UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ESCOLA DE ENGENHARIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Proposta de Ferramenta para Validação Temporal em Barramentos de Campo

por

RAFAEL WILD

Dissertação submetida à avaliação, como requisito parcial para a obtenção do grau de Mestre

Prof. Dr. Carlos Eduardo Pereira

Orientador

Porto Alegre, maio de 2000.

PROPOSTA DE FERRAMENTA PARA VALIDAÇÃO TEMPORAL EM BARRAMENTOS DE CAMPO

RAFAEL WILD

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Ing. Carlos Eduardo Pereira, UFRGS,

Dr. -Ing., Universidade de Stuttgart, Alemanha.

Banca Examinadora:

Prof. Dr. Altamiro Amadeu Susin, UFRGS.

Doutor em Informática, INPG - Grenoble, França.

Prof. Dr. João Manuel Gomes da Silva Junior, UFRGS.

Doutor em Engenharia de Controle, UPS/LAAS - Toulouse, França.

Profa. Lúcia Regina Rodrigues Horta Franco, EFEI.

Doutora em Engenharia, Escola Politécnica/USP – São Paulo.

Coordenador do PPGEE: _____

Prof. Dr. Altamiro Amadeu Suzim

Porto Alegre, maio de 2000.

Dedico este trabalho à memória de Luciana.

Agradecimentos

Quero agradecer a meus pais, meus irmãos e a toda minha família pelo incentivo fundamental para que este trabalho fosse realizado. Também ao irmão Felipe e aos amigos Alex, Maria Carmen e Mário Guimarães pela companhia em tantos momentos deste trabalho.

Agradeço especialmente ao meu orientador, Prof. Carlos Eduardo Pereira, pela confiança depositada e pela dedicação no acompanhamento desta dissertação. Agradeço ao bolsista João Rubens pela ajuda prestada na montagem da planta Fieldbus usada como estudo de caso. Também gostaria de agradecer a colaboração dos integrantes do Laboratório de Automação e colegas Ronaldo Husemann, Leandro Becker, Wilson Pardi Jr., Carlos Mitidieri, Leandro Tibola, Isabel Fernandes, Aline Flores, Marcelo Göetz e Sérgio Suess. Agradeço também a todas as pessoas que, apesar de não mencionadas, contribuíram para este trabalho.

Resumo

Sistemas de controle industriais precisam atender a requisitos temporais para garantir seu correto funcionamento, sendo por isto considerados sistemas tempo-real. Quando tais sistemas são distribuídos, tais como redes de sensores, atuadores e controladores interligados por barramentos de campo, a comunicação desempenha um papel importante no comportamento temporal. Este trabalho propõe uma ferramenta para validar o comportamento temporal da comunicação em um protocolo de barramento de campo, o *Foundation Fieldbus*. A proposta inclui a especificação de requisitos e a visualização da validação. Pretende-se auxiliar a compreensão do comportamento temporal e possibilitar a depuração de sistemas tempo-real distribuídos. O sistema desenvolvido encontra-se operacional e foi validado em diversos estudos de caso, os quais são descritos no presente trabalho.

Abstract

Industrial control systems must fulfill strict timing requirements in order to assure they work correctly. Such systems are called real-time systems. When such systems are distributed, as fieldbuses are, communication plays an important role in the temporal behavior. This work proposes a tool for the validation of temporal behavior of a fieldbus protocol, the *Foundation Fieldbus*. This proposal includes timing requirements specification, a validation engine, and visualization possibilities. The tool aims to provide support to designers in order to achieve a better understanding of temporal behavior as well as in the debugging of distributed real-time systems. The developed tool has been validated through different case studies that are described in this work.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

LAS	*Link Active Scheduler*, Escalonador Ativo de Enlace
PID	Algoritmo de controle Proporcional Integral Derivativo
CLP	Controlador Lógico Programável
OSI	*Open Systems Interconnection*, modelo para interconexão de sistemas abertos
ISO	*International Organization for Standardization*, Organização Internacional de Normalização
FIP	*Factory Instrumentation Protocol*, um protocolo de barramento industrial
PROFIBUS	*Process Fieldbus*, um protocolo de barramento industrial

1. Introdução

1.1 Automação Industrial e Distribuição do Processamento

Modernas arquiteturas distribuídas de automação industrial são caracterizadas por redes de dispositivos de campo, usualmente conectadas através de um barramento de comunicação, chamado de barramento de campo (*fieldbus*). Em sistemas de controle e automação deste tipo, a instrumentação é constituída por dispositivos sensores e atuadores de diversas naturezas, capazes de executar processamento local e comunicar-se entre si. Redução dos custos de cabeamento, aumento na qualidade e na quantidade de informação oriunda da planta, flexibilidade de reconfiguração, possibilidade de estratégias de controle mais complexas, estão entre algumas vantagens proporcionadas por esta abordagem. Os benefícios tecnológicos e funcionais obtidos através da utilização de barramentos de campo fazem desta solução o atual estado da arte em instrumentação e controle de plantas industriais.

Barramentos de campo inserem-se em um contexto que privilegia a distribuição do processamento como opção de arquitetura. A introdução da instrumentação digital permitiu melhoras revolucionárias em termos de condicionamento de sinais, transmissão de grandezas e facilidade e flexibilidade de instalação e configuração. Ao mesmo tempo, a utilização de controle digital para processos trouxe consigo um maior poder de controle e acesso à informação sobre a planta. Dois fatores, a disponibilidade, *dentro do instrumento*, de capacidade computacional significativa, e a existência de uma gama de informações produzida pelo instrumento digital além da simples grandeza medida, levaram à proposta de um novo modelo para a conexão e operação da instrumentação: o barramento de campo. Os dispositivos comunicam-se entre si, ligados por um

barramento comum de dados, digital e serial, e são capazes de processar estratégias de controle implementadas. Em função de suas características inovadoras no âmbito da automação industrial e da natureza potencialmente crítica de suas aplicações, sistemas baseados em barramentos de campo são interessantes objetos de estudo visando sua validação.

1.2 Automação Industrial e Sistemas Tempo-Real

Tipicamente, em sistemas computacionais de automação e controle industriais, as especificações requerem que a atuação sobre o sistema controlado seja correto tanto do ponto de vista lógico como também do ponto de vista temporal. Os valores de entrada devem ter sido obtidos em um tempo determinado para que sejam válidos, e o valor de saída deve ser calculado (a partir dos valores de entrada e valores de variáveis de estado) e passado aos atuadores dentro de um prazo estabelecido para que seja útil. Sistemas em que é fundamental a garantia de que estes requisitos temporais serão atendidos são chamados **sistemas tempo-real**. Requisitos de tempo podem muitas vezes ser críticos em sistemas tempo-real de automação e controle em chão-de-fábrica, no sentido de que o não atendimento a estes requisitos pode resultar em catástrofe (danos materiais e a seres humanos ou mesmo morte).

Embora exista uma idéia difundida que o atendimento a requisitos de tempo seja questão apenas de melhorar suficientemente a capacidade computacional do sistema, o trabalho de pesquisa em tempo real provou que isto não é verdade [STA88]: “estruturas computacionais apropriadas para sistemas que requerem tempo de resposta limitado diferem fundamentalmente daquelas para sistemas que requerem alta taxa computacional”[SON95]. Sistemas de controle e automação tempo-real possuem ainda, normalmente, outras características [BUR90], tais como graus diversos de tamanho e complexidade, necessidade de confiabilidade e segurança, controle concorrente de subsistemas diversos, e também interação com interfaces de hardware. Por este motivo, sistemas tempo-real necessitam de suporte adequado, em nível de hardware, projeto de software, sistema operacional, linguagem de programação, e verificação em tempo de execução.

