Controle de Célula de Produção de Tempo Real com DMIs*

Leandro Azevedo Cassol, Avelino Francisco Zorzo
{cassol, zorzo}@inf.pucrs.br

FACIN - PUCRS - Av. Ipiranga, 6681 - 90619-900 - Porto Alegre - RS

Abstract. This paper presents the design of a controlling system for a production cell. The design uses an abstraction called Dependable Multiparty Interaction (DMI), which is used to enclose all interactions between the devices of the production cell. This paper also presents a short description of the production cell case study. The goal of the paper is to show that real time and fault tolerance requirements can be satisfied in a controlling software implemented with DMIs.

Keywords: Distributed Systems, Dependable Multiparty Interaction, Real Time, Fault Tolerance

1 Introdução

O aumento do uso dos computadores em quase todos os aspectos da vida moderna tem conduzido a uma necessidade de elevar a confiança dos sistemas de computadores e dos próprios computadores. Existem muitas áreas onde os computadores desempenham tarefas críticas. Um exemplo é a área de tempo real. Nesta área, uma falha nos computadores pode ocasionar resultados catastróficos, pois os resultados devem estar corretos não somente do ponto de vista lógico, mas também devem ser gerados no momento correto.

Alguns sistemas críticos e de tempo real envolvem atividades concorrentes complexas. Em alguns casos, estas atividades concorrentes podem trabalhar juntas, cooperando, para resolver um determinado problema. Em outros casos, as atividades podem ser completamente independentes ou podem ser essencialmente independentes apesar de necessitar concorrer para compartilhar recursos comuns do sistema. Na prática, diferentes espécies de concorrência podem coexistir em uma aplicação complexa, que irá necessitar de um mecanismo de suporte geral para controlar e coordenar as atividades concorrentes complexas.

Neste trabalho, mostra-se como projetar e implementar aplicações críticas de tempo real usando uma abstração de controle geral, chamada Interações Multi-Participantes Confiáveis [1]. Como estudo de caso, foi usado um modelo de uma célula de produção que descreve um problema real da indústria.

2 A Célula de Produção FZI

O modelo da Célula de Produção, usado nesta seção, foi desenvolvido pelo *Forschungszentrum Informatik* (FZI), Karlsruhe, Alemanha, como um estudo de caso que apresenta propriedades de tempo real [2]. Esta célula de produção é composta de duas esteiras transportadoras (esteira alimentadora e esteira depósito), um leitor de código de barras, quatro unidades de processamento (UP) e dois guindastes (ver Figura 1). A Célula de Produção FZI utiliza

* Pesquisa financiada pela HP-Brasil (Convênio CPAD/FACIN/HP)

um conjunto de atuadores e sensores. Os atuadores podem ser usados para alterar o estado do sistema, como ligar e desligar a esteira alimentadora, mover os guindastes, entre outros. Já os sensores fornecem informações sobre o estado do sistema, e.g. se há um bloco no final da esteira alimentadora ou quais são as posições dos guindastes.

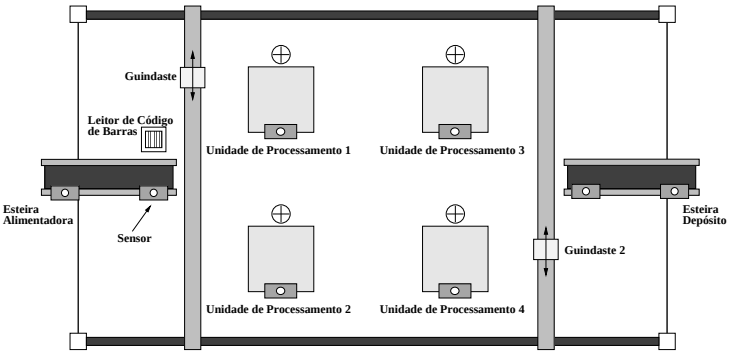

Figura 1. Estrutura da Célula de Produção FZI

Processamento dos Blocos. Os blocos entram no sistema através da esteira alimentadora e são transportados por ela até o sensor que está localizado no final dela detectar a presença de um bloco, ocasionando a parada da esteira. Nesse momento, o leitor de código de barras transmite as informações para o sistema de controle depois de ter lido de cada bloco.

