Uma Abordagem Para Desenvolvimento de Software de Sistemas de Tempo Real Crítico em Arquiteturas Distribuídas

Paulo Roberto Pierri Tepedino
Instituto de Pesquisas Meteorológicas / Unesp
e:mail - Pierri@Ipmet1.Uesp.Ansp.Br

Tereza Gonçalves Kirner
Universidade Federal de São Carlos
Departamento de Computação

RESUMO

Esse artigo apresenta uma abordagem integrada para o desenvolvimento de software de sistemas de tempo real crítico em arquiteturas distribuídas. A abordagem abrange a especificação e o projeto deste tipo de software e seu principal objetivo é procurar garantir um dos requisitos fundamentais desses sistemas, que é a previsibilidade.

É empregada uma modelagem baseada em grafo, em conjunto com escalonamento Pre-Run-Time ótimo, para especificar, projetar e verificar e validar a computação necessária. Isto confere à abordagem características importantes, tais como, eficiência, eficácia e a possibilidade de um refinamento sucessivo, através de iterações entre as suas fases.

ABSTRACT

This paper presents an integrated approach for the development of Distributed Hard Real-Time Systems Software. It covers the specification and the design and its foremost goal is to achieve predictability, a fundamental requirement in this systems.

It is employed a graph-based model together with a Pre-Run-Time optimal scheduling to specify, design and verify and validate the necessary computation. This confers important features to the approach. It becomes effective, efficient and there is the possibility of a successive refinement of the design through iteractions between its phases.

1 Introdução

Os sistemas de tempo real crítico "Hard Real-Time Systems - HRTS" são de importância crucial para a sociedade contemporânea. A maioria dos HRTS são sistemas de segurança crítica ([1] e [19]), para os quais falhas no sistema computacional de tempo real podem conduzir à perda de vidas humanas e catástrofes econômicas e ecológicas ([4], [6] e [9]) . O sistema de software é o principal componente do HRTS, pois deve implementar e assegurar praticamente todas as funções críticas de controle requeridas pela aplicação.

Na computação de tempo real crítico, a exatidão do sistema depende do resultado lógico e do tempo em que a computação é realizada. Isso define um requisito básico de qualidade de HRTS, que é a previsibilidade ([15]). Devido às características intrínsecas dos sistemas distribuídos, às restrições de temporização e aos sincronismos entre os programas a construção desses sistemas não é trivial.

Estudando-se as técnicas de especificação de sistemas de tempo real apresentadas na literatura ([2], [3] e [18]), percebe-se que estas técnicas não são claras, quanto a definição de restrições de

temporização exigidas pela aplicação. Isto dificulta o emprego de tais técnicas para o desenvolvimento de HRTS. Além disto, não há uma demonstração clara de como estas técnicas de especificação são utilizadas nas fases seguintes do desenvolvimento do software, principalmente quando se prevê a implementação em uma arquitetura distribuída.

Para tentar garantir a previsibilidade, os paradigmas tradicionais de desenvolvimento valem-se de testes e/ou simulações exaustivas para minimizar a presença de faltas de temporização. Tais abordagens apresentam custos altíssimos e não são capazes de garantir que o comportamento do software será previsível ([1], [5], [7] e [19]).

Em contraposição às técnicas e metodologias tradicionais, este artigo descreve uma abordagem para o desenvolvimento de software distribuído HRTS, que emprega um modelo baseado em grafo e escalonamento Pre-Run-Time. Devido ao emprego destes elementos, existe a possibilidade da realização de um refinamento rigoroso das especificações e de uma verificação e validação automática da computação distribuída projetada. Estes procedimentos se constituem em um nível médio/alto no emprego de métodos formais ([1] e [17]), que reduzem a ocorrência de erros no desenvolvimento do software.

O Tópico 2 apresenta os fundamentos teóricos e no Tópico 3 se descreve detalhadamente a abordagem. Ênfase especial é dada às fases de especificação e, principalmente, projeto, em razão de sua importância e complexidade. A fase de implementação é discutida sucintamente. As conclusões são apresentadas no Tópico 4 e, a seguir, é listada a bibliografia que subsidiou o trabalho.

2 Fundamentação da Abordagem

Para maior clareza na descrição dos fundamentos da abordagem, que está baseada primariamente no paradigma clássico de ciclo de vida de engenharia de software ([11]), apresenta-se em tópicos seus princípios básicos, objetivos e os conceitos adotados.

2.1 Objetivos e Princípios Básicos da Abordagem

A abordagem visa a construção de HRTS distribuídos, previsíveis, grandes e complexos, onde o fator de utilização dos recursos computacionais não é baixo e existe um número relativamente grande de programas com restrições temporais justas e que possuem diferentes características de temporização e de interdependência.

Além disto, trata especificamente da computação crítica de tempo real e contempla sistemas distribuídos compostos por nodos uniprocessados e multiprocessados, interconectados por um sub-sistema de comunicação de topologia arbitrária. Os elementos processadores poderão possuir diferentes características de desempenho, porém a unidade de tempo[1] de cada um deles deverá ser idêntica. O sub-sistema de comunicação poderá apresentar canais de comunicação com diferentes desempenhos e também a sua temporização será determinada em unidades de tempo idênticas às utilizadas para os elementos processadores[2].

A eficácia da abordagem é estabelecida pela sua capacidade de determinar garantias de um escalonamento viável para os programas projetados em um sistema distribuído atendendo às restrições críticas de temporização e aos sincronismos impostos aos programas do sistema, de uma maneira global. Isto conduz a um comportamento temporal previsível da aplicação.

---

1 Elementos processadores são "Self-Timed" ([4]), isto é, temporizados por si próprios, evoluindo em intervalos discretos de tempo. Sempre há um mínimo intervalo de tempo no qual o elemento processador não é passível de interrupção ou há um certo conjunto desses intervalos que caracterizam um "quantum". A noção de unidade de tempo de elemento processador está relacionada com estes conceitos.

2 Em razão da necessidade de compartilhar recursos computacionais escassos, utiliza-se a metodologia de projeto de compartilhamento do elemento processador ([7]).