Redes de automação e controle apresentam-se em um contexto de sistemas tempo-real distribuídos. Neste caso, o comportamento das estratégias de controle adotadas depende não apenas do comportamento do processamento interno nos

dispositivos, mas passa a ser função também da comunicação entre os dispositivos. Existe a necessidade de que, além do processamento e atuação, a comunicação entre dispositivos que fazem parte do sistema e se distribuem no espaço atenda aos requisitos de tempo. Neste sentido, as propostas de barramentos de campo exigem uma concepção integrada, em que todos os componentes de implementação da comunicação (em software ou hardware) devem prover suporte tempo-real, para que o sistema global possa garantir tempo-real. Em particular, a arquitetura para a comunicação no *Foundation Fieldbus* [FIE98], abordado no trabalho presente, a qual é baseada em camadas hierárquicas de acordo com o modelo OSI da ISO [SOA95], prevê suporte a tempo-real na definição de suas camadas de comunicação.

1.3 Monitoração em Sistemas Tempo-Real Distribuídos

Sistemas tempo-real de controle e automação devem interagir com a planta sob controle, e por este motivo são fortemente acoplados ao processo executando na planta industrial. Seu projeto baseia-se em um conjunto de suposições sobre o sistema e o ambiente que o cerca [JAH95]. Tais suposições dizem respeito, por exemplo, ao tempo que uma determinada comunicação dentro do sistema levará para ser executada; ou a freqüência com que algum acontecimento externo possivelmente ocorrerá. O comportamento temporal da estratégia de controle depende, portanto, de fatores externos ao âmbito estrito do sistema controlador, ou seja, geralmente não pode ser determinado completamente antes do tempo de execução e da real interação com o ambiente. Adicionalmente, por mais cuidadosa que seja a modelagem, podem ocorrer situações em que esta não seja mais válida. A experiência tem demonstrado que sistemas complexos dificilmente podem ser completamente verificados de maneira formal, e que é muito difícil assegurar que uma determinada implementação é livre de erros, ou que surja uma situação não prevista inicialmente, colocando em risco assim a correção temporal do sistema. Esta é a principal motivação para a monitoração de aplicações tempo-real (ver por exemplo [CHO91] e [PLA84]).

A monitoração consiste na observação dos eventos do sistema, tornando possível validar seu comportamento temporal e o cumprimento de requisitos temporais em tempo de execução. Possibilidades abertas pela monitoração em sistemas tempo-real e distribuídos são, por exemplo:

- avaliação do desempenho da aplicação;

- identificação da possibilidade de indisponibilidade de algum recurso;
- requisitos temporais sob risco de não atendimento pela ocorrência de alguma condição não prevista.

1.4 Monitoração em Barramentos de Campo

Barramentos de campo, inserindo-se em um contexto de sistemas tempo-real e distribuídos, compartilham de sua problemática relativa à validação do comportamento em tempo de execução. Em um sistema de controle baseado em barramento de campo, existem múltiplos instrumentos gerando informação, que deve ser comunicada através de um barramento de dados único e compartilhado. Em um tal ambiente surge a necessidade de **escalonar** a comunicação, isto é, coordenar a ocupação do barramento pelas mensagens de cada instrumento, sem que haja superposição de mensagens e de maneira que cada mensagem cumpra os requisitos de tempo impostos pela sua utilização.

As ferramentas disponíveis para configuração em redes de campo normalmente apóiam a validação do sistema resultante apenas de forma restrita. Mecanismos apresentados por estas ferramentas incluem diagramas em nível de usuário, com indicação de falha de conexões de comunicação entre instrumentos através de cor diferenciada; ou então, listagens de mensagens com registro de tempo (de instante de comunicação). A estas propostas, apesar da utilidade da informação apresentada, falta entretanto um maior refinamento em termos de validação de requisitos temporais e visualização do comportamento da comunicação em tempo de execução.

A importância da tecnologia de barramentos de campo para o cenário da automação e controle industrial e a carência de ferramentas de apoio à monitoração e validação de requisitos temporais para estes sistemas, levou a proposta de um trabalho que abordasse estas questões. A ferramenta desenvolvida para tal validação pode se tornar uma contribuição para observar o desempenho tempo-real de qualquer configuração em barramentos de campo; e os resultados de tal validação podem esclarecer algumas condições dentro das quais o desempenho tempo real de uma solução *Foundation Fieldbus*, ou de outros protocolos de barramentos de campo, é aceitável ou não, e em que termos é aceitável ou não. A informação gerada pode ser útil tanto para o engenheiro de aplicação avaliar sistemas já implementados, quanto para auxiliar no projeto de novos sistemas.

O presente trabalho está inserido em um projeto que resulta de uma parceria entre a sub-rede Controle de Processos - Fieldbus do Projeto RECOPE - Redes Cooperativas de Pesquisa, um programa multiinstitucional de pesquisa com recursos FINEP. Conta ainda com o apoio da empresa Smar Equipamentos Industriais, de Sertãozinho, SP, destacado fabricante na área de protocolo e dispositivos *Foundation Fieldbus*.

1.5 Objetivos do Trabalho

Fieldbus é uma tecnologia emergente, que apresenta as vantagens de um barramento de campo aliadas a uma padronização internacional, que o torna aberto, acessível e interoperável. Para muitos dos sistemas de automação e controle que requerem tal tecnologia, tempo-real é uma exigência fundamental de projeto. Este trabalho procura investigar o desempenho tempo-real do Fieldbus, não somente em termos de violações de requisitos de tempo, mas também em termos de características temporais do atendimento a estes requisitos. Propõe para tanto uma ferramenta para monitoração, validação e visualização da comunicação no barramento. O validador desenvolvido tem a função de, tendo por entrada requisitos temporais formalizados, e um histórico registrado de ocorrências de eventos do sistema, gerar informação a respeito da comparação deste com aqueles: concordância e variações de temporização em tempo de execução. A verificação em princípio ocorre em tempo posterior à execução do sistema (utilizando o histórico gerado).

A ferramenta permite revelar alguns dados significativos a respeito do desempenho tempo-real do sistema: variação no comportamento da comunicação em relação a diferentes estratégias de escalonamento de mensagens; carga periódica e não-periódica; variação na periodicidade de mensagens periódicas; tamanho do ciclo configurado, e o cumprimento de requisitos fim-a-fim (aqueles estabelecidos em função de entradas e saídas externas [RYU98]). Espera-se assim, com os resultados deste trabalho, contribuir para a pesquisa do desempenho tempo real dos barramentos de campo, em especial *Foundation Fieldbus*. Adicionalmente, o conhecimento adquirido com o uso do validador pode ser incorporado a estratégias mais elaboradas de escalonamento de mensagens que venham a atender os requisitos temporais especificados. É importante ressaltar que, apesar da presente implementação estar

vinculada ao protocolo *Foundation* Fieldbus, a proposta é genérica, podendo ser adaptada a outros protocolos.

Esta dissertação está estruturada da seguinte forma: o capítulo 2 apresenta uma análise das propostas em monitoração de sistemas tempo-real e distribuídos; o capítulo 3 contém uma revisão sobre a tecnologia *Foundation Fieldbus*; o capítulo 4 apresenta a proposta da ferramenta para validação, e o capítulo 5 contém uma descrição de sua implementação e características finais. O capítulo 6 apresenta alguns estudos de caso demonstrando a utilização da proposta; e por fim, o capítulo 7 contém conclusões sobre a presente dissertação e direções para futuros trabalhos.

