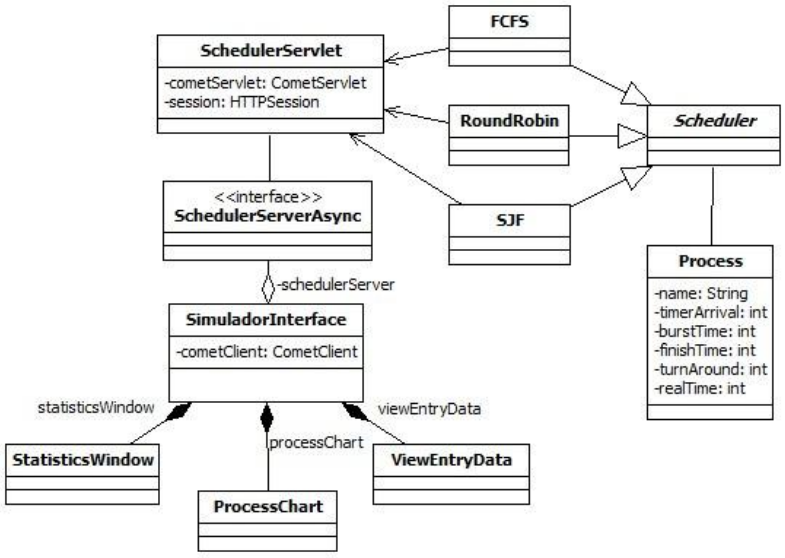

Figura 2: Modelo Conceitual do Sistema.

Do lado do cliente, possui as seguintes funcionalidades: (i) ***SimuladorInterface***, que é a base da interface gráfica do sistema e controla todos os componentes gráficos da aplicação; (ii) ***ViewEntryData***, que recebe os dados especificados pelo usuário para que a simulação seja realizada; (iii) ***ProcessChart***, que exibe a animação em tempo real da execução dos processos em forma de gráfico; (iv) ***StatisticsWindow***, que apresenta os resultados de cada processo após a execução do algoritmo; e (v) ***SchedulerServerAsync***, que realiza a comunicação com a *servlet* que está localizada no servidor.

Já no servidor, as funcionalidades são: (i) ***SchedulerServer***, que é a *servlet* que recebe as requisições do cliente e inicia a execução do algoritmo escolhido; (ii) ***Process***,

que é a representação de um processo com dados necessários para gerar os resultados; e (iii) ***Scheduler***, que possui as características genéricas a todos os algoritmos de escalonamento (*FCFS, SJF e RoundRobin*).

4. Apresentação do Simulador

O simulador apresentado neste trabalho, conforme a descrição acima possui três telas: uma para a entrada de dados, outra para demonstrar a simulação dos algoritmos e a última para fazer uma comparação entre os resultados.

4.1. Entrada de dados

A entrada dos dados necessários para que seja realizada a simulação dos algoritmos é feita a partir da primeira interface (veja Figura 3), que permite ao usuário escolher os parâmetros para realizar o escalonamento desejado. Nesta tela existem os seguintes campos:

- **Número de Processos:** define o número de processos que serão simulados;
- **Tipo de Escalonamento:** campo onde é selecionado o tipo de algoritmo de escalonamento que será utilizado na simulação;
- **Tempo de Chegada (*Arrival Time*):** seleciona em qual tempo o processo chegará à fila de prontos para que possa ser executado pela CPU;
- **Tempo de Serviço (*Burst Time*):** define o tempo de CPU que o processo irá precisar para ser executado por completo. Tanto esse campo, quanto o de Tempo de Chegada são gerados dinamicamente pela interface para cada processo de acordo com o que foi selecionado em Número de Processos;
- **Botão Simular:** após serem selecionados os parâmetros o botão Simular manda ao servidor os dados para que possa ser executado o algoritmo.

Figura 3: Primeira Interface da Aplicação.

Inicialmente os dados que vêm selecionados nos campos Tempo de Chegada e Tempo de Serviço são gerados de forma aleatória pelo sistema, podendo ser alterados na forma que o usuário desejar.

4.2. Simulação

Depois que a entrada de dados é concluída a aplicação passa para a interface de simulação (veja a Figura 4). Essa segunda tela é composta pelos seguintes componentes:

- **Gráfico:** ponto chave da aplicação em questão de animação, pois ele é atualizado constantemente pelo servidor, conforme o processamento do algoritmo, facilitando assim a visualização por parte do usuário do que está ocorrendo no tipo de escalonamento selecionado. Este gráfico é composto de dois eixos: o eixo x corresponde aos processos que estão em execução, e o eixo y corresponde aos tempos de CPU. Quando ocorrem as atualizações desse gráfico, aparecem os tempos utilizados pelo processo;
- **Rótulo de Processos:** nesse rótulo o usuário pode visualizar qual processo está sendo atendido pela CPU em tempo real. Quando o escalonamento é finalizado aparece uma mensagem de finalização nesse componente;
- **Botão Voltar:** esse botão tem a função de fazer com que a aplicação retorne a tela de entrada de dados, para que o usuário possa fazer outras simulações.

