Título do Artigo

|  |
| --- |
| **Autor 1**  [Autor1@meu-email](mailto:Author1@mymail) |

|  |  |
| --- | --- |
| Resumo | Aqui aparece o resumo em Português. Este resumo deve ter no máximo 300 palavras e deve ser escrito em itálico, Times 10. Se desejar, use o estilo Abstract do menu estilo.  **Palavras-Chave:** O resumo deve ser seguido de no máximo 6 palavras-chaves |

|  |  |
| --- | --- |
|  |  |

1. Seção 1

Este é um exemplo de seção, que deve ser formatado utilizando a fonte Times, tamanho 10. Referências no texto deverão ser feitas utilizando números [2,3].

Figuras e tabelas devem ser chamadas pela sua numeração dentro do texto, e não como “a figura abaixo”. Veja a Tabela 1 como exemplo de tabela. Veja a Figura 1 como exemplo de figura contendo código. Código dentro de uma figura tem que ser digitado usando a fonte **Courier New** tamanho 11.

Figuras podem ocupar uma coluna (vide Figure 3) ou as duas (vide Figura 2). Você não precisa colocar código dentro das figuras – observe o seguinte

**se** *n* = 0*x* := 1

n > 0(**val** *n*  {0  *n* < *N*} *x*:[true,*x* = *n*!]) (*n –* 1);

*x* := *x**n*

**fim-se**

A seção REFERÊNCIAS ao final do artigo apresenta o template genérico para referências. O formato das referências é descrito na seção 4.

1.1 Estilos

Títulos de subseções devem ser escritos em Times 12. Enumerações devem seguir o exemplo:

* Como este indicado aqui
* E assim por diante.

Como outra alternativa, você pode enumerar itens

1. deste modo
2. que pode ajudar em referências seguidas.

A edição de fórmulas costuma ser problemática. NÃO DEFINA AS SUAS PRÓPRIAS FONTES! A maioria das fórmulas pode ser escrita usando somente as fontes **Times** e **Symbol**.

Um exemplo de texto contendo fórmulas e símbolos é a expressão (**res** ** *p*)(**) = |[**var** *l*  *p*[**\*l*] ; ** := *l*]|. Entretanto, repare que por serem fontes especiais, as fontes padrão Times e Symbol não podem ser usadas. Quanto mais você usar fontes especiais, mais complicado fica o processo de edição eletrônica.

2. Seção 2

Por outro lado, as linguagens concorrentes orientadas a objeto têm uma boa capacidade de representação de soluções para problemas que envolvam concorrência, devido ao seu alto poder de abstração e da possibilidade de reuso de código.[9]

A implantação de um sistema de controle como um programa de objetos concorrentes é baseado em modelos de concorrência, o que permite a representação do sistema como uma rede de elementos cooperativos, neste caso os Agentes. O uso da mesma classe base para a derivação de todas as classes garante a uniformidade da interface para todos os elementos, facilitando a programação e minimizando erros de implementação.

*// include the definition of base classes*

#include <**Agent.h**>

*// definition of the agent classes*

class**SUltraSound***:* public**PSensor**{

...

};

class**BRunAround***:* public**Behavior**{

...

};

*// implementation of the agent classes*

*.... /\* methods of* **SUltraSound** *class \*/*

*.... /\* methods of* **BRunAround** *class \*/*

*// main program*

**BRunAround****runaround**(...);

**SUltrasound****su1**(...),**su2**(...);

**MainAgent***::***Main**( ) {

...  */\* startup code \*/*

}

**Figura 1 :** Exemplo de como apresentar código dentro de uma figura – para centrar a legenda use o estilo **figuraesq**

O objeto Agente apresentado aqui descende dos atores definidos em [5]. No modelo de ator, um ator é um agente computacional que executa ações em resposta a mensagens recebidas. Ao receber uma mensagem, um ator pode executar ações tais como enviar mensagens a outros atores, criar outros atores ou executar um script. O Agente trabalha da mesma maneira, processando uma mensagem antes de aceitar outra..

Um Agente pode ser representado de forma simplificada pelos seus principais componentes. Conforme mostrado na Figura 5, este consiste em:

* um buffer de mensagens, usado para armazenar as mensagens que estão chegando;
* um buffer de entrada, onde a mensagem que está sendo processada é guardada;

A desativação de um Agente também pode ocorrer via uma mensagem de DESATIVAÇÃO. Neste caso todos os Agentes na lista de destinatários recebem uma mensagem DESATIVADORA informando que este Agente foi desativado. Estes agentes devem escolher entre se desativarem (a ação default) ou proceder sem a informação enviada por este Agente. Após enviar a mensagem de DESATIVAÇÃO o agente se desativa chamando o método End.

Há duas formas de mediação e arbitragem usadas neste framework, ambos usando o mecanismo definido acima:

* o mecanismo *conquer*; e
* o mecanismo *channel*.