2. Motivação sobre o barramento de campo Foundation Fieldbus

Barramentos de campo constituem-se em um moderno paradigma para sistemas de controle e automação industrial. Tal paradigma é caracterizado pela utilização de dispositivos (instrumentos) de campo, com capacidade local de processamento, e que comunicam-se entre si. Combinam-se assim benefícios oriundos da instrumentação digital, da distribuição do processamento, e da comunicação de dados entre dispositivos. Barramentos de campo, enquanto componentes de sistemas de controle e automação industrial, são considerados sistemas tempo-real, significando que devem atender a determinados requisitos temporais, além de garantir a correção lógica do processamento. Em tais sistemas de natureza distribuída, a comunicação desempenha um papel crítico no comportamento temporal, e, por conseqüência, no atendimento a requisitos temporais. Para validar ou depurar sistemas tempo-real distribuídos, em especial a comunicação em um barramento de campo, é importante a disponibilidade de suporte à monitoração e validação temporal. Este suporte destina-se à validação temporal do sistema como um todo, e também de seus componentes (tais como o escalonador e o método de acesso ao meio da comunicação, e a implementação do processamento nos dispositivos).

Entre as diversas propostas existentes para uma normatização na área de barramentos de campo, destaca-se o *Foundation Fieldbus*, oriundo de uma organização chamada *Fieldbus Foundation* ([FIE98], [FIE99]. As características de padronização "aberta" (não-proprietária) para interoperabilidade, proposição de um nível de interface de usuário normatizado e abordagem a atendimento a requisitos temporais, fazem do

Foundation Fieldbus uma proposta de interesse para estudo. Serão apresentados tópicos relevantes para a compreensão da estrutura funcional do *Foundation Fieldbus*, em especial da comunicação, que será objeto da monitoração e validação propostas pelo presente trabalho.

2.1 Barramentos de Campo e Foundation Fieldbus

A instrumentação digital introduziu características importantes na utilização de instrumentos de campo. O pré-processamento da medição, através do uso de microcontroladores incorporados aos dispositivos de campo, tornou as medidas mais confiáveis e precisas, a comunicação digital tornou a informação mais imune a ruídos, e o poder computacional destinado a processar os dados de sensores e atuadores possibilitou a implementação de estratégias de controle mais refinadas, seguras, e velozes. A princípio, a arquitetura escolhida para abrigar a instrumentação digital foi centralizada: um computador central recebendo os dados dos sensores, processando-os e na seqüência enviando os resultados para os respectivos atuadores. Esta arquitetura de fato exige uma alta capacidade computacional da unidade de processamento central. Adicionalmente, todo instrumento possui uma ligação física ao computador central e a reconfiguração ou manutenção de apenas um componente (dispositivo ou seção de programa) freqüentemente exige parada de todo o sistema.

Neste contexto de novas possibilidades e algumas limitações, a capacidade intrínseca de processamento computacional dos dispositivos de campo tornou-se a chave para uma nova proposta em instrumentação de campo: a distribuição do processamento. Isto significa que o programa de controle pode ser fracionado em blocos menores, e computado dentro dos próprios dispositivos de campo. Tais dispositivos são capazes de comunicar-se entre si, tornando possível a configuração ou reprogramação remota do dispositivo, a transmissão da grandeza medida em unidades de engenharia e a transmissão de informações adicionais importantes para diagnóstico do próprio dispositivo. As variáveis de entrada e saída, incluindo as intermediárias, são passadas entre os dispositivos. Dentre as vantagens advindas do uso de arquiteturas distribuídas, nas quais sensores e atuadores com capacidade local de processamento podem comunicar-se através de protocolos industriais, destacam-se:

- melhor desempenho pelo processamento paralelo (ou equivalentemente, a necessidade de unidades de processamento com menos capacidade computacional);
- maior disponibilidade pela possibilidade de diversas fontes para a mesma informação;
- maior confiabilidade, por que o sistema pode se recuperar, ao menos parcialmente, de diversos tipos de falhas;
- e maior flexibilidade para ampliação ou reconfiguração, por que a manutenção do sistema pode se limitar aos componentes, não atingindo o todo.

Uma das arquiteturas propostas para a distribuição da instrumentação, e sem dúvida a mais difundida, foi o barramento de campo – uma rede de comunicação de dados digital, bidirecional, multiponto, serial, associada a protocolos de enlace de dados e de interface com o usuário, utilizada para ligar entre si instrumentos de campo como controladores, sensores e atuadores [SIL95). Conforme mencionado, a capacidade de processamento local e de comunicação digital proporciona a distribuição do controle e geração da informação, com objetivo de atingir estratégias mais refinadas de controle.

Arquiteturas mais simplificadas, conhecidas como Entrada e Saída Distribuídos, ou como *Sensorbuses*, utilizam a possibilidade de comunicação apenas para passar a informação ao controle central – como se fossem placas de entradas e saídas remotas de um controlador lógico programável (CLP). Apesar de não possuírem todas as funcionalidades propostas para os barramentos de campo, e de constituirem-se normalmente de protocolos proprietários, são capazes de atingir alguns benefícios importantes: redução dos custos de instalação e aquisição remota de dados.

Entretanto, para abordagens mais elaboradas, um passo importante é a padronização: instrumentos de diversas procedências devem comunicar-se entre si, garantindo a interoperabilidade que é característica, por exemplo, do antigo padrão 4-20mA. Um *fieldbus* padrão garantiria a interconexão de equipamentos de qualquer fabricante entre si, incluindo a possibilidade de redundância entre instrumentos diferentes mas com as mesmas funcionalidades. A padronização *fieldbus* passou por diversos estágios, desde as normas alemã e francesa, que resultaram nos protocolos Profibus e FIP, respectivamente, até a proposta de padrão internacional patrocinado pela

ISA e IEC (ISA SP-50 e IEC61158). Esta está ainda em fase de normatização - alguns de seus documentos já foram aprovados e são norma internacional, outros são ainda rascunho. Este trabalho aborda o protocolo *Foundation Fieldbus* [FIE98], uma tecnologia desenvolvida por uma organização independente, a *Fieldbus Foundation*, e que é baseada nos trabalhos da ISA e IEC.

2.2 Escalonamento de Mensagens no Foundation Fieldbus

A possibilidade de dispositivos de campo realizarem localmente tanto processamento de dados como tomadas de decisão constitui-se em conceito chave em tecnologias de barramento de campo tais como o *Foundation Fieldbus*. Estas características tecnológicas levam a arquitetura de controle à descentralização e à distribuição. Em um barramento de campo com comunicação entre os dispositivos, o controle de um processo pode ser feito localmente: o valor de entrada medido por um dispositivo é passado diretamente, pelo barramento, ao dispositivo atuador, o qual pode incorporar as funções de controle – ver Figura 2.1. Em um ambiente assim, mensagens são eventos relevantes a serem monitorados: a comunicação é coordenada em função da ocupação do barramento por cada mensagem de dispositivo, e torna-se importante validar o comportamento temporal do sistema, de tal forma que cada mensagem cumpra seus requisitos de tempo.

A tecnologia *Foundation Fieldbus* disponibiliza um conjunto de blocos funcionais padronizados, que são conjuntos implementados de dados e algoritmos, com o objetivo de prover funcionalidades para a aplicação, seja em caráter isolado ou ligados a outros blocos funcionais. São exemplos controladores PID, blocos de entrada e saída, e blocos de transdutor. Ligações entre blocos funcionais significam o intercâmbio de informações entre os mesmos. No caso de os blocos conectados residirem em diferentes dispositivos, o que é comum em aplicações em barramentos de campo, mensagens são passadas pelo barramento.

