Capítulo 1

Introdução sobre o Tempo Real

Esse capítulo visa esclarecer o entendimento de tempo real dos autores, definir conceitualmente os Sistemas de Tempo Real e apresentar os problemas e desafios que lhes são relacionados.

1.1 Os Sistemas de Tempo Real

Na medida em que o uso de sistemas computacionais prolifera na sociedade atual, aplicações com requisitos de tempo real tornam-se cada vez mais comuns. Essas aplicações variam muito em relação à complexidade e às necessidades de garantia no atendimento de restrições temporais. Entre os sistemas mais simples, estão os controladores inteligentes embutidos em utilidades domésticas, tais como lavadoras de roupa e videocassetes. Na outra extremidade do espectro de complexidade estão os sistemas militares de defesa, os sistemas de controle de plantas industriais (químicas e nucleares) e o controle de tráfego aéreo e ferroviário. Algumas aplicações de tempo real apresentam restrições de tempo mais rigorosas do que outras; entre esses, encontram-se os sistemas responsáveis pelo monitoramento de pacientes em hospitais, sistemas de supervisão e controle em plantas industriais e os sistemas embarcados em robôs e veículos (de automóveis até aviões e sondas espaciais). Entre aplicações que não apresentam restrições tão críticas, normalmente, são citados os vídeogames, as teleconferências através da Internet e as aplicações de multimídia em geral. Todas essas aplicações que apresentam a característica adicional de estarem sujeitas a restrições temporais, são agrupados no que é usualmente identificado como Sistemas de Tempo Real.

Metodologias e ferramentas convencionais são usadas, em uma prática corrente, no projeto e implementação de sistemas de tempo real. A programação dessas aplicações é feita com o uso de linguagens de alto nível, em geral eficientes, mas com construções não deterministas ou ainda, com linguagens de baixo nível. Em ambos os casos, sem a preocupação de tratar o tempo de uma forma mais explícita, o que torna difícil a garantia de implementação das restrições temporais. Os sistemas operacionais ou núcleos de tempo real, que gerenciam interrupções e tarefas e permitem a programação de temporizadores e de *"timeout"*, são para muitos projetistas as ferramentas suficientes para a construção de sistemas de tempo real. Embora esses suportes apresentem mecanismos para implementar escalonamentos dirigidos a prioridades, essas prioridades nunca refletem as restrições temporais definidas para essas aplicações.

Essas prioridades são determinadas usualmente a partir da importância das funcionalidades presentes nessas aplicações; o que não leva em conta, por exemplo, que o grau de importância relativa de uma função da aplicação nem sempre se mantém igual durante todo o tempo de execução desta.

Essas práticas correntes têm permitido resolver de forma aceitável e durante muito tempo certas classes de problemas de tempo real nas quais as exigências de garantia sobre as restrições temporais não são tão rigorosas. Entretanto essas técnicas e ferramentas convencionais apresentam limitações. Por exemplo, a programação em linguagem *"Assembly"* produz programas com pouca legibilidade e de manutenção complexa e cuja a eficiência está intimamente ligada à experiência do programador. Acrescenta-se a essas limitações, as dificuldades advindas do uso de ferramentas e metodologias que permitem a verificação apenas da correção lógica nessas aplicações. Em conseqüência, o software obtido dessa forma é considerado altamente imprevisível e sem garantia de um comportamento correto durante todo seu tempo de uso; situações perigosas podem resultar da utilização de software assim produzido.

Além do aspecto dessas ferramentas convencionais não tratarem da correção temporal, um outro fato que influi sobre o que se pode considerar como prática corrente é o desconhecimento do que seria tempo real. A grande maioria dos sistemas de tempo real vem sendo projetada e implementada até hoje a partir de uma visão errada de que tempo real é simplesmente reduzido a uma questão de melhoria no desempenho [Sta88].