A eficiência da abordagem está relacionada com o emprego de uma linguagem de fácil utilização pelos projetistas na especificação do sistema (restrições de tempo) e com a possibilidade de um refinamento rigoroso desses requisitos especificados em um conjunto racionalmente organizado de programas distribuídos. Desta forma, garante-se uma maior facilidade e segurança para a especificação do sistema, uma utilização eficiente dos recursos computacionais escassos e elimina-se a ocorrência de erros na transformação dos requisitos computacionais especificados em um conjunto de programas que realizem a computação necessária.

A integração entre as fases da abordagem é determinada pela utilização de um modelo computacional uniforme através de todo o processo de desenvolvimento. A integração entre as fases de especificação, projeto e implementação é estritamente necessária para que ocorram iterações entre as diversas fases, de modo a possibilitar refinamentos sucessivos, alterações de especificações e de projeto, bem como viabilizar futuras manutenções ([12]).

2.2 Conceitos Adotados

Os principais problemas para o projeto de software distribuído HRTS são as questões de decomposição modular do software, a integração de sub-sistemas projetados por diferentes equipes, a alocação dos programas nos elementos processadores da arquitetura, a caracterização da carga imposta ao sub-sistema de comunicação e a caracterização dos diferentes desempenhos dos diversos recursos do sistema ([7] e [15]). Os conceitos adotados para tratar essas questões são apresentados a seguir.

a) Decomposição Modular de Software

Os problemas de complexidade, manutenibilidade e eficiência do projeto de software distribuído HRTS estão diretamente correlacionados com a sua decomposição modular. Essas questões são endereçadas pela abordagem através do modelo de grafo, principalmente pelo emprego dos grafos de especificação.

Cada uma das restrições de temporização exigidas pela aplicação definirá um grafo de especificação correspondente. O modelo de grafo permite a representação hierárquica da computação, assim, um nó do grafo de especificação poderá, inicialmente, representar todo um sub-sistema. Partindo-se de um nível alto de abstração para a especificação da computação necessária, definida por um pequeno número de nós do grafo de especificação, deverão ocorrer refinamentos sucessivos, possibilitados pela iteração entre as diferentes fases da abordagem, e, ao final do projeto do sistema, um nó definirá um módulo altamente coeso. Esse módulo será implementado fisicamente através de um programa[3], que poderá ser alocado a um elemento processador da arquitetura e ter garantias de um escalonamento viável.

Desta forma, há uma certa combinação dos métodos de "Decomposição por Restrições Temporais" e de "Maximização da Concorrência". Isso garante eficiência ao projeto e permite o controle da complexidade e da manutenibilidade ([7] e [16]). Um estudo mais detalhado destas questões de decomposição modular é apresentado por Tepedino & Kirner ([17]).

b) Integração de Sub-Sistemas HRTS

A integração de sub-sistemas projetados por diferentes equipes em um único sistema poderá se dar de duas maneiras. Caso o sub-sistema a ser integrado possua alguma computação com a mesma funcionalidade (nós idênticos) ou alguma relação de precedência com as computações do sistema já existente, nesse caso deverá ocorrer a união dos grafos de comunicação global. Caso contrário, o grafo de comunicação global desse novo sub-sistema deverá ser adicionado ao conjunto de grafos de comunicação global que representam o sistema.

---

3 A unidade básica de representação dos requisitos computacionais e de escalonamento é um nó do modelo de grafo. Esse nó poderá denotar, em um sentido genérico, um conjunto de computações. Ao final da fase de projeto, um nó definirá, de modo estrito, um único módulo coeso.

Caso haja a união de grafos de comunicação global de diferentes sub-sistemas, será necessária a reexecução das etapas de supressão de computações redundantes e de análise de viabilidade. Caso contrário, apenas a análise de viabilidade precisará ser reexecutada.

c) Alocação da Computação

A alocação da computação nos diversos elementos processadores do sistema é realizada pelos projetistas e pode consistir de duas etapas. Na primeira etapa, seciona-se cada grafo de comunicação global em um número de sub-grafos que seja, no máximo, igual ao número de nodos do sistema utilizado. Cada sub-grafo é associado a um nodo do sistema e diferentes sub-grafos podem ser associados a um mesmo nodo. Caso existam nodos multiprocessados, é realizada a segunda etapa, onde faz-se a alocação da computação do sub-grafo aos elementos processadores do nodo.

A alocação da computação em nodos e elementos processadores deve ser feita de forma que todas as computações atendam às restrições de temporização impostas pela aplicação. Para isso, o atraso introduzido pelo sub-sistema de comunicação precisa ser caracterizado, de modo a permitir a verificação de que nenhum Deadline[4] será violado devido a atrasos de comunicação. Quando da alocação da computação nos diferentes elementos processadores, deverão ser definidas as possíveis relações de exclusão entre computações que executam em um mesmo nodo.

d) Caracterização da Carga no Sub-Sistema de Comunicação

Os atrasos introduzidos pelo sub-sistema de comunicação são expressos no sistema através da modelagem da carga de comunicação, caracterizada pelos nós de operações de transmissão de dados. Esses nós conectam diferentes sub-grafos alocados em diferentes nodos e possuem parâmetros temporais determinados adequadamente, de modo a contribuir para tornar o sistema robusto.

As operações de transmissão são dotadas de parâmetros temporais Release-Time e Deadline, que garantem uma relativa flexibilidade para a construção do sistema e permitem a verificação das restrições de temporização dos programas e alguma tolerância a falhas de temporização no sub-sistema de comunicação. Além disso, os diversos nodos do sistema distribuído não necessitam manter uma sincronização justíssima.

Os nós de operação de transmissão devem ser também alocados a recursos específicos do sub-sistema de comunicação, a fim de permitir a caracterização do seu desempenho e também a análise deste recurso. Esses nós são completamente descritos por uma tupla (r, tt, d, p, ci), onde o parâmetro tt descreve o Tempo Máximo de Transmissão de Dados (pior caso), ci indica o recurso exato do sub-sistema de comunicação (Circuito) em que a operação de transmissão será realizada e os demais parâmetros possuem função idêntica de quando caracterizam computações[4].

Por razões de eficiência, considera-se que a comunicação entre computações de um mesmo nodo, mesmo que este seja multiprocessado, é realizada através de variáveis globais localizadas na memória global compartilhada. Dessa forma, não cabe caracterizar uma carga no sub-sistema de comunicação. Para esses casos abordagem prevê relações explícitas de exclusão.

e) Caracterização do Desempenho dos Recursos do Sistema

A caracterização do desempenho de cada elemento processador é dado pelo conjunto de computações a ele alocado (nós de um determinado sub-grafo), juntamente com a sua política de escalonamento. As operações de transmissão, que denotam a carga de comunicação, devidamente modeladas e alocadas a recursos do sub-sistema de comunicação, em conjunto com a sua política de escalonamento, caracterizam o desempenho do recurso.