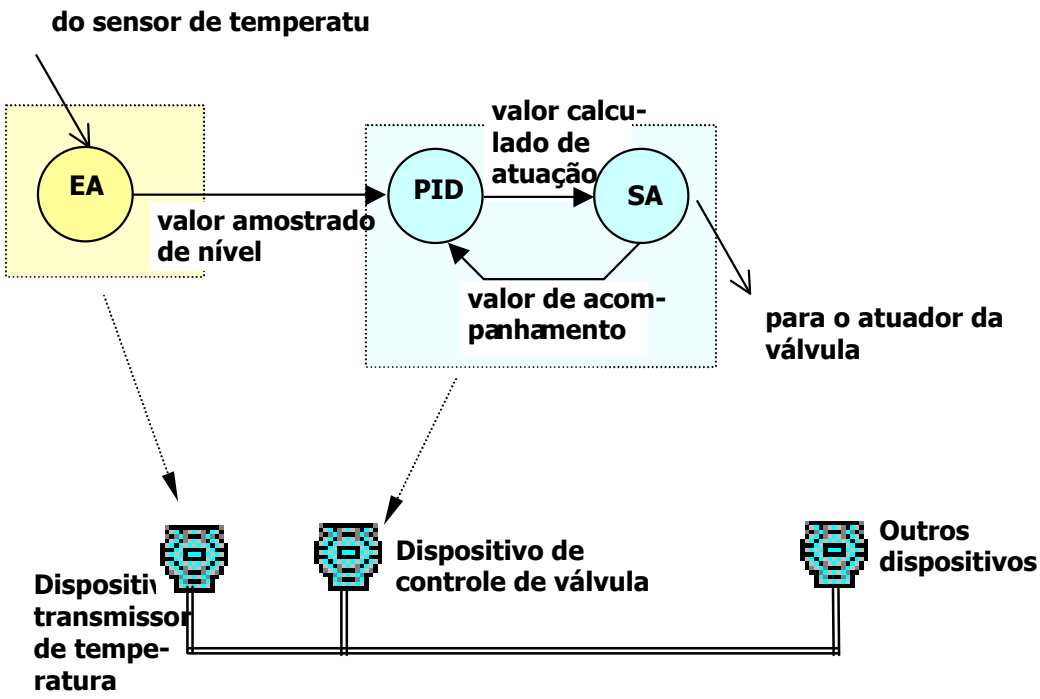

Figura 2.1– Esquema de Aplicação *Foundation Fieldbus*

A arquitetura pode ser melhor compreendida através do exemplo na Figura 2.1. O exemplo apresenta uma aplicação (estratégia de controle) em que uma válvula deve ter sua abertura controlada em função da temperatura medida em um ponto do processo. São necessários para implementar esta estratégia um dispositivo do tipo “transmissor de temperatura” e um “atuador de válvula”. Estando ligados em rede, estes dispositivos implementam esta estratégia através de blocos funcionais. Um bloco funcional “Entrada Analógica” (EA) disponibiliza para o sistema a leitura do sensor do dispositivo de temperatura. Uma “Saída Analógica” (SA) transfere para o atuador o valor de atuação adequado, e um controlador PID calcula o valor de atuação a partir da leitura do sensor. Estes blocos são pré-programados, sendo incluídos na aplicação e depois interligados para que passem a informação de um para outro, através de mensagens no barramento. Por fim, estes blocos são alocados nos dispositivos onde os algoritmos serão efetivamente computados. No exemplo apresentado, o bloco PID é alocado no dispositivo de controle da válvula. É importante observar que o bloco PID poderia também ser alocado para rodar no transmissor de temperatura. Esta configuração alternativa, apesar de não interferir na funcionalidade da aplicação, pode influenciar o

comportamento temporal da comunicação, pelo fato de alterar as mensagens intercambiadas no barramento e suas temporizações.

O protocolo *Foundation Fieldbus* define dois tipos de comunicação (mensagens no barramento): periódica e aperiódica. Os dados que possuem algum tipo de restrição temporal, tais como aqueles utilizados em estratégias de controle, são comunicados através de mensagens periódicas. No exemplo, uma mensagem periódica seria o valor de temperatura que é passado do bloco funcional EA no transmissor de temperatura para o bloco PID no atuador de válvula. Todas mensagens periódicas devem ser escalonadas em tempo de configuração. Isto é, informações temporais sobre estas mensagens, como duração da mensagem e tempo de processamento do dispositivo para emitir tal mensagem, devem ser conhecidas de antemão. Já dados que não possuem restrições temporais são comunicados através de mensagens não-periódicas. Tais dados podem ser informações do processo passado para supervisão e visualização. Mensagens não-periódicas são escalonadas para comunicação durante o tempo de execução, utilizando o recurso do barramento apenas durante os intervalos em que não há mensagens periódicas escalonadas.

A gerência da comunicação e de acesso ao meio é responsabilidade de um dispositivo que, além de sua função normal, efetua o arbitramento do barramento. Esta entidade é chamada de "Escalonador Ativo de Conexão", conhecido pela sigla LAS (do inglês "*Link Active Scheduler*"), e possui uma tabela com todas as mensagens escalonadas para cada dispositivo[1]. Mensagens periódicas, tais como o valor de temperatura disponibilizado pelo bloco EA, são publicadas no barramento, para que todos os dispositivos que utilizarão estes dados possam acessá-las. Isto ocorre de acordo com o seguinte procedimento: (i) o dispositivo LAS, ao chegar o momento configurado de transmissão, envia ao transmissor a mensagem *Invoca Dados* ("*Compel Data*", CD); (ii) em resposta a esta mensagem o transmissor publica o valor de saída do bloco EA no barramento, utilizando a mensagem *Publicação de Dados* ("*Data*", DT).

Durante intervalos em que não há mensagens periódicas escalonadas, o LAS circula entre os dispositivos presentes no barramento uma passagem de permissão ("*Pass Token*", PT), permitindo aos dispositivos que possuam mensagens não periódicas para transmitissão que acessem o barramento. As mensagens aperiódicas são comunicadas de acordo com o seguinte procedimento: (i) o dispositivo LAS, quando há um intervalo de tempo sem mensagens periódicas escalonadas, envia ao transmissor a mensagem *Passagem de Permissão* (PT); (ii) em

[1] Este modelo de comunicação é chamado de "produtor-consumidor"

resposta a esta mensagem o transmissor publica os dados configurados para transmissão, utilizando uma ou mais mensagens *Publicação de Dados* (DT); (iii) após a mensagem DT, ou se não houver nenhum dado a ser transmitido, o transmissor envia ao LAS a mensagem *Devolução de Permissão (*"*Return Token*", RT).

Além disso, o LAS executa outras tarefas de gerenciamento da rede, tais como manutenção da lista de dispositivos conectados, ativação de novos dispositivos, e sincronização de relógios dos dispositivos. Devido a sua importância, no caso de falha do dispositivo executando o LAS, automaticamente outro dispositivo previamente assinalado torna-se responsável pelo escalonamento das mensagens. A gerência dos dispositivos conectados ocorre da seguinte forma: existe uma lista de dispositivos correntemente ativos, para os quais o LAS emite mensagens PT. De tempos em tempos, o LAS emite mensagens de *Teste de Endereço* ("*Probe Node*:", PN) a dispositivos que não estejam nesta lista. Caso este dispositivo tenha sido adicionado à rede, responderá com uma mensagem de *Resposta ao Teste* ("*Probe Response*", PR), e será adicionado à lista. Também a intervalos máximos determinados é emitida uma mensagem de sincronização de barramento, para que sejam acertados os relógios internos dos dispositivos de acordo com o relógio do mestre.