As exigências de segurança num número cada vez maior de aplicações e a ligação dessa com a correção temporal desses sistemas colocam em xeque as metodologias e ferramentas convencionais, sob pena de perdas em termos financeiro, ambiental ou humano. Essas aplicações necessitam de algoritmos, de suportes computacionais e de metodologias que ultrapassam as características das ferramentas até então utilizadas e lançam novos desafios para os projetistas desse tipo de sistemas. Nas aplicações mais críticas, são assumidos situações extremas e pessimistas com hipóteses de carga de pico e cenários de falhas, sendo que mesmo nestas situações extremas, de pior caso, o sistema de tempo real deve manter as restrições temporais impostas pelo seu ambiente. Há quem considera que a famosa lei de Murphy é o paradigma pessimista ideal a ser considerado para as análises de Sistemas de Tempo Real com restrições temporais críticas [But97].

Antes de apresentarmos abordagens para que se garanta a correção temporal em sistemas de tempo real é necessário que se entenda o conceito de tempo.

1.2 O Tempo: Diferentes Interpretações

A noção de tempo em sistemas informáticos é difícil de ser expressa, podendo assumir diferentes enfoques. Na seqüência abaixo, são apresentados alguns desses enfoques com as suas respectivas interpretações que serão utilizadas nesse livro

[Ray91], [Mot92], [Kop92a], [Jos91]. Alguns desses enfoques são colocados em contraposição:

- *Tempo na Execução* considerado como um recurso a ser gasto durante a execução de um programa como outros recursos físicos ou lógicos e o *Tempo na Programação* visto como uma grandeza a ser manipulada pelo programa como outros tipos de variáveis; na execução o tempo intervém de forma implícita, enquanto na programação, ele pode ser visto de forma implícita ou explícita;
- *Tempo Lógico* que é definido a partir de relações de precedência entre eventos, o que permite o estabelecimento de ordens causais sobre um conjunto de eventos e o *Tempo Físico* que é um tempo métrico que permite expressar quantitativamente a distância entre eventos e estabelecer ordens totais entre eventos;
- *Tempo Denso* que segue a natureza uniforme e continua do tempo físico e é isomorfo ao conjunto dos reais e o *Tempo Discreto*, uma simplificação geralmente aceita do anterior, porém isomorfo em relação ao conjunto dos naturais positivos;
- *Tempo Global*, noção abstrata que permite ao usuário de um sistema distribuído ter acesso a um instante de referência único em qualquer parte do sistema e o *Tempo Local* observável localmente nos diferentes nós de um sistema distribuído; tanto o tempo global quanto o tempo local podem ser físicos ou lógicos;
- *Tempo Absoluto* com referência estabelecida a partir de um evento global e o *Tempo Relativo* tendo como referência um evento local; o tempo absoluto é sempre global, enquanto o tempo relativo é sempre local.

Dependendo dos tipos de sistemas e das abordagens usadas na descrição de seus comportamentos, alguns desses enfoques podem ser assumidos para representar a noção de tempo. O que é relevante num sistema de tempo real é que o tempo, de forma implícita ou explícita, é um componente essencial e intrínseco no comportamento deste.

1.3 Conceituação Básica e Caracterização de um Sistema de Tempo Real

Excetuando sistemas computacionais – normalmente identificados como *Sistemas Transformacionais* que calculam valores de saída a partir de valores de entrada e depois terminam seus processamentos como ocorre, por exemplo, com compiladores, programas de engenharia econômica e programas de cálculo numérico –, uma grande parte dos sistemas computacionais atuais interagem permanentemente com os seus ambientes. Entre esses, distingue-se os chamados *Sistemas Reativos* que reagem enviando respostas continuamente à estímulos de entrada vindos de seus ambientes. Sistemas de tempo real de uma forma geral se encaixam neste conceito de sistemas reativos:

- Um *Sistema de Tempo Real (STR)* é um sistema computacional que deve reagir a estímulos oriundos do seu ambiente em prazos específicos.