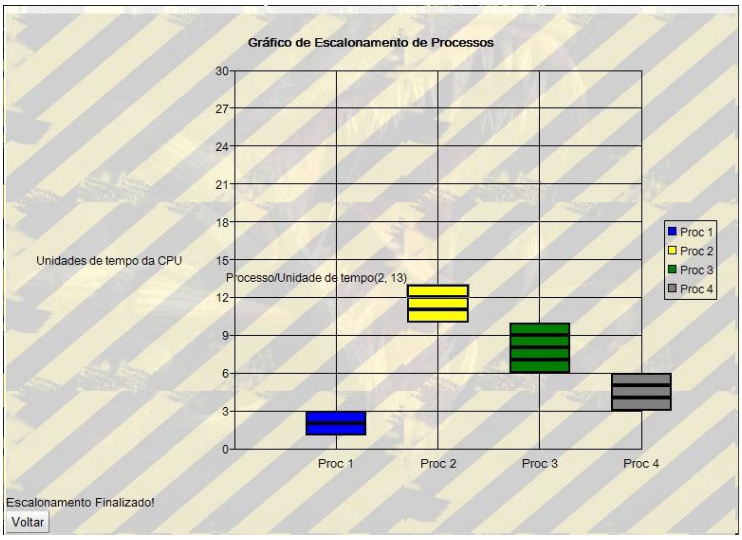

Figura 4: Interface de Simulação.

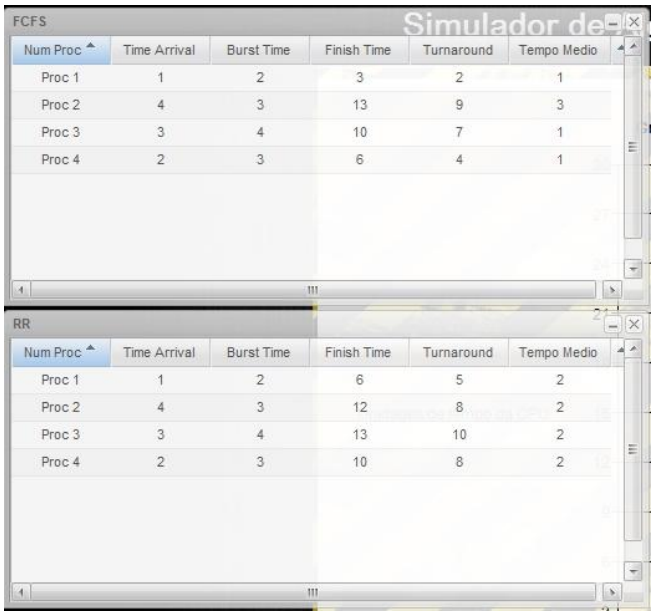

Figura 5: Resultado da Simulação.

4.3. Resultado da Simulação

A terceira tela (veja a Figura 5) do sistema fornece uma visualização de todos os dados relevantes em relação aos processos que foram usados na simulação, para que possam ser analisados pelo usuário. Esses dados são: Número do Processo, *Time Arrival* (tempo de chegada do processo), *Burst Time* (tempo de execução), *Finish Time* (tempo em que o processo finalizou), *Turnaround* (tempo que o processo levou para terminar desde sua chegada à fila de prontos) e Tempo Médio (tempo em média que o processo ficou na fila de prontos).

No caso da Figura 5 demonstra-se também outra funcionalidade do simulador, que é a possibilidade do usuário fazer comparações entre algoritmos e tempos diferentes, pois, essa janela de dados é gerada paralelamente às outras. O usuário pode deixá-la aberta, executar outra simulação e compará-las.

5. Considerações finais

Haja vista a dificuldade encontrada na aprendizagem dos conceitos da disciplina de Sistemas Operacionais torna-se importante o desenvolvimento e o uso de ferramentas computacionais didáticas como o ambiente de simulação proposto neste trabalho.

Este ambiente gera animações que ilustram o funcionamento dos principais algoritmos de escalonamento, de forma que o usuário possa melhor compreendê-los. Os resultados permitem fazer comparações entre os algoritmos simulados, para se observar as peculiaridades de cada um e suas diferenças.