A seqüência de mensagens a ser publicada é definida pelo algoritmo do dispositivo LAS, conforme Figura 2.2 [FIE98]. As mensagens periódicas são publicadas conforme configuração do escalonamento no tempo; após a publicação de mensagem periódica, caso haja intervalo suficiente, a passagem de permissão aperiódica é passada a cada dispositivo por sua vez até que seja tempo de outra mensagem periódica.

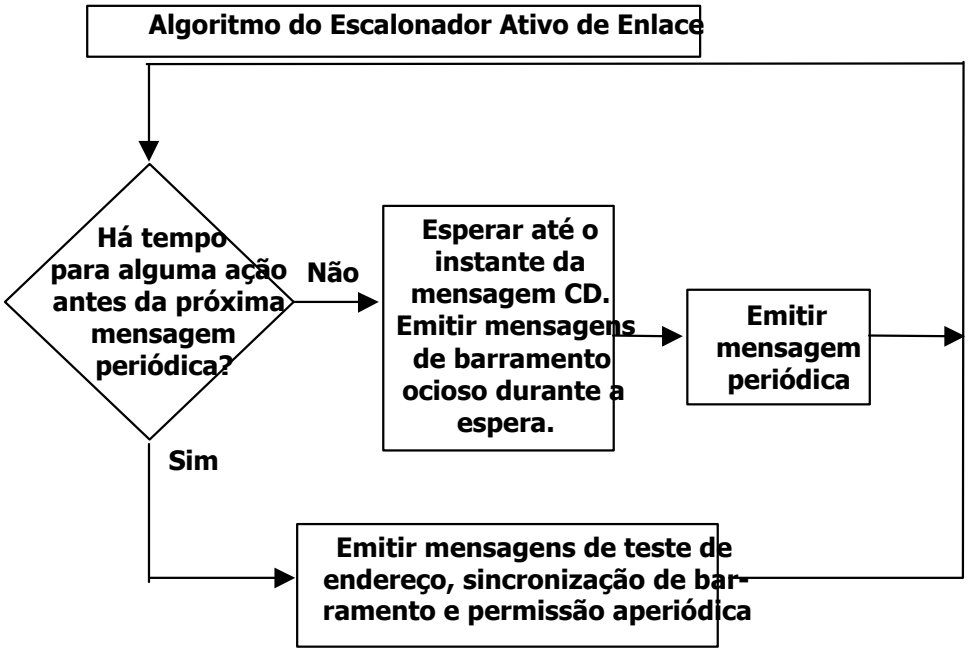

Figura 2.2 – Algoritmo de escalonamento do LAS

Mnemônico Mensagem	Quadro de controle	Função
CD2	1011 LFPP	LAS requisita dados periódicos do dispositivo
DT1	1101 LFPP	Dispositivo responde com dados a passagem de permissão aperiódica
DT3	0101 LFPP	Dispositivo responde com dados a requisição de dados periódicos
TD	0001 0F01	Distribuição de tempo – sincronização de barramento
PN	0010 0110	Teste de endereço
PR	0010 0111	Resposta a teste de endereço
PT	0011 0FPP	Passagem de permissão aperiódica
RT	0011 0100	Devolução de permissão aperiódica
Idle	0001 0FPP	Barramento ocioso
Legenda: L: tamanho dos endereços associados (0=2 bytes, 1=4 bytes). F: uso final de permissão, indicando que a seqüência de transmissão de dados em andamento deve ser encerrada. PP: prioridade da mensagem, valores entre 01 (maior) e 11 (menor).		

Tabela 2.1– Principais mensagens Fieldbus [FIE99]

A Tabela 2.1 contém um resumo das principais mensagens *Foundation Fieldbus* abordadas neste trabalho, com o valor de seu quadro de controle (a parte da mensagem que identifica sua função) e sua respectiva funcionalidade.

O Escalonador de Conexão Ativa é um conceito fundamental da tecnologia Foundation Fieldbus, sendo um passo importante na direção de um comportamento tempo-real determinístico e da superação de deficiências de tecnologias de barramento de campo relacionadas a requisitos temporais, alta carga de processamento, e escalonabilidade da comunicação. Como o acesso ao barramento é arbitrado pelo LAS seguindo uma estratégia de escalonamento definida pelo escalonador, pode-se garantir o cumprimento dos prazos para as mensagens periódicas.

Existem comercialmente disponíveis ferramentas para *Foundation Fieldbus* tais como NI-Fbus (National Instruments), Delta-V (Fischer Rosemount) e Syscon (Smar). Estas ferramentas são destinadas primariamente a configuração de redes *fieldbus*. Apesar de possuirem funcionalidades destinadas ao exame do barramento, tais como captura e interpretação de mensagens e registro de tempo, não possuem tratamento para requisitos temporais.

3. Estado da Arte em Monitoração e Visualização de Sistemas Tempo-Real

Monitoração e validação em sistemas tempo-real tem merecido diversas abordagens, em função da importância para o desenvolvimento de tais sistemas, e da multiplicidade de arquiteturas alvo existentes. Serão revisadas algumas das propostas clássicas, juntamente com conceitos associados a cada proposta, em cada uma das principais possibilidades de monitoração: por programa, por equipamento, e híbridas.

3.1 Monitoração e Validação em Sistemas Tempo-Real Distribuídos

Em um sentido amplo, monitoração significa obter informação de um sistema em tempo de execução usualmente não disponível através da análise estática do sistema com o fim de identificar eventos relevantes e sua ordenação relativa ou instante de ocorrência. O objetivo da monitoração é tornar possível uma validação do comportamento em tempo de execução em relação a determinados requisitos, com fins de teste e depuração do sistema, além de otimização e mesmo escalonamento dinâmico de tarefas. Para tanto, a monitoração abrange a obtenção e o registro de informação acerca dos eventos, o processamento da informação avaliando requisitos de desempenho e comportamento, e a visualização dos resultados.

A maneira de coletar os eventos relevantes do sistema alvo da monitoração pode ser baseada em programa (*software*), baseada em equipamento (*hardware*), ou ainda híbrida [TSA96]. Embora seu objetivo seja o mesmo (coletar informação sobre os eventos do sistema), cada maneira possui características distintas em relação a custo de implementação, forma de utilização, e influência sobre o sistema. O emprego de uma ou outra maneira depende do sistema alvo, das possibilidades de implementação e até mesmo dos objetivos propostos para a monitoração.

3.1.1 Monitoração por Código (Instrumentação de Código)

A abordagem baseada em programa relaciona-se a inserção de código adicional no programa tempo-real original (referida como “instrumentação de código”). O código é adicionado em locais-chave do programa, como por exemplo pontos de entrada ou retorno de função, e o registro do evento ocorre pela execução da seção de código. Dados relacionados são armazenados, identificando o evento juntamente com informações adicionais (o instante de ocorrência do evento, parâmetros passados para a função, valores de retorno).