Cada bloco possui um código de barras que contem informações para o seu processamento: *i)* quantas UP (no mínimo em uma e no máximo em quatro), quais e os tempos, mínimo e máximo, que são necessários para o processamento do bloco; *ii)* o tempo máximo que o bloco pode gastar em todo o sistema; e *iii)* se deve respeitar a ordem de processamento que foi informada no código de barras, pois um bloco pode ser processado sem respeitar essa ordem.

O controlador da célula deve garantir que os guindastes levem os blocos para a correta UP. Um bloco somente pode ser colocado em uma UP quando ela estiver desocupada. Existem dois tipos de UP: *i)* prensa: é ligada pelo controlador da célula e será desligada automaticamente quando ela terminar o processamento; e *ii)* forno: está sempre ligado. Um bloco é processado durante o tempo que permanecer nesse dispositivo. Depois de os blocos terem passado pelas UP, eles devem ser colocados na esteira depósito.

Propriedades. As propriedades do sistema estão organizadas em três classes: *safety properties*, *liveness properties* e *correctness properties*. Se todas as *safety properties* são satisfeitas, então nenhum dispositivo será danificado. Evitar colisões dos guindastes, evitar colisões entre os blocos e garantir que os blocos fiquem em áreas seguras são classificadas como *safety properties*. As *liveness properties* asseguram a ausência de *deadlocks* no sistema, isto é, todos os blocos são introduzidos no sistema pela esteira alimentadora e irão sair do sistema pela esteira depósito. Já as *correctness properties* asseguram que todas as informações lidas pelo controlador a partir do código de barras serão respeitadas [2].

3 Interações Multi-Participantes Confiáveis

Um mecanismo que abriga diversos processos executando um grupo de atividades em conjunto é chamado de interação multi-participantes (*multiparty interactions*) [3,4]. Em uma in-

teração multi-participantes, diversos processos (*threads*, objetos) de alguma forma "se reúnem" para produzir resultados combinados e intermediários, usam este estado para executar atividades em conjunto, e então abandonam a interação e continuam suas execuções normais. Mecanismos existentes para interações entre diversos participantes não fornecem recursos para tratar possíveis falhas que podem ocorrer durante a execução da interação. Em alguns, o sistema simplesmente pára como resposta a uma falha. Isto não é aceitável em diversas situações.

Em [1], o tratamento de exceções é adicionado ao mecanismo de interação multi-participantes. Este novo mecanismo é chamado de Interações Multi-Participantes Confiáveis (*Dependable Multiparty Interaction* - DMI). Uma DMI é uma interação entre diversos participantes que fornece recursos para: *i)* Tratar Exceções Concorrentes [5,6]; e *ii)* Garantir Consistência na Saída. Discussões a respeito das características de uma DMI podem ser encontradas em [1].

4 O Controlador da Célula de Produção

Para a construção de um controlador para o simulador da Célula de Produção FZI foi desenvolvido um projeto que satisfaz os requisitos de segurança *(safety)* e de tempo real (*correctness*) do estudo de caso (ver Seção 2), assegurando a ausência de *deadlocks* no sistema (*liveness*). O projeto foi separado em um conjunto de DMIs que controla as interações entre os dispositivos; um conjunto de controladores dos dispositivos que executam as DMIs; e um escalonador de DMIs que determina a ordem na qual as DMIs são executadas. Os requisitos de segurança são satisfeitos no nível de DMIs, enquanto os outros requisitos são atendidos pelos controladores dos dispositivos e pelo escalonador de DMIs (requisitos de tempo real).

Dessa forma, o *software* controlador para toda a Célula de Produção FZI consiste simplesmente em um conjunto de DMIs, em controladores dos dispositivos e em um escalonador de DMIs. Este controlador foi implementado na linguagem de programação Java.