O atendimento desses prazos resulta em requisitos de natureza temporal sobre o comportamento desses sistemas. Em conseqüência, em cada reação, o sistema de tempo real deve entregar um resultado correto dentro de um prazo específico, sob pena de ocorrer uma falha temporal. O comportamento correto de um sistema de tempo real, portanto, não depende só da integridade dos resultados obtidos (correção lógica ou *"correctness"*) mas também dos valores de tempo em que são produzidos (correção temporal ou "*timeliness"*). Uma reação que ocorra além do prazo especificado pode ser sem utilidade ou até representar uma ameaça. Descrições semelhantes de sistemas de tempo real são encontradas na literatura da área ([Aud93], [You82], [StR88], [StR90], [Jos91], [Kop92b], [LeL90] e [But97]).

A maior parte das aplicações tempo real se comportam então como sistemas reativos com restrições temporais. A reação dos sistemas de tempo real aos eventos vindo do ambiente externo ocorre em tempos compatíveis com as exigências do ambiente e mensuráveis na mesma escala de tempo. A concepção do sistema de tempo real é diretamente relacionada com o ambiente no qual está relacionado e com o comportamento temporal do mesmo. Na classe de Sistema de Tempo Real na qual se encontram os sistemas embutidos (*"Embedded Systems"*) e os sistemas de supervisão e controle, distingue-se entre o Sistema a Controlar, o Sistema Computacional de Controle e o Operador. O Sistema a Controlar e o Operador são considerados como o Ambiente do Sistema Computacional. A interação entre os mesmos ocorre através de interfaces de instrumentação (compostas de sensores e atuadores) e da interface do operador. A figura 1.1 representa esse tipo de Sistema de Tempo Real.

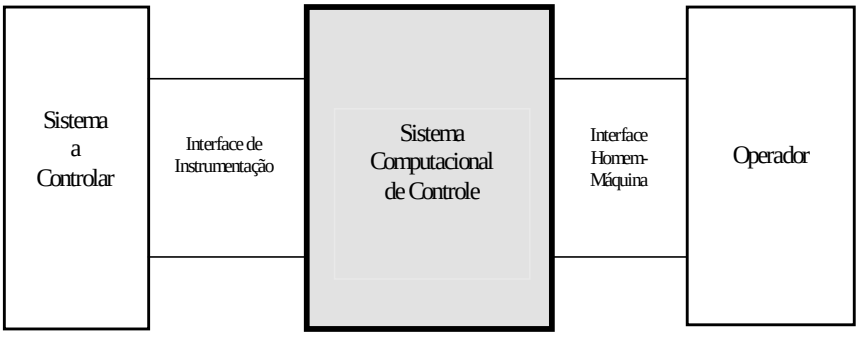

Figura 1.1: Sistema de Tempo Real

Existem também situações nas quais as restrições temporais não são impostas pelo comportamento dinâmico de um eventual Sistema a Controlar mas pelas exigências dos serviços a serem oferecidos a um usuário humano ou computacional (p.ex. no caso do

manuseio ou da apresentação de vídeos em sistemas multimídias). Nesses casos utiliza-se a noção de *Serviço de Tempo Real* como sendo um serviço que deve ser oferecido dentro de restrições de tempo impostas por exigências externas ao próprio Sistema Computacional [DEL91]. Então, numa generalização podemos assumir:

- Um *Sistema de Tempo Real* deve ser então capaz de oferecer garantias de *correção temporal* para o fornecimento de todos os seus serviços que apresentem restrições temporais.

1.4 A Previsibilidade nos Sistemas de Tempo Real

Uma das crenças mais comuns é que o problema de tempo real se resolve pelo aumento da velocidade computacional. A rapidez de cálculo visa melhorar o desempenho de um sistema computacional, minimizando o tempo de resposta médio de um conjunto de tarefas, enquanto o objetivo de um cálculo em tempo real é o atendimento dos requisitos temporais de cada uma das atividades de processamento caracterizadas nesses sistemas [Sta88]. Ter um tempo de resposta curto, não dá nenhuma garantia que os requisitos temporais de cada processamento no sistema serão atendidos. O desempenho médio não tem importância para o comportamento de um sistema composto de diversas atividades com restrições temporais. Mais do que a rapidez de cálculo, para os sistemas de tempo real, importa o conceito de *previsibilidade*.