As duas principais características desta abordagem são a flexibilidade (não há equipamento adicional, alterações atingem apenas linhas de código) e a interferência no funcionamento do sistema (já que a monitoração utiliza a capacidade computacional do sistema alvo). Enquanto que a primeira característica torna esta abordagem atraente para sistemas em que a inclusão de equipamento dedicado a monitoração seria difícil ou cara, a segunda torna necessário um cuidado especial para não comprometer o desempenho e para controlar e mesmo prever o nível de perturbação do comportamento do sistema.

A instrumentação pode ser acrescentada ao código da aplicação ou ao sistema operacional. O primeiro caso possibilita a monitoração no nível de processo, disponibilizando eventos como chamadas e retornos de função, e estruturas de dados da aplicação. Este nível de detalhe é útil para depuração dentro de processos da aplicação. Já o segundo caso relaciona-se à monitoração em nível de sistema, sendo coletados eventos tais como mudança de estado do processo, interrupções externas, operações de entrada e saída, e comunicação entre processos, juntamente com parâmetros importantes

(por exemplo, identidade dos processos envolvidos, registro de tempo, valores em variáveis alteradas). Monitoração neste nível é direcionada ao comportamento global da aplicação e do sistema e seu desempenho. Alguns eventos, como chamadas a funções de sistema, podem ser disponibilizadas por monitoração em ambos os níveis.

Propostas de monitoração na aplicação podem ser encontradas em [CHO91] e [JOY87]. O trabalho [CHO91] apresenta uma arquitetura de monitoração, que inclui instrumentação de código, um verificador de atendimento de requisitos temporais, baseado na linguagem temporal RTL [JAH86], e duas possibilidades de monitoração dos requisitos: síncrona (validação embutida na instrumentação) e assíncrona (validação efetivada em um processo monitorador à parte). Já o artigo [JOY87] apresenta uma ferramenta de monitoração para um sistema de processos que interage apenas pela passagem de mensagens. A idéia é incluir uma versão de biblioteca de comunicação inter-processos "instrumentada", onde processos que utilizem tal versão coletem seus eventos e os comuniquem a um processo monitorador (chamado console). São gerados registros em texto, sem registro de tempo, e é apresentado um console gráfico que mostra estados de processos. Existe também um console estatístico e de *deadlock*, e um verificador de protocolo.

Monitoração no sistema operacional é utilizada em sistemas operacionais que possam ter o código de instrumentação incluído. Em [MIL86], é descrita uma arquitetura de monitoração para (o sistema operacional) BSD UNIX com suporte no núcleo do sistema operacional: ao ser criado um processo, uma conexão com um processo especial monitorador é criada. Chamadas a rotinas de sistema relacionadas a comunicação entre processos são registradas no processo monitorador através da conexão. O monitor inclui registro de tempo (local). Esta arquitetura é adotada com objetivos de transparência (o usuário não precisa construir a estrutura de monitoração) e eficiência (estrutura residindo no núcleo do sistema reduz a troca de contexto). Tokuda e outros, em [TOK88], descrevem um monitorador embutido para ARTS, um sistema operacional tempo-real para aplicações distribuídas. Sua arquitetura é composta por uma parte do núcleo do sistema, que identifica eventos, um processo em cada máquina responsável por coletar os

eventos, e um visualizador (que apresenta diagramas de Gantt e estatísticas), formado por um conjunto de processos em uma máquina não necessariamente ARTS.

3.1.2 Monitoração com Dispositivos

O uso de equipamento dedicado é outra abordagem para a questão de coletar eventos para monitoração, registrando os sinais presentes no barramento de dados do sistema alvo. O dispositivo pode ser incorporado construtivamente à arquitetura sendo monitorada, ou então simplesmente conectado a um barramento de dados disponível. O dispositivo monitorador registra eventos e dados associados, inclusive registro de tempo, sem consumir recursos computacionais da arquitetura alvo. No entanto, deve-se ressaltar que os eventos que podem ser coletados são apenas aqueles presentes no barramento monitorado.

A solução baseada em equipamento é mais cara por envolver projeto e construção de dispositivo específico, e por este mesmo motivo é menos flexível. Entretanto, pode constituir-se na única alternativa para sistemas críticos, por não interferir no desempenho do sistema monitorado. Também, no caso de barramento de dados disponível na arquitetura alvo, pode tornar-se a alternativa que envolve menos interferência, no sistema já configurado, para sua instalação.

Também é importante nesta abordagem a interpretação dos sinais monitorados, uma vez que os mesmos são coletados em um nível muito baixo de abstração. Assim sendo, é necessário uma elaboração deste conjunto de dados para que os eventos possam ser reconhecidos em seu devido nível de abstração. Tal processamento pode ser efetuado tanto pelo próprio equipamento monitorador, como por programa, em outro módulo de monitoração.

Um modelo de monitoração por equipamento é apresentado por [TSA90]. Trata-se de uma proposta de monitoração para uma arquitetura de processamento distribuída. Neste modelo, o monitor interfaceia com os dispositivos monitorados através do barramento interno de dados dos dispositivos, exigindo para sua implementação uma interface dedicada a cada tipo de barramento. O registro de eventos é realizado com base de tempo local. O trabalho de [LIU89], de maneira similar, descreve um modelo de

monitoração (e sua implementação) por equipamento. A arquitetura alvo é um sistema de multiprocessamento, com grupos de dispositivos processadores fortemente acoplados, conectados a outros grupos. Cada grupo possui um equipamento monitor dedicado, acoplado a seu barramento interno, executando coleta de eventos com registro de tempo. A implementação é dirigida ao barramento Multibus porém o modelo proposto é de natureza geral. No caso desta arquitetura, a monitoração dos sinais é passiva, porém os dados resultantes são comunicados através do mesmo canal (barramento) que registra os eventos, gerando desta maneira uma interferência no funcionamento normal do sistema.

O trabalho de [BRA89] expõe uma proposta algo diferente. Descreve um monitorador de hardware dedicado, construído embutido em um sistema processador paralelo (RP3). Novamente, os eventos são gerados em baixo nível de abstração: os diversos dispositivos integrantes do processador (controlador principal, unidade aritmética, memórias) sinalizam seus eventos, através de linhas de I/O especiais, implementadas no projeto do circuito integrado, para um controlador de monitoração. Ou seja, ao invés de os eventos serem detectados apenas pelos sinais correspondentes em um barramento comum de dados, os dispositivos geram ativamente os sinais identificadores, através de implementação física de linhas dedicadas no dispositivo. Os registros são acessíveis posteriormente através do barramento principal que interconecta os processadores. A proposta aborda apenas registro de eventos com registro de tempo.

3.1.3 Sistemas Híbridos de Monitoração

A instrumentação de código, enquanto estratégia de monitoração, tende a ser mais simples de usar e de menor custo em relação ao uso de equipamento dedicado. Por outro lado, equipamento dedicado permite reduzir ou eliminar a perturbação no sistema monitorado devido a carga adicional causada pela execução do código de instrumentação. Sistemas híbridos de monitoração buscam combinar os benefícios das abordagens baseadas em programa e em equipamento.

Em propostas híbridas de monitoração, normalmente uma instrumentação do código é inserida com o objetivo de gerar os eventos. Estes eventos são identificados por instruções de baixo nível, tais como acessos a posições específicas de memória ou portas

de entrada ou saída. Instruções deste tipo têm um retardo para sua execução muito pequeno, causando portanto uma interferência pequena no comportamento temporal do sistema, sem perder a flexibilidade de reconfiguração característica da instrumentação de código. A coleta e registro de eventos e instante de ocorrência fica a cargo de equipamento específico capaz de detectar e processar os sinais produzidos pelo código de instrumentação.