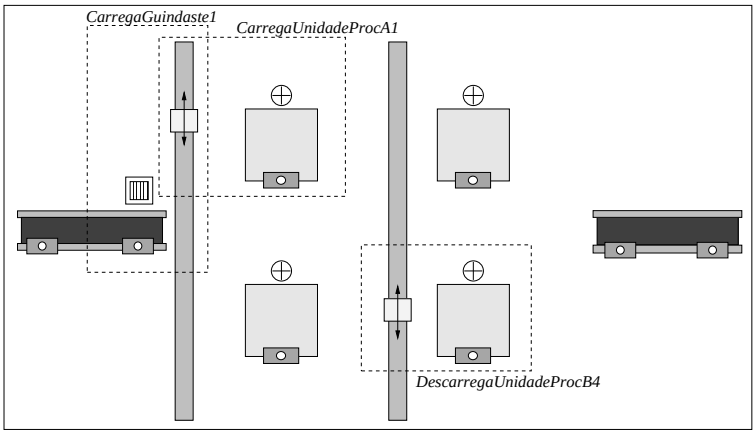

Figura 2. O conjunto de DMIs na Célula de Produção FZI

A Figura 2 mostra três das vinte e três DMIs que estão presentes na Célula de Produção. Cada DMI está representada por um retângulo pontilhado, a fim de salientar quais os dispositivos estão envolvidos na DMI. Devido aos possíveis processos de recuperação, duas DMIs, que possuem seus respectivos retângulos pontilhados sobrepostos, não podem ser executadas

concorrentemente, porque o mesmo dispositivo (ou bloco) não pode estar envolvido em mais do que uma DMI ao mesmo tempo [1].

Os movimentos de quase todos os dispositivos são desempenhados pelas DMIs e os dispositivos envolvidos na DMI são desligados antes do término dela. Assim, todos os dispositivos estão imóveis quando não estão sob o controle da DMI. O único dispositivo que não é controlado pelas DMIs é a esteira depósito, que é ativada pelo ambiente que engloba a célula.

4.1 Funcionamento

O conjunto de DMIs do controlador executa várias operações críticas, i.e. passagem de um bloco entre os dispositivos. A ativação destas DMIs e a ordem em que elas são executadas é a responsabilidade, respectivamente, dos controladores dos dispositivos e do escalonador de DMIs, que pode ser feito através de uma grande DMI. Os participantes neste outro nível de DMI, são ativados por *threads* de execuções externas, criadas imediatamente após o sistema iniciar sua execução.

Foram desenvolvidas vinte e três DMIs para controlar as interações entre os dispositivos: *CarregaCelula* (CC), *CarregaGuindaste1* (CG1), *RecuaGDT1* (RG1), *RecuaGDT2* (RG2), *CarregaUnidadeProcA*$_i$ (CUPA$_i$), *CarregaUnidadeProcB*$_i$ (CUPB$_i$), *DescarregaUnidProcA*$_i$ (DUPA$_i$), *Descarrega Unidade ProcB*$_i$ (DUPB$_i$), *ProcessaBloco* (PB), *CarregaEsteiraDeposito* (CED) e *DescarregaCelula* (DC). O índice i representa uma das quatro unidades de processamento, portanto existem quatro DMIs para cada DMI que possuir o índice i.

O ciclo de produção completo de um bloco na Célula de Produção FZI é o seguinte: um bloco entra no sistema da célula de produção através do ambiente. Este bloco é colocado no dispositivo esteira alimentadora pela DMI *CarregaCelula*. Após, a DMI *CarregaGuindaste1* é responsável por fazer o dispositivo leitor de código de barras ler as informações necessárias sobre o processamento do bloco. Essas informações são transmitidas, por esta DMI, para o dispositivo guindaste 1. A DMI *CarregaGuindaste1* também é responsável por fazer o dispositivo guindaste 1 agarrar o bloco no final da esteira alimentadora.

Quando o bloco estiver no dispositivo guindaste 1, a DMI *CarregaUnidadeProcA*$_i$ levará o bloco até a UP informada no código de barras do bloco. Após a chegada do bloco nesta UP, ele deverá ser processado. A DMI *ProcessaBloco* será responsável por processar o bloco. Terminando esta etapa, o *software* controlador poderá: *i)* fazer o dispositivo guindaste 1 pegar o bloco através da DMI *DescarregaUnidProcA*$_i$. Através desse caminho, o bloco deverá, mais tarde, ser colocado em outra UP, ocasionando um processo recursivo; ou *ii)* fazer o guindaste 2 pegar o bloco. Esta operação será realizada pela DMI *DescarregaUnidadeProcB*$_i$.