Um sistema de tempo real é dito ser *previsível* (*"predictable*") no domínio lógico e no domínio temporal quando, independentemente de variações ocorrendo à nível de hardware (i.e. desvios do relógio), da carga e de falhas, o comportamento do sistema pode ser antecipado, antes de sua execução. Para se poder prever a evolução de um sistema de tempo real e garantir dentro de certos limites as suas restrições temporais, é necessário definir um conjunto de hipóteses sobre o comportamento do ambiente externo no que diz respeito à carga e as falhas:

- *A hipótese de carga:* ela determina o que corresponde a carga computacional de pico (carga máxima) gerada pelo ambiente em um intervalo mínimo de tempo, entre cada reação do sistema de tempo real. Mesmo eventos que ocorrem esporadicamente como os que levam a situações críticas (p.ex. alarme em planta nuclear) devem ser levados em conta para determinar essa carga computacional;
- *A hipótese de falhas*: ela descreve os tipos e freqüências de falhas com os quais o sistema deve conviver em tempo de execução, continuando a atender os seus requisitos funcionais e temporais.

Consequentemente, de um ponto de visto rigoroso, para se assumir a previsibilidade de um sistema (ou de um serviço) de tempo real, precisa-se conhecer *a priori* o comportamento de um sistema, levando-se em conta a pior situação de carga ocorrendo, simultaneamente, com as hipóteses de falhas. Como se vê é necessário nesse caso que

as hipóteses de carga e de falha que descrevem o comportamento do ambiente sejam definidas de forma realista, o que nem sempre é uma tarefa simples.

Mas a garantia de previsibilidade não depende só da carga computacional ativada pelo ambiente e das hipóteses de falhas. Um conjunto de fatores ligados a arquitetura de hardware, ao sistema operacional e as linguagens de programação são também importantes. Ou seja, os tempos gastos, no pior caso, na execução de códigos da aplicação (tempo de computação) e em funções de suportes devem ser perfeitamente conhecidos ou determinados previamente. Esses tempos limites são necessários nas análises e verificações do comportamento temporal do sistema de tempo real.

Se consideramos os tempos de computação dos códigos de aplicação envolvidos, muitas das construções nas linguagens de programação de caracter geral são fontes de não determinismo, tornando os sistemas não previsíveis. Como exemplo, podemos citar construções de recorrência não limitadas (laços não limitados) ou ainda, primitivas da linguagem Ada como o *"delay"* que estipula um limite mínimo mas não garante um limite máximo na suspensão de uma tarefa e o *"select"* que na ativação de alternativas de fluxos de instruções faz uma escolha aleatória entre aquelas com guardas abertos. Os mecanismos e protocolos de acesso a recursos compartilhados disponíveis nestas linguagens, geralmente não limitados, podem também tornar os códigos programados não previsíveis. Entretanto, melhorias são obtidas em linguagens destinadas a programação de tempo real que eliminam chamadas recursivas e estruturas dinâmicas de dados e onde são permitidos apenas laços limitados no tempo. Além disso, estas linguagens permitem expressar comportamentos temporais através de construções apropriadas. Real-Time Euclid [KSt86] e Real-Time Concurrent C [GeR91] são exemplos dessas linguagens.

O hardware também influi sobre a previsibilidade de comportamento dos sistemas de tempo real. O processador com suas instruções de *"prefetch"*, os dispositivos de acesso direto à memória (DMA), e mecanismos de memória *"cache"* são outras fontes de não determinismo; o uso destes mecanismos ou dispositivos dificulta a determinação dos tempos de computação no pior caso e consequentemente a análise da previsibilidade. As características de sistemas operacionais e suportes são também determinantes para que se conheça os tempos limites envolvidos em seus processamentos.