Em [HAB90], é abordado um sistema híbrido de monitoração para um ambiente experimental específico de computação distribuída, ambiente este composto de estações conectadas por uma rede. A implementação consiste de um dispositivo, embutido na estação, e conectado ao seu barramento de dados. Os eventos são gerados através de instrumentação do código, e são passados para o dispositivo através de acessos a uma faixa de endereços de memória. Os dispositivos processam os eventos, gerando registro de tempo e análise dos dados. Os dispositivos comunicam-se entre si através de uma rede dedicada, não sobrecarregando a comunicação das estações; os dados são passados por esta rede para uma estação central para processamento global e apresentação gráfica. Trata-se de um trabalho interessante do ponto de vista tempo-real por propor uma solução para a questão do tempo global. O ponto de vista de [DOD92], para um monitorador dedicado ao sistema distribuído Harts, é semelhante. O sistema é composto por um dispositivo dedicado dentro de cada estação, mais instrumentação no código para gerar os eventos. A proposta limita-se a registrar eventos de diversas naturezas, gerando um registro que torne possível reconstruir a ordenação da computação realizada e das interrupções externas recebidas. A instrumentação de código permanece após a depuração do sistema, sendo uma estratégia para tornar sua interferência temporal “previsível”. Como em outros sistemas, os dados recolhidos pelo monitor são passados pelo barramento interno para um subsistema encarregado de comunicá-los a uma estação de processamento ou visualização, através de uma rede de comunicação separada daquela que une as estações.

A proposta apresentada por [MIN90], por fim, discute dois sistemas de monitoração, implementados em VLSI (circuito integrado). O primeiro consiste de um monitorador por equipamento somente (sem interferência no sistema alvo), com pontas

de prova a serem conectadas às linhas de dados do sistema alvo, e com diversos blocos de amostragem, cada uma com um reconhecedor de padrões para disparar a amostra e conectado a um barramento VME para comunicar dados recebidos. Já o outro sistema é híbrido (causando assim uma “pequena” interferência); as amostras são disparadas por acessos a endereços específicos da memória, mapeados para o dispositivo. O trabalho ressalta a maior capacidade de discriminar dados (p.e., eventos oriundos de diferentes processos) do sistema híbrido, apesar da interferência causada no sistema alvo.

3.2 Pós-processamento e Apresentação dos Dados

Além da preocupação básica com a coleta, o pós-processamento e a apresentação dos dados desempenham um papel importante na monitoração de sistemas tempo-real. O pós-processamento almeja gerar informação adicional a partir da massa de dados, seja de natureza estatística, seja comparando os eventos com requisitos lógicos e funcionais aos quais devam obedecer. Já a apresentação dos dados é fundamental para resumir a enorme quantidade de dados recolhidos, e para separar e ressaltar aqueles aspectos mais importantes ou urgentes, tais como desvios do comportamento esperado para o sistema.

O objetivo da monitoração sobre um sistema tempo-real é tornar possível sua validação e auxiliar na sua depuração. Como os sistemas tempo-real interagem com o ambiente, simplesmente uma verificação formal da sua implementação não é suficiente para garantir sua adequação; e é muito difícil obter uma verificação deste tipo para sistemas mais complexos como por exemplo sistemas distribuídos. Os dados provenientes da monitoração devem portanto prover as necessidades da compreensão e depuração do próprio sistema, relacionadas a teste operacional, detecção e correção de falhas, verificação de segurança, análise de desempenho e otimização do sistema. Além disso, devem possibilitar o relacionamento das características do próprio sistema com o ambiente no qual está inserido, validando o modelo que foi utilizado para construir as estratégias de controle, e que é composto pelas grandezas de controle, variáveis intervenientes, as respectivas faixas de variação, a freqüência de ocorrência de eventos externos, etc.

Dentre os trabalhos de monitoração já citados, diversos abordam este processamento da informação coletada. Por exemplo, [TSA90] apresenta o pós-processamento dos registros de eventos gerados por sua proposta de arquitetura, destinado a relacionar o registro com o evento: associar os endereços virtuais aos processos que geraram o evento, identificar o tipo de evento ocorrido (criação de processo, mudança de estado), gerar e ordenar listas de ocorrências relacionados entre si (como por exemplo processos relacionados a determinada estratégia de controle). A partir deste processamento, a informação torna-se menos redundante e mais significativa, permitindo o acompanhamento dos acontecimentos de interesse no sistema.

Como já mencionado, sistemas tempo-real devem garantir o atendimento a requisitos temporais. Isto significa que o evento que é resposta do sistema a outro evento deve ocorrer dentro de um intervalo de tempo determinado. Este intervalo é determinado pelo requisito que relaciona o evento resposta a um ou mais eventos, que podem ser externos ou mesmo eventos intermediários, internos ao próprio sistema. Caso ocorra fora deste intervalo (com atraso, ou antes), o funcionamento da estratégia pode ser comprometido. A capacidade de o sistema elaborar o evento resposta no intervalo especificado vai depender de fatores como capacidade de processamento, granularidade do relógio disponível, interferências externas (interrupções), sincronização entre dispositivos no caso de sistemas distribuídos. Trabalhos como [CHO91] e [RAJ92] propõe a especificação de requisitos temporais baseados em uma lógica temporal, RTL, e um verificador de atendimento aos requisitos. A especificação do comportamento temporal desejado de um sistema, e validação do comportamento em tempo de execução comparando-o com os requisitos, torna estas propostas de monitoração mais completas e adequadas a validação de sistemas tempo-real.

A apresentação dos dados coletados e processados é um passo importante para a depuração e validação do sistema. Neste aspecto, [JOY87] inclui um console textual com a finalidade de interagir com a monitoração. Através do console é possível alterar a filtragem de eventos coletados, introduzir pontos de verificação, e visualizar o histórico de eventos. Além da apresentação em texto, um console gráfico é capaz de mostrar estado de processos através de uma representação que é atualizada a cada evento. Existe também

um console estatístico e de *deadlock*, e um verificador de protocolo que age a partir de um conjunto de requisitos de interação entre processos. Em [HAB90], é proposta uma análise de dados, através de regras constituídas por relações temporais entre eventos. Os resultados da monitoração são apresentados utilizando representações gráficas do histórico. É interessante observar que, por ser uma proposta de monitoração de tipo híbrido, os eventos a serem registrados dependem do programa do sistema; para tal fim existe um arquivo de definição, utilizado pelo subsistema de monitoração e alterado pelo usuário, que mapeia os eventos para posições de memória que acessam o dispositivo monitorador. Ainda neste aspecto, [TOK88] introduz em sua arquitetura de monitoração um visualizador composto por um conjunto de processos em uma máquina à parte do sistema alvo. Este visualizador apresenta o comportamento do sistema através de diagramas de Gantt, que permite acompanhar a execução dos processos e sua temporização, e estatísticas do sistema, como utilização de CPU e escalonabilidade de tarefas, e suporte a depuração (pontos de parada, etc.). Entretanto este trabalho não propõe analisador de requisitos.