Quando o bloco estiver no dispositivo guindaste 2, ele poderá: *i)* voltar para uma UP através da DMI *CarregaUnidadeProcB*$_i$, para continuar o seu processamento; ou *ii)* ser colocado no dispositivo esteira depósito pela DMI *CarregaEsteiraDeposito.*

Após o bloco ser colocado na esteira depósito, a DMI *DescarregaBloco* será responsável por fazer o bloco sair da Célula de Produção. Neste ciclo de produção não foram citadas as DMIs *RecuaGDT1* e *RecuaGDT2*. Estas duas DMIs são utilizadas para os guindastes irem para áreas seguras. Estas são áreas onde somente um guindaste poderá chegar. Dessa forma, no momento em que um dos guindastes estiver executando a sua DMI de recuo, estará garantido que ele não poderá colidir com o outro guindaste.

A maioria das DMIs projetadas para o estudo de caso da Célula de Produção FZI, têm dois participantes: um que recebe o bloco como um argumento de entrada e o outro que retorna o bloco como um argumento de saída. O dispositivo que tem o bloco como um argumento de entrada, passa para o dispositivo que tem o bloco como um argumento de saída.

A DMI *CarregaGuindaste1* usa três objetos externos e é composta de três papéis[1]: *EsteiraAlimentadoraRole*, *LeitorCodigoBarrasRole* e *Guindaste1Role*. Na Figura 3, a passagem de um bloco de um papel para outro é representada por uma seta sólida com uma direção. Como pode ser visto nessa figura, o papel *EsteiraAlimentadoraRole* recebe o objeto Bloco como um parâmetro de entrada. Depois, é realizada a passagem física do Bloco da esteira alimentadora para o leitor de código de barras, que neste momento faz a leitura do código de barras e depois envia esse objeto para o papel *Guindaste1Role*, que representa o guindaste que está pegando fisicamente o objeto. A partir desse momento, o papel *Guindaste1Role* retorna esse objeto como um parâmetro de saída. Os acessos aos objetos externos bloco, esteira alimentadora, leitor de código de barras e guindaste 1 são representados por setas tracejadas. Já as atividades desempenhadas pelos papéis (*roles*) são representadas por quadrados pontilhados.

Durante a implementação do projeto da Célula de Produção FZI encontraram-se alguns problemas. Para resolver estes problemas foram tomadas as seguintes decisões: *i)* realizar o processamento dos blocos na Célula de Produção, sempre respeitando a ordem que foi informada no código de barras; e *ii)* para evitar a ocorrência de *deadlocks*, o número máximo de blocos que a célula de produção pode processar ao mesmo tempo foi estipulado em dois.

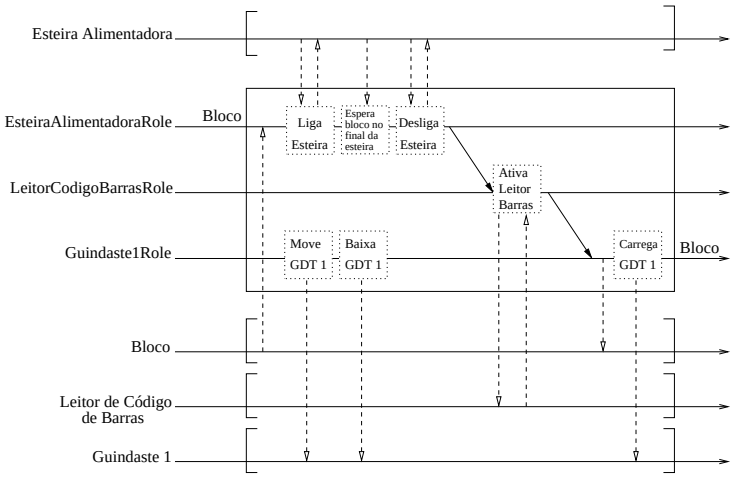

Figura 3. A DMI *CarregaGuindaste1*

4.2 Escalonador de DMIs

De forma a fazer com que as DMIs sejam executadas de maneira correta e que respeitem os requisitos de tempo real necessários para essa célula, foi necessária a implementação de um escalonador em vez de adicionar novas características de tempo real dentro de uma DMI.

