APOS: AMBIENTE para o PROJETO e OPERAÇÃO de SISTEMAS

Orlando G. Loques F., Julius C. B. Leite
Grupo de Sistemas de Computação
Dept. Eng. Elétrica, PUC-RJ, Cx. Postal 38063
CEP 22452, Rio de Janeiro, RJ

Sumário: O trabalho descreve atividades visando a criação de um ambiente para o projeto e suporte de operação de sistemas distribuídos de tempo real. Tres aspectos são considerados: (i) modularidade, (ii) segurança de funcionamento, e (iii) sistema operacional distribuído.

Introdução

Este projeto visa prover um ambiente de desenvolvimento e suporte projetado para a construção e operação de grandes sistemas distribuídos de computação, com longo ciclo de vida. Exemplos comuns deste tipo de sistema são encontrados em aplicações de tempo real, tais como: processamento de transações, e controle e monitoração de processos. Esta classe de aplicações possue uma série de requisitos, os quais tem que ser obrigatoriamente considerados em diversos níveis do projeto do sistema. De modo a suprir estes requisitos, tres tópicos são de interesse principal para nossa pesquisa.

(i) Modularidade

De maneira a alcançar este objetivo -- modularidade, foi adotada uma metodologia de projeto de software similar a proposta em [KRAM 83]. Associada a esta metodologia existe um ambiente de suporte projetado para a construção e operação de sistemas distribuídos de tempo real. A proposta inicial era baseada em extensões da linguagem Pascal [LOQU 85, LOQU 86a]. Na fase atual, a linguagem "C" está sendo utilizada, devido as suas facilidades para o desenvolvimento de sistemas [NEST 87, NIGR 87].

A metodologia adotada permite que um sistema seja composto por um conjunto de módulos, cada qual contendo variáveis próprias (locais) e encapsulando uma tarefa (ou processo). O uso de variáveis globais é restringido, os módulos comunicam-se com outros módulos através de uma interface definida por portas usadas para a troca de mensagens. A comunicação é usualmente feita por meio de transações síncronas do tipo pedido-resposta ("request-reply"), similares a chamadas remotas de procedimento [BIRR 84]. Contudo, uma transação assíncrona também é disponível, oferecendo flexibilidade para programção de situações especiais.

Os módulos de software podem ser desenvolvidos e testados independentemente da configuração de sistema aonde serão usados. Por exemplo, (a) a especificação de comunicações locais e remotas é transparente ao nível da programação; (b) portas são objetos locais visíveis apenas no interior dos módulos, assim evitando a interdependência entre módulos. Uma linguagem especial de configuração permite a especificação dos tipos de módulos (contexto), a partir do qual o sistema será construído; a declaração das instâncias de módulos que serão criadas no sistema; a interligação de suas portas de comunicação, bem como o mapeamento das instâncias às estações. Desta forma, módulos podem ser mantidos em uma biblioteca e re-utilizados em diversas aplicações. A estruturação hierárquica de sistemas também é suportada ao nível de configuração.

(ii) Segurança de funcionamento

A obtenção desta meta implica em assegurar a correção do projeto do sistema e na disponibilidade de técnicas de tolerância a falhas para uso durante a sua operação.

No nível de programação módulos são equivalentes a tipos globais, unicamente identificados, os quais podem ser instanciados, e tem suas operações especificadas através da exportação de interfaces bem definidas. Ao nível da configuração a consistência do interfaceamento de módulos é testada, assegurando que somente portas do mesmo tipo (de dados) sejam interligadas. Estas qualidades contribuem para a eliminação de erros de projeto (algoritmicos).

Na área de tolerância a falhas o nosso objetivo é integrar ao ambiente diversas técnicas que permitam o suprimento de diferentes requisitos de confiabilidade. Estamos experimentando técnicas, independentes da aplicação, as quais permitem a obtenção de maneira automática e transparente ao nível de programação, de redundância passiva e ativa de módulos [LOQU 86b,LOQU 87,LEIT 87a,b]. O tipo especifico de redundância a ser implementado é definido através de declarações no nível de configuração do sistema aplicativo. Cada um destes tipos de redundância possui características próprias, quanto ao tipo de falhas toleradas e resposta em tempo real oferecida. O suporte de ambos os tipos de redundância em um mesmo ambiente facilita a avaliação e comparação das duas técnicas. Além disto, permite que um projetista selecione a que melhor convier para uma determinada aplicação.

Em um âmbito mais geral, ferramentas para avaliação de confiabilidade e desempenho, deverão ser incluídas no ambiente. Uma nova possibilidade que se abre é o uso de técnicas de inteligência artificial para a diagnose de falhas e controle da reconfiguração de sistemas. Estamos também investigando o suporte de modelos orientados a objetos para o projeto de sistemas distribuídos, e.g., [STEM 86]. Isto permite uso de esquemas baseados em capacidades ("capabilities"), para o controle de acesso a objetos compartilhados, o que pode prover maior segurança de funcionamento aos sistemas aplicativos.