---

4 Veja a descrição completa dos parâmetros de computações HRTS no Item 3.2 - Projeto de Software.

3 Descrição Detalhada da Abordagem

A abordagem pode ser descrita, em mais alto nível, através do diagrama esquemático apresentado na Figura 1.

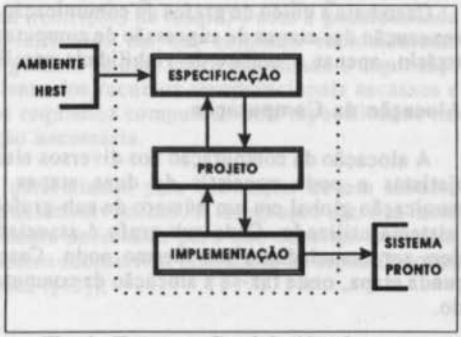

Fig. 1 - Estrutura Geral da Abordagem

O ambiente HRTS fornece subsídios para o desenvolvimento, através de iterações entre as fases de especificação, projeto e implementação, até a obtenção do sistema de software pronto. A Figura 2 apresenta a estrutura detalhada da abordagem.

Para melhor compreensão de seus detalhes, apresenta-se cada uma das fases e etapas através do desenvolvimento de um exemplo de um sistema simples. Esse exemplo foi originalmente proposto por Mok [7] e aqui se apresenta estendido para favorecer as demonstrações necessárias.

O sistema consiste de um controle automático de processo, que possui quatro sinais de entrada (*s*, *t*, *u* e *v*) e dois sinais de saída (*y* e *z*). Os sinais *s*, *t* e *u* são periódicos e o sinal *v* é esporádico. Os sinais *s*, *t* e *u* ocorrem a cada 80 ms. O mínimo intervalo entre ocorrências sucessivas do sinal *v* é de 160 ms. As ações de controle, as quais determinam a geração dos sinais de saída *y* e *z*, relacionadas com os sinais de entrada *s*, *t* e *u* devem sempre ser realizadas em, no máximo, 40 ms após a sua leitura. As ações de controle relacionadas com o sinal *v* devem sempre ser tomadas em 99 ms. Os sinais de entrada *s*, *t* e *v* produzem o sinal de saída *y* e o sinal de entrada *u* produz a saída *z*.

Esse sistema pode ser interpretado fisicamente como os sinais *s*, *t* e *u* expressando grandezas físicas, medidas no ambiente por sensores, o sinal *v* determinado por um operador e as saídas *y* e *z* sinais de controle para atuadores. Os sinais *y* e *t* são também utilizados para a determinação do estado[5] do processo controlado, de modo a otimizar decisões de controle. Considera-se que os programas necessários são

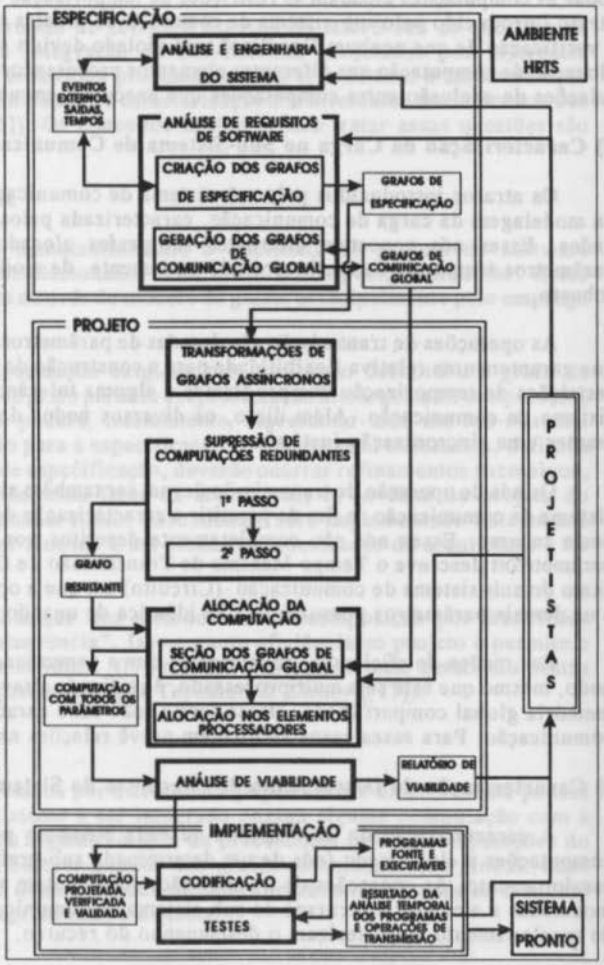

Fig. 2 - Estrutura Detalhada da Abordagem

5 O estado do processo controlado pode ser caracterizado pelo conjunto de ações e/ou reações a que o processo está submetido ou que deve executar, e indica a sua situação em um determinado instante de tempo.

fornecidos por uma biblioteca já existente e que os respectivos Tempos Máximo de Computação (c) já foram calculados e equivalem a seis unidades de tempo de elemento processador. Considera-se, ainda, que as operações de transmissão de dados possuem o parâmetro tt igual a 2 unidades de tempo. Adota-se que a unidade de tempo de elemento processador equivale 1 ms.

3.1 Especificação de Software

A fase de especificação da abordagem é composta de duas etapas: "Análise e Engenharia do Sistema" e "Análise de Requisitos de Software", descritas a seguir.

a) Análise e Engenharia do Sistema

Essa etapa consiste em estabelecer os requisitos do ambiente HRTS, os quais são definidos como um conjunto de relações, ordenadas temporalmente, entre os eventos externos ao sistema computacional de tempo real e as saídas necessárias. Deve-se notar que essas relações devem ser estabelecidas em função do Período ou do Mínimo Intervalo Entre Ativações Consecutivas[6] e do tipo dos eventos (esporádico ou periódico) para cada uma das saídas do sistema. O tempo máximo para a geração da saída (Deadline) já deverá ser especificado. Isso é feito obtendo-se informações ao nível do interfaceamento do software com os outros componentes do sistema, como por exemplo, os atuadores, os sensores e o processo controlado.