O escalonador de DMIs é responsável por determinar a ordem em que as DMIs são executadas. Assim, os controladores dos dispositivos se comunicam com o escalonador de DMIs para agendar a ordem de execução das DMIs. Essa comunicação é realizada através da troca de mensagens entre eles, isto é, os controladores informam ao escalonador o estado dos dispositivos ou

[1] Cada papel é responsável por abrigar as instruções de cada participante de uma DMI.

algum pedido e ficam aguardando uma resposta do escalonador para executarem alguma DMI. Por sua vez, o escalonador recebe as mensagens dos controladores, verifica qual ação deve ser tomada e, após, envia mensagens para os controladores dos dispositivos que participarão da ação, a fim de que eles executem uma determinada DMI.

O estudo de caso apresenta dois tipos de requisitos de tempo. O primeiro refere-se ao tempo máximo que cada bloco pode gastar em toda a célula, já o segundo refere-se ao tempo mínimo e máximo necessários para o processamento de cada bloco nas UP. Para satisfazer o primeiro tipo de requisito, toda a vez que um guindaste estiver carregando um bloco, será verificado se o *deadline* desse bloco é menor que o *deadline* do outro bloco que está presente na célula de produção. Dessa forma, o bloco que possuir o menor *deadline* será processado primeiro. Já para satisfazer o segundo tipo de requisito, quando existirem dois blocos sendo processados ao mesmo tempo, os *deadlines* desses blocos são comparados para verificar qual bloco tem o menor *deadline*, ou seja, qual bloco deve ser retirado primeiro de uma UP.

4.3 Requisitos de Tempo

Os requisitos de tempo do estudo de caso foram satisfeitos através do escalonador de DMIs. Porém, para ter uma maior segurança quanto ao cumprimento desses requisitos, foram realizadas diversas medições de tempo de execuções das DMIs desenvolvidas para o estudo de caso. Para realizar essas medições foi utilizado um computador PC IBM Pentium Celerom, de 330 MHz, com 64 MB de memória RAM e tendo como sistema operacional o Linux Red Hat 7.0 (kernel 2.2.16-22). As medições dos tempos de execução das DMIs foram realizadas a partir de iterações compostas pelo processamento de quatro blocos. Os valores foram obtidos em duas etapas, sendo que em cada uma houve a repetição de cada iteração mil vezes, totalizando duas mil vezes o processamento de quatro blocos.

Como o processamento de cada bloco envolve a execução de algumas DMIs, após as duas etapas de medição dos tempos, obteve-se vários valores correspondente ao tempo de execução de cada DMI. Para cada DMI foram descartados os valores que estavam destoando em relação aos demais valores. Com base nesses novos valores, determinou-se que os maiores valores de cada DMI corresponderiam ao tempo máximo de execução de cada uma (ver Tabela 1).

DMIs	Tempo Máximo (seg.)
CG1	5
CUPA1 e CUPA2	5,8
CUPA3 e CUPA4	7,9
DUPA1 e DUPA2	3
DUPA3 e DUPA4	5,9
CUPB1 e CUPB2	5,1
CUPB3 e CUPB4	5,4
DUPB1 e DUPB2	7,7
DUPB3 e DUPB4	5,2
CED	7,6

Tabela 1. Tempo máximo de execução das DMIs

Unidades de Processamento	Tempo Mínimo (seg.)
1	26,1
2	26,1
3	25,7
4	25,7
(1,2) ou (2,1)	34,9 e 38,9
(1,3) ou (1,4)	34,5 e 36,7
(2,3) ou (2,4)	34,5 e 36,7
(3,1) ou (3,2)	39,9 e 38,5
(4,1) ou (4,2)	39,9 e 38,5
(3,4) ou (4,3)	39,5 e 36,3

Tabela 2. Tempo mínimo de processamento de um bloco

Com base nos tempos da Tabela 1, foi realizada a soma dos tempos de execução das DMIs que estão envolvidas no processamento de um bloco em uma e duas UP. Para cada UP foi obtido um valor. Todos os valores podem ser visualizados na Tabela 2.