(iii) Sistema Operacional Distribuído

Sistemas distribuídos de tempo real apresentam particularidades, as quais levantam problemas interessantes e de difícil resolução, associados a sua arquitetura e engenharia. Neste sentido estamos usando a arquitetura básica, a qual provê o suporte de operação para os sistemas aplicativos, como um veículo para experimentos e testes. Esta atividade, visa a implementação de sistemas operacionais otimisados para o suporte de sistemas aplicativos requerendo alto desempenho. Dentre os tópicos em consideração se incluem:

- Sistema de Comunicação, [SOUZ 87].
- Sistema Distribuído de Arquivos, [RIHL 87].
- Gerenciamento Dinâmico de Configuração, [NIGR 87].

Convém ressaltar, que o sistema operacional em desenvolvimento suporta a capacidade de reconfiguração dinâmica. Isto facilita a realização de: (i) mudanças evolutivas, visando a incorporação de novas funcionalidades e tecnologias; (ii) mudanças operacionais, necessárias para o redimensionamento do sistema, atividades de manutenção, ou reorganização para a recuperação de falhas.

Na fase inicial do projeto, seu desenvolvimento era centrado em torno de um ambiente MS-DOS. Isto possibilitava o uso de módulos escritos em diferentes linguagens suportadas por este sistema operacional, que por várias razões esta disponível para nosso trabalho. Correntemente, estamos evoluindo para um ambiente de desenvolvimento centrado em torno de um sistema compatível com o Unix. Isto permitirá que nossas atividades sejam realizadas sobre uma base mais firme e ampla que o MS-DOS. Por exemplo, não será necessário desenvolver um servidor de arquivos para um ambiente concorrente, desde que um já bastante completo é disponível no sistema Unix; será suficiente estende-lo para o caso distribuído. Além disto, diversas ferramentas para o desenvolvimento de software existentes no ambiente Unix também poderão ser utilizadas. Inicialmente deveremos ter o sistema operacional alvo e o Unix convivendo, o que é bastante conveniente para a criação de um ambiente de desenvolvimento. Em uma etapa futura, contudo, desejamos obter um núcleo de suporte voltado para a metologia adotada, bem como para aplicações de tempo real. Por sua menor exigência de recursos, esta opção seria a mais indicada para prover o suporte de operação em estações controladoras baseadas em microcomputadores de menor porte.

Referências Relacionadas

[BIRR 84] Birrell, A.D., and Nelson, B.J., Implementing Remote Procedure Calls, ACM Trans. on Computer Systems, Vol. 2, N. 1, fevereiro 1984, p. 39-59.

[KRAM 83] Kramer, J., et al., Conic: An Integrated Approach to Distributed Control Systems, IEE Proc., Vol. 130, Pt. E, N. 1, janeiro 1983, p. 1-10.

[LEIT 87a] J. C. B. Leite, O. G. Loques F., L. B. de Souza, Gemini: A Modular and Reliable Distributed System, 30th Midwest Symposium on Circuits and Systems, Syracuse, New York, agosto de 87.

LEIT 87b] J.C.B. Leite, O.G. Loques Filho, e L.B. de Souza, Modularidade e Confiabilidade: Uma Experiência em Sistemas Distribuídos, 2o. Simpósio em Sistemas de Computadores Tolerantes a Falhas, Campinas, agosto de 1987.

[LOQU 85] Loques Filho, O.G., Souza, V.S., Uma Arquitetura de Software para Sistemas Distribuídos de Computação, 5o. Simp. sobre Desenvolvimento de Software Básico, UFMG, Belo Horizonte, Minas Gerais, novembro 1985.

[LOQU 86a] Loques Filho, O.G., Um Esquema de Software para a Construção de Sistemas Distribuídos de Tempo Real, VI Congresso da Sociedade Brasileira de Computação, XIII SEMISH, Recife, Pernambuco, julho 86.

[LOQU 86b] Loques Filho, O., Kramer, J., Flexible Fault-Tolerance in Distributed Systems, IEE Proceedings, V.133, Pt.E, N.6, novembro 1986, p. 319-331.

[LOQU 87] Loques Filho, O., Kramer, J., Flexible and Transparent Module Replication in Distributed Systems, TENCON 87, IEEE Region 10 Conference and Exhibition, Seoul, Korea, agosto de 1987.

[NEST 87] Nestorov, A., Núcleo de Suporte Local, com Capacidade de Configuração Dinâmica, para Sistemas Distribuídos de Tempo Real, Tese de Mestrado, DEE-PUC/RJ, finalização prevista para novembro de 1987.

[NIGR 87] Nigri, M.A., Gerenciamento de Configuração de Sistemas Distribuídos de Tempo Real, Tese de Mestrado, DEE-PUC/RJ, finalização prevista para novembro de 1987.

[RIHL 87] RIHL, A. P., Extensão de um Sistema de Arquivos para um Ambiente Distribuído, Tese de Mestrado, DEE-PUC/RJ, finalização prevista para março de 1988.

[SOUZ 87] Souza, L.B., Leite, J.C.B., Loques, O.G., O Protocolo de Acesso da Rede Gemini, XIV SEMISH, Salvador, julho de 87.

[STEM 86] Stemple, D. W., et al, Functional Addressing in Gutemberg: Interprocess Communication without Process Identifiers, IEEE Transactions on Software Engineering, V.12, No. 11, novembro 1986, p. 1056-1075.