Para o exemplo temos o seguinte:

Eventos de Entrada

s : período = 80 ms, deadline = 40 ms, tipo = periódico
t : período = 80 ms, deadline = 40 ms, tipo = periódico
u : período = 80 ms, deadline = 40 ms, tipo = periódico
v : mínimo intervalo = 160 ms, deadline = 99 ms, tipo = esporádico

Eventos de Saída

y : período = 80 ms, deadline = 40 ms
z : período = 80 ms, deadline = 40 ms

Relações Ordenadas Temporalmente

O resultado dessa etapa são essas três relações, ao lado apresentadas.

b) Análise de Requisitos de Software

Uma vez estando definidos os requisitos do ambiente HRTS, deve-se passar a enfocar mais propriamente os requisitos do software. Isso é feito nessa etapa, onde são definidos dois tipos de grafos, os grafos de especificação e os grafos de comunicação global. Dando prosseguimento ao desenvolvimento do exemplo, temos o seguinte:

---

6 Relações assíncronas e restrições temporais assíncronas possuem, ao invés de Período, o parâmetro Mínimo Intervalo Entre Ativações Consecutivas - Mina. Restrições de tempo assíncronas são criadas para tratar eventos essporádicos.

Criação dos Grafos de Especificação

Os grafos de especificação são grafos acíclicos, que representam restrições de tempo definidas ao nível da aplicação e caracterizam a funcionalidade do software, através da especificação da função de cada nó do grafo, do seu Tempo Máximo de Computação (c), das relações de precedência e da comunicação entre esses nós. Para a criação de um grafo de especificação é considerado o tipo da restrição de tempo (periódica ou assíncrona) e o Período ou o Mínimo Intervalo Entre Ativações Consecutivas. O limite de tempo para a execução da computação denotada pelo grafo (Deadline) deve ser especificado, porém não é um parâmetro que influi na criação de um grafo de especificação. Esses grafos são construídos, pelos projetistas, a partir das relações definidas na etapa anterior.

Criação do Grafo de Especificação # 1:

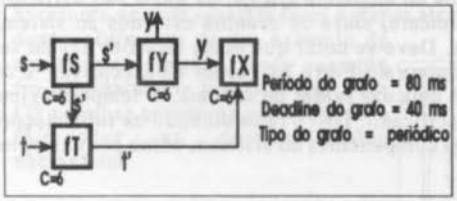

Fig. 3 - Grafo de Especificação # 1

Criação do Grafo de Especificação #2:

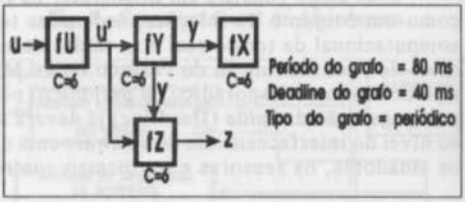

Fig. 4 - Grafo de Especificação # 2

Criação do Grafo de Especificação # 3:

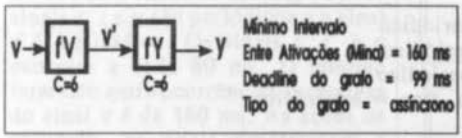

Fig. 5 - Grafo de Especificação # 3

A funcionalidade dos nós é a seguinte: fS: Pre-processamento do sinal *s*; fT: Pre-processamento do sinal *t*; fY: Cálculo do sinal *y*, considerando a variável de estado (*x*); fX: Cálculo da variável de estado (*x*); fU: Pre-processamento do sinal *u*; fZ: Cálculo do sinal *z*; fV: Pre-processamento do sinal *v*. Para o cálculo do sinal *z* é considerada a saída de fY. Estes grafos de especificação ainda serão completados com informações sobre o relacionamento entre diferentes grafos de especificação, isto após a geração dos grafos de comunicação global. A modelagem do sistema propiciada por estes grafos, que contém informações parciais, e os grafos completos permite o trabalho de diferentes equipes no sistema. No âmbito de cada equipe, o projeto pode ser compreendido e documentado. Os sub-sistemas projetados pelas diferentes equipes podem ser prontamente integrados em um único HRTS e, desta forma, o projeto de sistemas grandes e complexos pode ser adequadamente realizado.

Geração dos Grafos de Comunicação Global

Os grafos de comunicação global podem ser grafos cíclicos e denotam sub-sistemas compostos pela união de grafos de especificação. Grafos de especificação que possuem nós com funcionalidade idêntica ou relacionamentos de precedência são agrupados em um único grafo de comunicação global. A composição desses grafos visa dar aos projetistas uma visão global do sistema e também auxiliam refinando a especificação do relacionamento entre diferentes grafos de especificação. O grafo de comunicação global obtido a partir dos grafos de especificação criados na etapa anterior é apresentado na Figura 6.

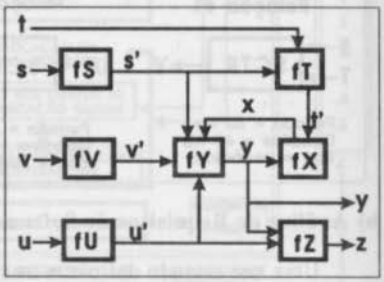

Fig. 6 - Grafo de Comunicação Global

A partir do grafo de comunicação global, constata-se que o nó fY possui como entrada os sinais *s'*, *u'*, *v'*, e *x*. Assim, os grafos de especificação podem ser acrescidos com informações completas dos diversos relacionamentos entre nós de diferentes grafos de especificação. Os grafos completos de especificação são apresentados a seguir.

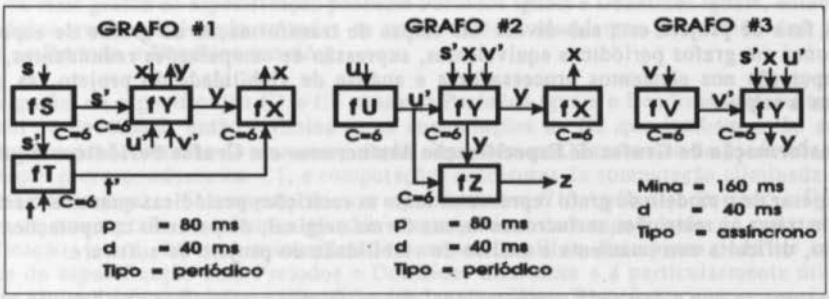

Fig. 7 - Grafos de Especificação Completos

O produto da etapa de análise de requisitos de software são os grafos de especificação e os grafos de comunicação global. A dinâmica de funcionamento do modelo de grafo é análoga a de diagramas de estado/transição ([20]). Um evento no ambiente externo poderá determinar alterações de estado no sistema, requerendo a execução de um determinado nó (ou um conjunto de nós) do modelo de grafo. Assim, o sistema executará as ações necessárias, denotadas pela funcionalidade de cada nó do modelo, para a produção dos resultados necessários.