Em suma, com este trabalho foi possível desenvolver um ambiente educacional *web* com o uso de tecnologias abertas e atuais que proporcionam a criação de aplicações interativas, para auxiliar alunos e professores interessados, podendo ser acessado por meio de um *browser* sem a necessidade de instalação de *plugins.*

Uma das possíveis continuidades deste trabalho seria o acréscimo de uma análise estatística, para que o próprio sistema realize comparações entre os resultados obtidos de cada simulação, por meio de uma visualização mais robusta.

Sendo a disciplina de Sistemas Operacionais muito extensa, podem ser desenvolvidas simulações para outros conceitos da disciplina. Além disso, a abordagem proposta também pode ser estendida a outras disciplinas da computação, tendo em vista a facilidade no desenvolvimento de aplicações para Internet com as tecnologias utilizadas e a melhor compreensão dos conceitos abordados.

6. Referências

Chernich, R. (1994). The design and construction of a simulated operating system. In Proceedings of the Asia-Pacific Information Technology in Teaching and Education Conference, p. 1033–1038, Brisbane, Australia. Disponível em: http://davidtjones.wordpress.com/publications/the-design-and-construction-of-a-simulated-operating-system/. Acesso em 3 out. 2010.

Downey, A. B. (1999). “Teaching experimental design in an operating systems class”. Proceedings of the 30th ACM SIGCSE.

Google *Code*. (2010) Google *Web Toolkit*. Disponível em: http://code.google.com/intl/pt-BR/webtoolkit/. Acessado em: 3 nov. 2010.

Isotani, Shigueo; Jorge, C.H.; Quitério Júnio, N.M.; Silva, F. S.; Isotani, Seiji. (2009). Uma Ferramenta de Apoio à Aprendizagem de Sistemas Operacionais. Wei. Disponivel em <http://csbc2009.inf.ufrgs.br/anais/pdf/wei/st02_05.pdf>. Acesso em 22 mar. 2012.

Junior, M. L. A. (2007). AJAX em Java com o Google Web Toolkit. Disponível em: http://www.guj.com.br/article.show.logic?id=189. Acessado em: 4 nov. 2010.

Loiola, Rita (2009). Geração Y – Revista Galileu – Edição 219. Disponível em: http://revistagalileu.globo.com/Revista/Galileu/0,,EDG87165-7943-219,00-GERACAO+Y.html. Acesso em: 5 nov. 2010.

Machado, F.B., Maia, L.P. (2004). Um Framework Construtivista no Aprendizado de Sistemas Operacionais - Uma Proposta Pedagógica com o uso do Simulador SOsim. XII WEI, XXIV Congresso da SBC, Salvador, BA.

Maia L. P. e Pacheco Jr, A. C. (2003). A simulator supporting lectures on operating systems. In FIE'03: 33th Annual Conference on Frontiers in Education, p. 13-17, Vol.2.

Maziero, C. A. (2002). Reflexões sobre o ensino prático de Sistemas Operacionais. Anais do X Workshop sobre Educação em Computação (WEI2002), XXII Congresso da SBC.

Piaget, J. (1932). The Moral Judgement of the Child. The Free Press, Glencoe, Illinois. Disponível em: http://www.archive.org/details/moraljudgmentoft005613mbp. Acesso em: 2 out. 2010.

Rocha, A. R., Schneider, A., Alves, J. C., Silva, R. M. A. (2004) "wxProc: Um Simulador de Políticas de Escalonamento Multiplataforma", INFOCOMP Journal of Computer Science, Volume 3 (1) 43- 47. Disponível em <http://www.dcc.ufla.br/infocomp/artigos/v3.1/art08.pdf>. Acesso em 22 mar. 2011.

SBC, Sociedade Brasileira de Computação (2003). Currículo de Referência da SBC para Cursos de Graduação em Computação e Informática. Disponível em: http://www.sbc.org.br/educacao. Acesso em: 20 set. 2010.

Silberschatz, A., Galvin, P. B., Gagne, G. (2008). Sistemas Operacionais com Java. Tradução de Daniel Vieira. 7ª edição. Rio de Janeiro: Campus.

Soares, C. D., Filgueiras, D. S., Silva, F. F. (2009) Google Web Toolkit. Disponível em:http://kenai.com/projects/projetofabioclerio/sources/projfabioclerio/content/Artigo%20Final%20GWT.pdf?rev=22. Acessado em: 3 nov. 2010.

Stasko, J. (1997). Using student built algorithm animations as learning aids. In Proceedings of the 28th Technical Symposium on Computer Science Education, pages 25–29. Disponível em: http://portal.acm.org/citation.cfm?id=268091. Acesso em: 4 out. 2010.

Whithers, J. and Bilodeau, M. (1992). An examination of operating systems laboratory techniques. ACM SIGCSE *Bulletin,* 24(3):60-64. Disponível em: http://portal.acm.org/citation.cfm?id=142083. Acesso em: 6 out. 2010.