**Figure 2:** Exemplo de figura ocupando 2 COLUNAS

O mecanismo *conquer* [7] funciona como um importante mecanismo de arbitragem para Agentes Comportamentais, formando um esquema onde

|  |  |
| --- | --- |
| **Função** | **Descrição** |
| OnPrimaryUnblock | Caso quem chamou não seja uma replica primária será bloqueada. Caso vire primária será desbloqueada. |
| Checkpoint | Usado por replicas primárias para gerar checkpoints. |
| GetCheckpoint | Retorna checkpoint. |
| WhatsMyRole | Retorna o papel de quem chamou. |

**Tabela 1:** Exemplo de tabela – o estilo do título é **Table Titulo**; o estilo do conteúdo é **Table text**

  
  
**Figura 3:** Exemplo de figura ocupando UMA COLUNA – legenda com estilo **figura**

3 Seção 3

Este framework foi desenvolvido originalmente numa estação SUN. Depois num PC rodando Linux usando um cross-compiler baseado no GNU C++, e gerando código para uma CPU Motorola 68XXX. Ele foi testado num sistema VME usando uma CPU Motorola 68030, e posteriormente numa CPU Motorola 68332. Com exceção das rotinas de troca de tarefas implementadas em Assembly, e de rotinas dependentes de hardware para interfaceamento com canais seriais, implementadas em C, todo o software foi implementado em C++, e roda em qualquer CPU de 32 bits.

4 Como referenciar

Referências devem ser completas, de acordo com as seguintes regras:

A Revista não permite referências a artigos não publicados (por exemplo, submetido para publicação, ou comunicação pessoal). Referências bibliográficas devem ser numeradas e chamadas no texto dentro de colchetes.

Formatos de referências:

1. Artigos em revistas/anais: Autores (iniciais seguidas pelo sobrenome). Título do artigo; título da revista ou dos anais da conferência, volume, número, páginas e ano

Exemplo (revista): X. B. Young, TítuloMeuArtigo. *Título da Revista* (em itálico), 3(2):10-20, 1995.

Exemplo (anais): X. B. Young. TítuloMeuArtigo. In *Anais do xxx* (em itálico), Rio de Janeiro, páginas 25-38, 1995

1. Artigos em Livro (ou Livro): Autores, título, editora e local, ano. (Caso seja capítulo de livro, título ou número do capítulo, título do livro, numeração das páginas)

Exemplo: X. B. Young. TítuloDoMeuCapítulo(ou do Livro). Editora, Rio de Janeiro, 1995.

1. Teses, Dissertações e Relatórios Técnicos: Autores (iniciais seguidas pelo sobrenome), título, número-do-relatório, instituição, mês e ano.

Exemplo (Tese) X. B. Young. TítuloDaMinhaTese. Tese de Doutorado, Universidade, Jan 1995.

Exemplo (Relatório) X. B. Young. TítuloDoMeuRelatório. Relatório Técnico RTnn, Universidade, Jan 1995.

1. WWW (referenciando páginas): Autor, título, endereço www completo, data do acesso. (referências às páginas WWW devem ser evitadas sempre que possível).   
   Exemplo: X. B. Young. MeuWWWTítuloDoArtigo. http://www.enderêço.completo, Jan. 1998

Referências

|  |  |
| --- | --- |
| **[1]** | G. R. Andrews, F. B. Schneider. Concepts and Notations for Concurrent Programming. *Computing Surveys*. 15(2):3-43, 1983. |
| **[2]** | R. A. Brooks. Autonomous Mobile Robots. In W. E. L. Grimson, , R. S. Paul (eds.) *AI In The 1990's And Beyond*, páginas 343-363, MIT Press, Massachusetts. 1987. |
| **[3]** | R. A. Brooks. A Robust Layered Control System for a Mobile Robot. AI Memo 864. Massachusetts Institute of Technology, 1985. |
| **[4]** | E. Gat. ALFA: A Language for Programming Reactive Control Systems. *In Proceedings of the IEEE International Conference on Robotics and Automation*. páginas 1116-1121, 1991. |
| **[5]** | C. Hewitt. Viewing Control Structures as Patterns of Passing Messages. *Artificial Intelligence*. 8: 323-364, 1977. |
| **[6]** | S. Matsuoka, A. Yonezawa. Analysis of Inheritance Anomaly in Object-Oriented Concurrent Programming Languages. In *Research Directions in Concurrent Object Oriented Programming*. MIT Press. páginas 107-150, 1993. |
| **[7]** | H. A. Schneebeli. Die Steuerung von Mehrfinger-Greifersystemen. Universidade de Karlsruhe, Alemanha, Março 1992. |
| **[8]** | J. E. M. Xavier. Uma Estrutura para a Construção de Sistemas de Controle Baseados em Agentes para Robôs Móveis. Dissertação de Mestrado, Universidade Federal do Espírito Santo, Novembro 1996. |
| **[9]** | A. Yonezawa, E. Shibayama, T. Takada et al. Modeling and Programming in an Object-Oriented Concurrent Language. In A. Yonezawa, M. Tokoro, (eds*.) Object-Oriented Concurrent Programming*. MIT Press. páginas 55-90, 1991. |