Diante do exposto acima, é importante enfatizar que em cada etapa do ciclo de desenvolvimento de um Sistema de Tempo Real, torna-se necessário o uso de ferramentas e metodologias apropriadas que permitem verificar o comportamento do sistema e sua implementação como previsíveis. A previsibilidade é o requisito necessário que deve ser introduzido em paradigmas de computação clássico para se poder atender aplicações de tempo real, em particular quando estas apresentam requisitos temporais rígidos.

Nesta seção, a noção de previsibilidade que foi introduzida está associada a uma antecipação determinista do comportamento temporal do sistema. Ou seja, o sistema é previsível quando podemos antecipar que todos os prazos colocados a partir das

interações com o seu ambiente serão atendidos. Mas na literatura, alguns autores como em [JLT85] também associam o conceito de previsibilidade a uma antecipação probabilista do comportamento do sistema, baseada em estimativas ou simulações que estipulam probabilidades dos prazos a serem atendidos. A previsibilidade probabilista é útil em sistemas onde a carga computacional não pode ser conhecida a priori e portanto, não se consegue uma garantia em tempo de projeto do atendimento de todos os prazos.

As discussões apresentadas acima sobre a previsibilidade, no que se refere a aspectos de implementação, estão ligadas a um tipo de abordagem na construção de sistemas de tempo real. Existem outras abordagens onde fatores ligados à implementação não são levados em conta e por hipótese, válida num grande número de aplicações, o sistema de tempo real é assumido com tendo um comportamento determinista e por conseqüência a sua previsibilidade garantida. Essas abordagens que apresentam formas distintas de conseguir a previsibilidade de sistemas de tempo real são introduzidas no item 1.6 e apresentadas em detalhe no transcorrer desse livro.

1.5 Classificação dos Sistemas de Tempo Real

Os sistemas de tempo real podem ser classificados a partir do ponto de vista da Segurança (*"Safety"*) em: Sistemas Não Críticos de Tempo Real (ou STRs brandos, *"Soft Real Time Systems"*) quando as conseqüências de uma falha devida ao tempo é da mesma ordem de grandeza que os benefícios do sistema em operação normal (ex. sistema de comutação telefônico, sistema de processamento bancário); e Sistemas Críticos de Tempo Real (ou STRs duros, *"Hard Real Time Systems"*), quando as conseqüências de pelo menos uma falha temporal excedam em muito os benefícios normais do sistema (ex. sistema de controle de vôo, ou de sinalização em ferrovias, sistema de controle de planta nuclear). Nesse caso, essa falha é dita catastrófica. Previsibilidades probabilista e determinista são associadas aos dois grupos distinguidos acima, respectivamente.

O grupo de sistemas críticos de tempo real pode ser subdividido em: Sistemas de Tempo Real Crítico Seguros em Caso de Falha (*"fail safe"*) onde um ou vários estados seguros podem ser atingidos em caso de falha (por exemplo, parada obrigatória de trens no caso de falha do sistema de sinalização ferroviário); e Sistemas de Tempo Real Crítico Operacionais em Caso de Falha (*"fail operational"*) que, na presença de falhas parciais, podem se degradar fornecendo alguma forma de serviço mínimo (por exemplo, sistema de controle de vôo com comportamento degradado mas ainda seguro).

Alguns autores apresentam a sua visão de sistemas de tempo real baseada no ponto de vista da implementação, distinguindo então: Sistemas de Resposta Garantida ("*guaranteed response system*") onde existem recursos suficientes para suportar a carga de pico e o cenário de falhas definido; e Sistemas de Melhor Esforço (*"best effort system"*) quando a estratégia de alocação dinâmica de recursos se baseia em estudos

probabilistas sobre a carga esperada e os cenários de falhas aceitáveis. Essa classificação é similar a anterior. A maior parte dos sistemas de tempo real vem sendo projetados de acordo com o paradigma de melhor esforço, satisfatório de fato apenas para os sistemas não críticos de tempo real e levando a situações danosas para os sistemas críticos de tempo real para os quais é aconselhado seguir o paradigma de resposta garantida.