3.2 Projeto de Software

A abordagem prevê que a fase de projeto é baseada na transformação dos requisitos, especificados na fase anterior, através de técnicas rigorosas, as quais podem ser automatizadas, contribuindo para a sua eficiência.

Nesta fase, todas as computações (nós do modelo de grafo) deverão ser descritas por uma tupla (r, c, d, p, n, ep, e, pr), a qual caracteriza completamente um programa. Os seus parâmetros[7] descrevem:

r : Release-Time. Indica um intervalo de tempo, contado a partir do início do Período, durante o qual o programa não poderá iniciar a sua execução;

c : Tempo Máximo de Computação. É a estimativa de pior caso no tempo execução do programa, estabelecido na fase de "Análise de Requisitos de Software";

d : Deadline. Especifica um intervalo de tempo, contado a partir do início do Período, em que o programa deve completar a sua execução;

p : Período;

n : Nodo. Indica o nodo do sistema distribuído em que o programa está alocado;

ep: Elemento Processador. Explicita em qual elemento processador do nodo o programa deverá executar;

e : Exclusões. É um conjunto que indica as relações de exclusão do programa, e;

pr: Precedências. É o conjunto que denota as relações de precedência do programa.

---

7 Os parâmetros temporais são expressos em unidades de tempo de elemento processador.

A fase de projeto está sub-divida nas etapas de transformação de grafos de especificação assíncronos em grafos periódicos equivalentes, supressão de computações redundantes, alocação da computação nos elementos processadores e análise de viabilidade do projeto, as quais são descritas a seguir.

a) Transformação de Grafos de Especificação Assíncronos em Grafos Periódicos Equivalentes

Apesar de o modelo de grafo representar tanto as restrições periódicas quanto as assíncronas, o fato de tratar as restrições assíncronas na sua forma original, disparando computações de modo aleatório, dificulta extremamente a análise de viabilidade do projeto de software.

Assume-se que a demanda computacional das restrições assíncronas não é muito elevada[8] e, dessa forma, é possível realizar a conversão heurística de um grafo de especificação assíncrono em um grafo periódico equivalente. Isso é feito pela criação do novo grafo com características diagramáticas idênticas ao anterior. Cada um dos nós do novo grafo deve possuir o Tempo Máximo de Computação (c) igualado ao do seu nó correspondente no grafo assíncrono e determinando-se os parâmetros *dp* (Deadline do grafo periódico equivalente) e *pp* (Período do grafo periódico equivalente), de forma que[9]:

1) Primeiramente se calcula o parâmetro *pp*. Inicialmente, se define o conjunto P, que é composto pelos divisores do parâmetro Mina (Mínimo Intervalo Entre Ativações). Para o exemplo empregado, o grafo de especificação # 3 deve ser convertido. O parâmetro Mina é 160 ms, então P = {1,2,4,5,8,10,16,...,80,160}. O parâmetro *pp* deve ser o maior elemento do conjunto P, tal que: Mina > = pp > = Sca e pp < =(Mina+1)/2, onde: Mina - Mínimo Intervalo Entre Ativações Consecutivas do grafo assíncrono e Sca - Somatório dos Tempos Máximo de Computação dos nós do grafo assíncrono. Portanto, pp < = (160 + 1) / 2 = 80 ms.

2) O novo Deadline (*dp*) deverá ser o maior número inteiro não negativo, tal que: da > = dp > = Sca e dp < = da - pp + 1. Portanto, dp < = 99 - 80 + 1 = 20 ms. O grafo transformado nessa etapa é apresentado na Figura 8.

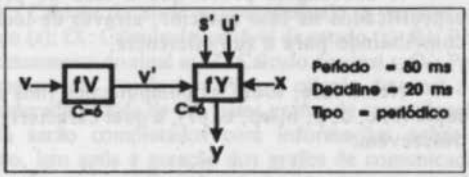

Fig. 8 - Grafo de Especificação # 3 Transformado

b) Supressão de Computações Redundantes

Se, por um lado, a especificação do sistema por meio de grafos de especificação pode conduzir a uma maior eficiência do projeto, bem como o facilitar o controle da complexidade e da manutenibilidade, por outro lado poderão ser também especificadas diversas computações redundantes, em diferentes grafos de especificação.

Para uma utilização eficiente dos recursos computacionais, todas as computações redundantes deverão ser suprimidas. Essa supressão é feita através da análise e redução de diversos grafos de especificação, que resultam, ao final do processo, em um único grafo acíclico, onde os parâmetros temporais não mais serão atribuídos às restrições de tempo, mas sim, atribuir-se-á os parâmetros temporais (p - Período e d - Deadline) aos nós do grafo. Essa supressão consiste dos seguintes passos:

8 Somente se a demanda computacional de uma restrição assíncrona for da ordem de 0.5, isto é, o Somatório dos Tempos Máximo de Computação (Sca) dividido pelo Deadline do grafo for menor ou igual a 0.5, será possível obter garantias de um escalonamento viável ([7]). Isso determina o interesse em transformá-la em uma restrição periódica.

9 Mok ([7]) demonstra que, para que a análise de viabilidade das computações assíncronas não se torne um problema computacionalmente intratável (Np-hard), a conversão heurística apresentada é uma estratégia genérica para solucionar a questão.

1) Se dois ou mais grafos de especificação possuem Períodos iguais e Deadlines iguais, então, cria-se um único grafo eliminando-se todas as computações redundantes pela união dos grafos. Atribui-se Período e Deadline aos nós.