Após a obtenção dos tempos da Tabela 1 determinou-se que os tempos mínimos necessários para realizar o processamento de cada bloco nas UP devem ser maiores que os valores apresentados na Tabela 2, pois se o *deadline* de um bloco for menor que esses tempos, a célula de produção poderá não cumprir o *deadline* em uma determinada UP. Um outro detalhe a ser acrescentado é que o tempo gasto para realizar o escalonamento foi desconsiderado por ser muito menor que o tempo gasto para executar uma DMI. Por exemplo, para executar o escalonamento leva-se microseg./miliseg. enquanto que o processamento mecânico de um bloco leva segundos ou até minutos.

Em algumas linhas da segunda coluna, da Tabela 2, têm-se dois valores. O primeiro valor foi obtido com o guindaste 1 fazendo a maior parte da operação na célula de produção, deixando só a retirada do bloco da célula de produção para o guindaste 2. Já o segundo valor que aparece nesta coluna foi obtido com o guindaste 2 fazendo a maior parte da operação, deixando somente a inclusão do bloco na célula de produção para o guindaste 1.

4.4 Tratamento de Exceções

O controlador da célula de produção irá levantar uma exceção toda a vez que o *deadline* de um bloco for menor que o tempo mínimo de processamento determinado na Tabela 2. Assim, quando um bloco irá ser processado, o escalonador verifica quanto tempo levará para processá-lo. Após, será levantada uma exceção se o tempo necessário para realizar o processamento do bloco for: *i)* menor que o tempo mínimo determinado no código de barras do bloco; *ii)* maior que o *deadline* do bloco na UP; e *iii)* maior que o *deadline* do bloco em todo o sistema.

Se uma exceção for levantada, o controlador irá interromper a execução do sistema. Nesse trabalho não foi realizado o tratamento das exceções levantadas durante a execução do controlador, pois o objetivo é mostrar que as DMIs podem ser usadas em sistemas de tempo real.

5 Conclusões

Em sistemas de tempo real existe uma dificuldade em compatibilizar dois objetivos fundamentais: garantir que os resultados sejam produzidos no momento desejado e dotar o sistema de flexibilidade para adaptar-se a um ambiente dinâmico e, assim, aumentar sua utilidade. Dessa forma, o uso de um escalonador deve assegurar que os *deadlines* das tarefas sejam cumpridos. Em alguns sistemas, mesmo em caso de falha de algum componente, as tarefas que nele estavam sendo executadas devem ser asseguradas. Para isso, o escalonador deve implementar tolerância a falhas. Um exemplo de um sistema que engloba estas duas características, tolerância a falhas e tempo real, é o estudo de caso da Célula de Produção FZI [2].

Para projetar e implementar o sistema de controle para a Célula de Produção FZI (estudo de caso) foi utilizado o mecanismo proposto em [1], DMI. Porém, o mecanismo de DMI não fornece, de maneira direta, as propriedades necessárias para poder ser usado em interações entre diversos participantes que possuem requisitos de tempo real.

Um método para atender os requisitos de tempo real pode ser a inclusão, nas próprias DMIs, dessas características. Este tipo de filosofia foi adotada em [7] e [8], onde foi estudado um método para incluir requisitos de tempo real em um mecanismo similar às DMIs, as *CA Actions* [9,10]. Uma outra forma de atender estes requisitos é a utilização de um escalonador de DMIs, ao invés de incluir novas propriedades nas DMIs. Dessa forma, é possível atender aos requisitos

de tempo real, para problemas similares aos da célula de produção, sem incluir estes requisitos nas DMIs. Parte desses resultados estão apresentados em [11] e [12].

As propriedades descritas na Seção 2 foram atendidas da seguinte maneira: *i)* as *safety properties* foram satisfeitas pelas DMIs; *ii)* as *liveness properties* foram atendidas pela maneira como foi projetado o sistema controlador e também pelo escalonador de DMIs; e *iii)* as *correctness properties* foram atendidas pelo escalonador de DMIs, por exemplo, o escalonador não deixa a DMI $DUPA_2$ retirar uma peça da UP 2 antes dela ter sido totalmente processada, ou ainda, a ordem com as quais as peças têm que passar pelas UP é respeitada pelo escalonador.