1.6 O Problema Tempo Real e Abordagens para a sua Solução

O Problema Tempo Real consiste então em especificar, verificar e implementar sistemas ou programas que, mesmo com recursos limitados, apresentam comportamentos previsíveis, atendendo as restrições temporais impostas pelo ambiente ou pelo usuário [Jos91]. Se consideramos esses aspectos de construção, tempo real pode ser visto inicialmente como um problema intrínseco de programação concorrente. Baseado então na maneira de tratar a concorrência surgiram duas abordagens diferentes amplamente discutidas na literatura nesses vinte últimos anos: a Abordagem Assíncrona1 e a Abordagem Síncrona.

A abordagem assíncrona cujo o entendimento mais usual foi introduzido por R. Milner [Mil80] trata a ocorrência e a percepção de eventos independentes numa ordem arbitrária mas não simultânea. As linguagens CSP, Ada e Real-Time Concurrent C seguem o paradigma definido nessa abordagem e podem ser consideradas como linguagens assíncronas. Essa abordagem visa uma descrição a mais exata possível de um sistema; para tal, baseia-se na observação durante a execução de todas as combinações de ocorrência de eventos (de forma não simultânea), conhecida como entrelaçamento de eventos (*"interleaving"*). Essa abordagem é considerada como orientada à implementação. Consequentemente, se torna necessário levar em conta na especificação e no decorrer do projeto, algumas características do suporte de software e de hardware das aplicações, o que fere eventuais exigências em termos de portabilidade. Por outro lado, a procura de uma descrição completa do comportamento e a introdução de considerações de implementação torna complexa a análise das propriedades do sistema por causa da necessidade de tratar com grande número de estados e do possível não determinismo introduzido. A abordagem assíncrona apresentada nesse livro é implementada usando linguagens e sistemas operacionais e está fundamentada no tratamento explícito da concorrência e do tempo de uma aplicação em tempo de execução; a questão do escalonamento de tempo real é o ponto principal do estudo da previsibilidade dos sistemas de tempo real.

A abordagem síncrona na forma introduzida em [Ber89] e [BeB91] tem o seu

[1] O termo *Abordagem Assíncrona* utilizado neste livro não é consenso na literatura de tempo real.

princípio básico na consideração que os cálculos e as comunicações não levam tempo. A abordagem síncrona se coloca num nível de abstração dos aspectos de implementação tal que o tempo é visto com granularidade suficientemente grossa para que essa hipótese seja verdadeira. A descrição do comportamento do sistema é nessa abordagem menos dependente das questões de implementação, o que é satisfatório do ponto de vista da portabilidade. A observação dos eventos nessa abordagem é cronológica, permitindo uma eventual simultaneidade entre eles. Nesta abordagem, a partir das suas premissas, a concorrência é resolvida sem o entrelaçamento de tarefas (*"interleaving"*) e o tempo não é tratado de maneira explícita. A abordagem síncrona é considerada como orientada ao comportamento de uma aplicação e a sua verificação. Por se situar num nível de abstração maior que no caso da abordagem anterior, a abordagem síncrona facilita a especificação e a análise das propriedades de sistemas de tempo real. Diversas linguagens como Esterel, Statecharts, Signal, Lustre – chamadas de linguagens síncronas -- seguem o paradigma definido nessa abordagem.

Nesse livro, apresentamos soluções que seguem essas duas abordagens para tratar o problema de tempo real. A abordagem assíncrona, dependente das ferramentas de implementação e que trata explicitamente o tempo e a concorrência têm introduzidos os conhecimentos necessários para o seu uso nos capítulos 2 e 3. A abordagem síncrona baseada na hipótese de sincronismo é discutida no capítulo 4. Exemplos ilustrativos permitindo entender essas duas abordagens, mostrando suas vantagens e limitações são mostrados no capítulo 5. É objetivo nesse livro apresentar os tipos de aplicações nas quais cada uma dessas abordagens é mais adaptada.