2) Se dois grafos de especificação C1 e C2 possuem Períodos iguais e Deadlines diferentes, com deadline1 < deadline2, então, elimina-se as computações de C2 que também estão em C1. Computações predecessoras de uma computação eliminada em C2 tornar-se-ão predecessoras da computação correspondente em C1, e computações sucessoras da computação eliminada em C2 tornar-se-ão sucessoras da computação correspondente em C1. Atribui-se Período e Deadline aos nós do novo grafo, de modo a atender aos requisitos de temporização de cada grafo de especificação, isto é, deverá ser considerado o menor Deadline. Este passo pode ser aplicado a grafos de especificação com Períodos e Deadlines diferentes e é particularmente útil se os Períodos são múltiplos. Existem sérias dificuldades em utilizar Períodos primos entre si e deve-se procurar que sejam múltiplos, se possível. Prosseguindo com o desenvolvimento do exemplo, temos o seguinte:

1º Passo da Supressão

É realizada a união dos grafos de especificação # 1 e # 2, com Períodos e Deadlines idênticos. Isso é mostrado na Figura 9.

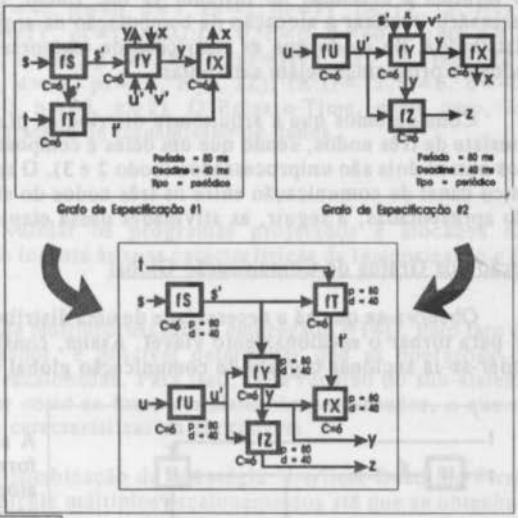

Fig. 9 - Primeiro Passo da Supressão

2º Passo da Supressão

Deve-se realizar a união do grafo obtido na Figura 9 com o grafo de especificação # 3, já transformado em seu correspondente periódico. O Grafo resultante é apresentado na Figura 10.

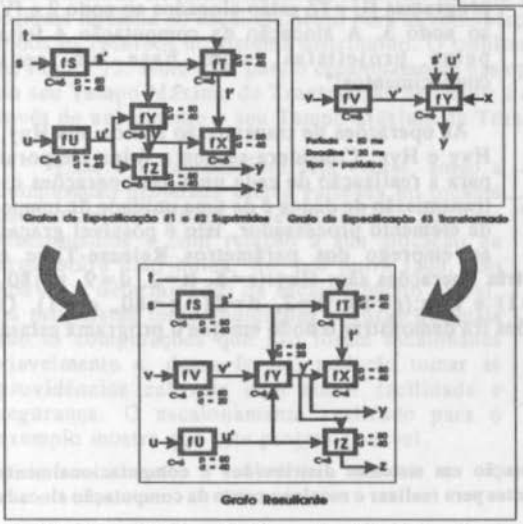

Fig. 10 - Segundo Passo da Supressão - Grafo Resultante

O resultado, além da supressão das computações redundantes, é a obtenção dos parâmetros Período e Deadline individuais para cada computação do sistema. É a computação especificada pelos nós do grafo resultante dessa etapa que deverá ser implementada para atender aos objetivos da aplicação.

c) Alocação da Computação

Em razão de se empregar sistemas distribuídos e, principalmente, pelo fato de sistemas uniprocessados poderem não atender à demanda computacional da aplicação, deve-se executar esta etapa. Considerando-se que o Tempo Máximo de Computação de cada um dos programas é 6 ms, o exemplo empregado é um bom caso de demanda que não é satisfeita por um uniprocessador. Tomando-se o grafo resultante da etapa anterior, verifica-se a necessária precedência dos programas fS, fU e fV em relação ao programa fY e desse em relação aos programas fZ e fX. Um escalonamento "Earliest-Deadline-First" ([8]) para essa computação é mostrado na Figura 11.

Percebe-se, assim, que esse escalonamento não atende às restrições de temporização impostas aos programas, de uma maneira global. Veja, por exemplo, que o Deadline do programa fY é 20 ms, mas a sua execução termina apenas no instante 24 ms. Também o programa fX, com Deadline 40 ms, só termina a execução no instante 42 ms. Então, torna-se necessário realizar a alocação da computação na arquitetura distribuída, de modo que as restrições de temporização de todos os programas sejam satisfeitas.

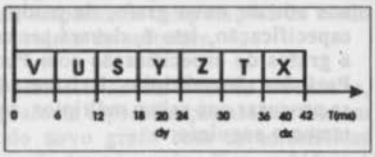

Fig. 11 - Escalonamento do Grafo Resultante em Uniprocessador

Consideremos que a arquitetura distribuída planejada para o desenvolvimento da aplicação consiste de três nodos, sendo que um deles é composto por dois elementos processadores (nodo 1) e os outros dois são uniprocessados (nodo 2 e 3). O sub-sistema de interconexão é definido por um único canal de comunicação entre os três nodos do sistema, por exemplo um barramento. Assim, são apresentadas, a seguir, as atividades dessa etapa de projeto empregando-se o exemplo.

Seção dos Grafos de Comunicação Global

Observa-se que há a necessidade de uma distribuição da computação precedente ao programa fY para tornar o escalonamento viável. Assim, considerando-se a disponibilidade de três nodos, poder-se-ia secionar o grafo de comunicação global da forma apresentada na Figura 12.

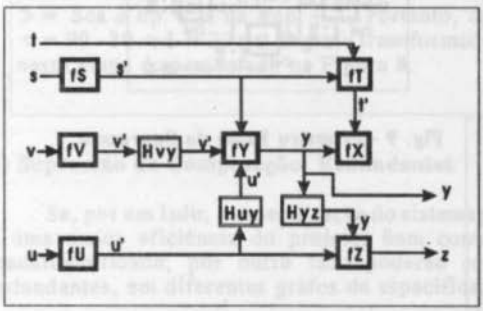

Fig. 12 - Seção do Grafo de Comunicação Global

A seção em três sub-grafos foi estabelecida de forma que os programas fS, fT, fY e fX sejam alocados a um nodo do sistema (nodo 1). Os programas fU e fZ estão alocados ao nodo 2 e fV ao nodo 3. A alocação da computação é feita pelos projetistas, com base em seus conhecimentos[10].