Neste trabalho, mostrou-se como usar as DMIs para projetar um sistema crítico: o da Célula de Produção FZI. Foi demonstrado que aplicando as DMIs para este estudo de caso ajudou-se a melhorar a construção do projeto para esse sistema, pois o uso de DMIs para implementar o sistema permite garantir todos os requisitos relacionados com as atividades concorrentes da Célula de Produção FZI [13]. Foi mostrado que a utilização de um mecanismo que pode confinar os erros e, conseqüentemente, fornecer tolerância a falhas também pode ser utilizado em sistemas de tempo real. Embora um projeto baseado em DMIs possa diminuir o desempenho de alguns sistemas, acredita-se que os benefícios ganhos por um projeto simples, usando componentes reutilizáveis e fornecendo ao sistema uma disciplina de tolerância a falhas, possibilitem construir aplicações críticas de tempo real com mais eficiência.

Referências

1. A. F. Zorzo. *Multiparty Interactions in Dependable Distributed Systems.* PhD thesis, University of Newcastle Upon Tyne, UK, 1999.
2. A. Lötzbeyer and R. Muhlfeld. Task Description of a Flexible Production Cell with Real Time Properties. Technical report, Forschungszentrum Informatik, Karsruhe, Germany - http://www.fzi.de/divisions/prost/projects/korsys/korsys.html, 1996.
3. M. Evangelist, N. Francez, and S Katz. Multiparty interactions for interprocess communication and synchronization. *IEEE Transactions on Software Engineering*, 15(11):1417–1426, 1989.
4. Y. J. Joung and S. A. Smolka. A comprehensive study of the complexity of multiparty interaction. *Journal of ACM*, 43(1):75–115, 1996.
5. R. H. Campbell and B. Randell. Error Recovery in Asynchronous Systems. *IEEE Transactions on Software Engineering*, 12(8):811–826, 1986.
6. A. Romanovsky, J. Xu, and B. Randell. Exception Handling and Resolution in Distributed Object-Oriented Systems. In *16th IEEE Int. Conf. on Distributed Computing Systems*, pages 545–552, Hong Kong, 1996. IEEE CS Press.
7. A. Burns, B. Randell, A. Romanovsky, R. Stroud, A. J. Wellings, and J. Xu. Temporal Constraints and Exception Handling in Object-Oriented Distributed Systems. *Design for Validation (DeVa) - Third Year Report, Esprit LTR Project 20072*, pages 3–25, Dezembro 1998.
8. A. Romanovsky, J. Xu, B. Randell, R. J. Stroud, and A. Burns. Analysis and Design of the Real-Time Production Cell. Technical report, Department of Computing Science (University of Newcastle Upon Tyne), UK, 1998.
9. B. Randell, A. Romanovsky, R. J. Stroud, J. Xu, and A. F. Zorzo. Coordinated Atomic Actions: from Concept to Implementation. Technical Report 595, Department of Computing Science (University of Newcastle Upon Tyne), UK, 1997.
10. J. Xu, B. Randell, A. Romanovsky, C. Rubira, R. J. Stroud, and Z. Wu. Fault Tolerance in Concurrent Object-Oriented Software through Coordinated Error Recovery. In *Proceedings of the 25th Int. Symp. on Fault-Tolerant Computing (FTCS-25)*, pages 450–457, Pasadena, USA, 1995. IEEE CS Press.
11. L. A. Cassol. Projeto de uma Célula de Produção com Requisitos de Tempo Real usando DMIs. In *Workshop de Teses e Dissertações em Computação Tolerante a Falhas (SCTF'01)*, pages 37–42, Florianópolis, SC, Brasil, Março 2001.
12. L. A. Cassol. Tempo Real em Interações Multi-Participantes Confiáveis. Master's thesis, Departamento de Informática (PUCRS), Porto Alegre, RS, Brasil, 2001.
13. A. F. Zorzo, A. Romanovsky, J. Xu, B. Randell, R. J. Stroud, and I. S. Welch. Using Coordinated Atomic Actions to Design Safety-Critical Systems: a Production Cell Case Study. *Software - Practice and Experience*, 29(8):677–697, 1999.