As operações de transmissão são os nós Huy, Hvy e Hyz. Considera-se que a folga temporal para a realização de cada uma das operações de transmissão de dados é de uma unidade de tempo de elemento processador, isto é possível graças ao emprego dos parâmetros Release-Time e Deadline (Item 2.2-d). Os parâmetros dessas três operações são: Huy:(r=6, tt=2, d=9, p=80, ci=1); Hvy:(r = 6, tt=2, d=9, p=80, ci=1) e Hyz:(r=18, tt=2, d=21, p=80, ci=1). O parâmetro n (nodo) de cada uma das computações irá demonstrar o nodo em que o programa estará alocado.

10 A problemática geral da alocação da computação em sistemas distribuídos é computacionalmente intratável, mesmo que existam algoritmos eficientes para realizar o escalonamento da computação alocada em cada processador ([7]).

Alocação dos Programas nos Elementos Processadores

Existe um nodo que é multiprocessado, então as computações associadas a esse nodo devem ser alocadas a elementos processadores específicos. Isso é feito determinando-se o parâmetro ep. Pode-se alocar fS, fY e fX no elemento processador 1 e fT no elemento processador 2. Então, ep de fT será 2 e os parâmetros ep de fS, fY e fX serão 1.

Por fim, as possíveis relações de exclusão entre grupo de computações que executam em um mesmo nodo deverão ser definidas. Para isso, define-se o parâmetro e, que indica as relações de exclusão da computação. No exemplo, os programas fT e fY são mutuamente exclusivos, pois está se considerando que a comunicação desses programas com o programa fS ocorre através de variáveis globais compartilhadas. Note também que fS possui precedência em relação aos programas fT e fS, por isso não se justifica que fS faça parte da relação de exclusão.

Aqui, todas as computações do sistema já estarão completamente descritas. Os seus parâmetros pertinentes são : fS:(r=0, c=6, d=40, p=80, n=1, ep=1, pr=fT, fX, fZ, fY, Hyz); fT:(r=0, c=6, d=40, p=80, n=1, ep=2, e=fY, pr=fX); fU:(r=0, c=6, d=40, p=80, n=2, pr=Huy, fY, fX, Hyz, fZ); fV:(r=0, c=6, d=20, p=80, n=3, pr=Hvy, fY, fX, Hyz, fZ); fY:(r=12, c=6, d=20, p=80, n=1, ep=1, e=fT, pr=fX, Hyz, fZ); fX:(r=12, c=6, d=40, p=80, n=1, ep=1); fZ:(r=21, c=6, d=40, p=80, n=2). O Release-Time, neste caso, foi determinado pelo instante de término das operações de transmissão de dados.

d) Análise de Viabilidade do Projeto

Essa etapa consiste em verificar e validar os programas projetados e alocados em uniprocessadores e em multiprocessadores, no tocante às suas características de temporização e de interdependência.

Para a sua consecução, um escalonador de multiprocessador é empregado ([14]). Pelo fato de se utilizar um escalonador de multiprocessador, é possível determinar que as operações de transmissão de dados sejam também por ele escalonadas. Para isso, cada recurso do sub-sistema de comunicação é entendido pelo escalonador como se fosse um elemento processador, o que na realidade não faz muita diferença, devido às características da abordagem.

O algoritmo implementado utiliza uma combinação da estratégia "Earliest-Deadline-First" com o método de minimização da latência. Calcula múltiplos escalonamentos até que se obtenha a mínima latência no escalonamento para as computações e operações de transmissão, considerando todos os recursos do sistema distribuído. O resultado dessa etapa é o escalonamento apresentado na Figura 13. Note que, para o escalonamento das operações de transmissão de dados, o somatório do seu Tempo Máximo de Transmissão de Dados e da sua folga temporal deve ser empregado, ao invés de unicamente o seu Tempo Máximo de Transmissão de Dados.

Caso o escalonamento seja viável, então a computação projetada estará validada no tocante às suas características de temporização, aos seus sincronismos e com relação à sua alocação na arquitetura distribuída. Se, por outro lado, não houver a determinação de escalonamento viável, os projetistas terão indicações seguras de quais são as computações que não foram escalonadas viavelmente e, dessa forma, poderão tomar as providências cabíveis com maior facilidade e segurança. O escalonamento realizado para o exemplo mostra que este projeto é viável.

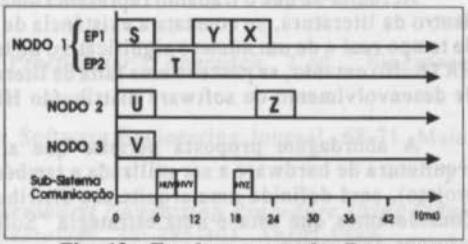

Fig. 13 - Escalonamento dos Programas e das Operações de Transmissão

3.3 Implementação

Caso os programas projetados ainda não existam e, por conseguinte, não foram estabelecidos os seus Tempos Máximo de Computação, então os programas deverão ser implementados e deverão passar por experimentos confiáveis, a fim de demonstrar que poderão atender às restrições de projeto (especificamente com relação ao parâmetro c). O mesmo deverá ocorrer com as operações de transmissão (parâmetro tt).

Se os programas implementados ou as operações de transmissão de dados não atenderem às restrições de temporização do projeto, então, ou o projeto deverá ser modificado para acomodar isso - através de uma possível reorganização dos parâmetros temporais ou de diferentes estratégias de alocação -, ou deverá ser feita uma atualização na arquitetura do hardware, buscando maior desempenho, o que na prática é bastante factível quando se emprega sistemas distribuídos.

De qualquer modo, é desejável, após a correta determinação dos parâmetros c e tt, que se reexecute a etapa de análise de viabilidade do projeto, com os parâmetros exatos. Isso visa, principalmente, verificar a utilização precisa de cada recurso do sistema. Essa taxa de utilização é um bom indicador da capacidade de crescimento do sistema, favorecendo o gerenciamento do seu ciclo de vida e também a determinação do balanceamento de carga. O cálculo da taxa de utilização (Tu), por recurso, é o seguinte: Tu = ((Te - TTge) / Te) * 100, onde: Te - Tamanho do Escalonamento e TTge : Tamanho Total de Gaps no Escalonamento.

Para o exemplo temos a seguinte taxa de utilização para cada recurso (note que o tamanho do escalonamento é 80 ms) :

Nodo 1: Elemento Processador 1 : ((80-62)/80)*100= 22.5 %
Elemento Processador 2 : ((80-74)/80)*100= 7.5 %

Nodo 2 : ((80-68)/80)*100= 15 %

Nodo 3 : ((80-74)/80)*100= 7.5 %

Sub-Sistema de Comunicação : ((80-71)/80)*100= 11.25 %

4 Conclusões

O artigo apresentou uma abordagem integrada e completa para o desenvolvimento de software distribuído HRTS, que enfatiza principalmente a especificação e o projeto. Esta abordagem foi definida como parte dos requisitos de uma dissertação de mestrado ([16]), cujo objetivo era contribuir para temas relacionados com a Engenharia de Software destes sistemas.

Acredita-se que o trabalho representa uma contribuição significativa para a área. Isso porque, dentro da literatura, se constata a existência de diversos trabalhos sobre especificação de sistemas de tempo real e de um número significativo de trabalhos relativos ao escalonamento da computação HRTS. No entanto, se percebe uma falta de literatura quando se deseja compreender todo o processo de desenvolvimento de software distribuído HRTS.

A abordagem proposta permite que a especificação do sistema seja independente da arquitetura de hardware a ser utilizada e também prevê que, em uma segunda instância (na fase de projeto), será definida uma arquitetura distribuída específica e faz-se a alocação da computação. Consideramos que esta é uma estratégia "Software-First" ([10]) adequada, que reduz custos e esforços de projeto.

Foram realizados testes com a abordagem, através do desenvolvimento de alguns sistemas simples e de um exemplo de HRTS distribuído da área de Meteorologia, denominado "Previsão de Microbursts Empregando Radar Doppler" ([13]). Estes desenvolvimentos foram realizados

manualmente e a análise de viabilidade do projeto foi realizada pelo escalonador implementado. Bons resultados foram obtidos com a sua utilização.

O objetivo final dos estudos é a confecção de uma ferramenta. Isso é bastante factível, principalmente, pelo rigor da abordagem, que possibilita a sua automação e, por conseguinte, um desenvolvimento eficaz e eficiente. Para a definição de uma ferramenta completa, algumas questões necessitam ser melhor refinadas. Por exemplo, a determinação dos parâmetros c e tt exatos, tópicos estes que não foram contemplados pelos estudos realizados. Devido às características intrínsecas da abordagem proposta, espera-se que este trabalho contribua para a uma maior qualidade e produtividade no desenvolvimento do software em questão.

5 Referências Bibliográficas

1. Bowen, J. & Stavridou, V. ***Safety-Critical Systems, Formal Methods and Standards***. Oxford, UK, Oxford University, 1992, 36p.

2. Davis, A. M. ***A Comparison of Techniques for the Specification of External System Behavior***. Communications of the ACM, 31(9): 1098-1115, Setembro, 1988.

3. Harel, D. ***On Visual Formalisms***. Communications of the ACM, 31(5): 514-531, Maio, 1988.

4. Koymans, R. & Kuiper, R. ***Paradigms for Real Time Systems***. Lecture Notes in Computer Science, Spring Verlag, 331: 159-174, Setembro, 1988.

5. Leveson, N. G. ***The Challenge of Building Process Control Software***. IEEE Software, 55-62, Novembro, 1990.

6. Leveson, N. G. ***Software Safety in Embedded Computer Systems***. Communications of the ACM, 34-46, Fevereiro, 1991.

7. Mok, A. K. L. ***Fundamental Design Problems of Distributed Systems for the Hard-Real-Time Environment***. Cambridge, MA, EUA, MIT, 1983. 183p.

8. Mok, A. K. L. ***The Design of Real-Time Programming Systems Based on Process Models***. In: IEEE Real-Time Systems Symposium, Los Alamitos, CA, EUA, Dezembro, 1984. Austin, EUA, IEEE Press, 1984, 5-17.

9. Neumann, P. G. ***Risks: Cumulative Index of Software Engineering Notes - Illustrative Risks to The Public***. ACM Software Engineering Notes, 22-26, Janeiro, 1989.

10. Nielsen, K. ***Ada in Distributed Real-Time Systems***. 1.ed. New Yord, NY, EUA, McGraw-Hill Book Company, 1990. 371p.

11. Pressman, R. S. ***Software Engineering: A Practitioner's Approach***. 2.ed. Singapore, McGraw-Hill Inc., 1987. 352p.

12. Pyle, I. C. ***Real-World Software Engineering***. Software Engineering Journal, 68-71, Maio, 1991.

13. Roberts, R. D. & Wilson J. W. ***A Proposed Microburst Nowcasting Procedure Using Single Doppler Radar***. Journal of Applied Meteorology, 28: 285-303, Abril, 1989.

14. Shepard, T. & Gagné, J. A. M. ***A Pre-Run-Time Scheduling Algorithm For Hard Real-Time Systems***. IEEE Transactions on Software Engineering, 17(7): 669-677, Julho, 1991.

15. Stankovic, J. A. ***Misconceptions About Real Time Computing: A Serious Problem for Next-Generation Systems***. IEEE Computer, 10-19, Outubro, 1988.

16. Tepedino, P. R. P. ***Uma Abordagem de Desenvolvimento de Software de Sistemas de Tempo Real Crítico em Arquiteturas Distribuídas***. São Carlos, UFSCar, Julho, 1993, 165p.

17. Tepedino, P. R. P. & Kirner, T. G. ***Produtividade e Qualidade no Desenvolvimento de Software Distribuído de Sistemas de Tempo Real Crítico: O Emprego de Métodos Formais e Modelo de Grafo***. In: XXVI Congresso Nacional de Informática e Telecomunicações, Brasília, DF, Outubro, 1993.

18. Ward, P. & Mellor, S. J. ***Structured Development for Real-Time Systems***. 1.ed. London, England, Yourdon Press, 1985. 155p.

19. Xu, J. & Parnas, D. L. ***On Satisfying Timing Constraints in Hard-Real-Time Systems***. In: Conference on Software for Critical Systems, New Orleans, Louisiana, EUA, 1991. ACM Press, 1991, 132-146.

20. Yourdon, E. ***Modern Structured Analysis***. 2.ed. Englewood Cliffs, New Jersey, EUA, Prentice-Hall Inc., 1989, 672p.