A visualização de dados de natureza temporal é ainda abordada por [TiS:98, apresentando o sistema de avaliação temporal TimeWiz, um programa destinado a tratar dados referentes a sistemas tempo-real. A proposta inclui formulários para introdução de dados e modelagem do sistema. A partir do conjunto de dados, que representa o sistema, podem ser efetuadas análises do tipo de escalonabilidade, máximo atraso e número de trocas de contexto. Os dados e o resultados das análises podem então ser visualizados graficamente; juntamente com apresentações gráficas padrão, podem ser configurados diversos tipos clássicos de gráficos e diagramas (Gantt, barras, bolhas, linhas, etc.). A abordagem até certo ponto “em aberto” dada ao tópico de visualização torna a proposta muito completa, embora trabalhosa para utilização por ser necessário configurar a apresentação gráfica. Já [GIR98] parte de uma premissa diferente: ao invés de favorecer a visualização através da possibilidade de configuração de várias formas clássicas de apresentação, este artigo investiga novas formas de visualização de grandes quantidades de dados. A idéia central é agrupar graficamente padrões repetitivos de comportamento, e ressaltar os desvios do comportamento, através de elementos gráficos (linhas, formas,

cores). Também o abrangente trabalho de [TSA96] investiga a representação gráfica do comportamento de sistemas tempo-real e sua importância. Ao propor os formatos de representação, ressalta duas abordagens genéricas para as diversas apresentações. Os grafos de estado de processo são compostos de elementos (formas: elipse, retângulo) representando os diferentes processos, características visuais (cores, padrão de preenchimento) indicando o estado dos processos (ativo, em espera, bloqueado) e setas entre os elementos representado as interações entre os processos (comunicação, sinais). Já os grafos de interação de processo são compostos de diversas linhas de tempo paralelas, uma para cada processo investigado. Sobre estas indica-se os eventos dos processos (com marcas: círculos, cruz, barra) e o estado dos processos (através de cores). As interações entre processos são representadas por setas entre as linhas, com as extremidades localizadas no instante de tempo em que a interação ocorreu.

3.3 Perspectiva das Propostas de Monitoração

A maior parte das propostas de monitoração analisadas enfoca primariamente a coleta de dados ([JOY87], [MIL86], [TOK88], [LIU89], [DOD92], [BRA89], [MIN90]). Estes trabalhos apresentam as diferenças conceituais e de implementação que orientaram os respectivos projetos de subsistema de coleta de dados. No entanto, a escolha do tipo de coleta de dados (por código, por dispositivo, ou híbrido) depende fortemente do sistema alvo da monitoração e do acesso que se tem a seus componentes. É necessário que o código fonte, quer da aplicação quer do sistema operacional, esteja disponível para que possa ser instrumentado. Da mesma forma, para que possa ser implementada uma monitoração em nível de dispositivo, é necessário que um barramento de dados esteja disponível para ser acessado – e a monitoração será limitada aos dados presentes neste determinado barramento. Adicionalmente, cada sistema alvo terá um projeto de coleta de dados, código ou dispositivo, específico em função de suas características de construção. Esta realidade é verificada também no presente trabalho na medida em que a implementação do subsistema de coleta de dados é condicionada pelas alternativas tecnológicas disponíveis e viáveis existentes para o tipo de sistema alvo escolhido.

Abstraindo-se a especificidade do subsistema de coleta de dados, emergem das propostas de monitoração analisadas abordagens arquiteturais de variada abrangência. Dentre os trabalhos citados no parágrafo precedente, dois incluem pós-processamento da informação e apresentação textual e gráfica dos dados monitorados. Em [JOY87] (embora sua abordagem não seja tempo-real) há um analisador de protocolo para os eventos; e [TOK88] apresenta os resultados de processamento de estatísticas (ocupação de recursos, escalonabilidade de tarefas, etc.). Outros trabalhos já citados levam adiante esta abordagem, e conferem uma maior importância às possibilidades de validação do comportamento do sistema alvo. Por exemplo, [CHO91] (sistemas tempo-real concorrentes como alvo) e [RAJ92] (sistemas tempo-real distribuídos como alvo) propõe a validação de requisitos temporais em sua arquitetura de monitoração: à medida que os eventos são coletados, seus registros de tempo são comparados com restrições expressas em uma lógica temporal (RTL). A validação de requisitos temporais a partir do histórico de eventos abre possibilidades importantes para a monitoração em sistemas tempo-real, embora estas propostas não trabalhem a apresentação destes resultados. O artigo de [HAB90] também propõe possibilidade de validação de requisitos através de regras que podem ser definidas pelo usuário, além de uma visualização dos dados coletados como parte da função do monitor, e configuração dos eventos a serem coletados. Entretanto, tanto o tipo de regras quanto as modalidades de apresentação propostas poderiam ser ampliadas para tornarem-se mais expressivas.

Por fim, há interessantes contribuições para a visualização de dados de natureza temporal, apesar de não constituirem propriamente propostas de monitoração. [TIM98] destaca-se pelo ambiente de trabalho computacional e pela proposta aberta de visualização: uma variedade de diagramas e gráficos tradicionais é disponível e o usuário é responsável por configurar a apresentação dos dados da maneira que convier. Já [GIR98] apresenta novas representações de dados temporais, que podem ser importantes para a compreensão de aspectos do comportamento do sistema não acessíveis pelas técnicas corriqueiras de análise e apresentação.

4. Proposta para a ferramenta

4.1 Introdução

A implementação e validação de sistemas tempo-real necessita de suporte computacional adequado em nível tanto de hardware como de software, como metodologias de projeto, arquiteturas adequadas de equipamento e sistema operacional, e ferramentas de desenvolvimento e de depuração de aplicação. Controle e automação industriais através de redes de dispositivos com capacidade própria de processamento, como são os barramentos de campo, são exemplos de sistemas em que a existência de restrições temporais, associada com a natureza distribuída da aplicação, tornam este suporte realmente importante para que os requisitos de projeto sejam cumpridos pela implementação.

Este trabalho propõe uma ferramenta capaz de auxiliar na avaliação do comportamento temporal da comunicação em sistemas de automação industrial que utilizem o protocolo *Foundation Fieldbus*. Para tanto, procura estender as funcionalidades dos monitores de barramento comercialmente disponíveis, para tornar possível a validação de requisitos temporais especificados sobre eventos relevantes (as mensagens da comunicação), e efetuar apresentação gráfica dos resultados desta validação. Esta ferramenta, funcionando em conjunto com um sistema de captura e registro dos eventos do barramento, proporciona uma compreensão abrangente e ao mesmo tempo detalhada do comportamento temporal do *Fieldbus*, através da interpretação e análise dos dados e sua apresentação visualmente elaborada. É importante observar que o uso da ferramenta de validação desenvolvida não se restringe ao protocolo *Foundation Fieldbus*, podendo ser integrada a qualquer outro protocolo industrial.

Neste capítulo, a ferramenta proposta será descrita, juntamente com sua inserção e funcionamento no sistema alvo. Uma visão do conjunto da arquitetura de monitoração e das suas principais características será apresentada, seguida por uma descrição detalhada de suas respectivas partes funcionais.

4.2 Características da Arquitetura Proposta

Uma avaliação de características temporais de sistemas tempo-real passa pela realização de uma arquitetura capaz de monitorar, validar e apresentar séries de informações sobre o comportamento temporal do sistema investigado. Características desejáveis para uma tal arquitetura são, segundo [TSA96] :

- a arquitetura de monitoração deve interferir o mínimo possível com o desempenho normal do sistema alvo;
- a arquitetura deve permitir que o usuário defina o perfil da validação a ser realizada, em função de necessidades que possam mudar de uma aplicação para outra;
- a arquitetura deve dispor de uma referência temporal que seja global para o sistema alvo, para que os instantes de ocorrência de eventos gerados em diversos componentes do sistema possam ser comparados.

A proposta apresentada neste trabalho inclui as capacidades mencionadas: monitoração, validação e apresentação do comportamento temporal dos sistemas alvo de investigação. Também objetivou-se satisfazer as características mencionadas, consideradas importantes para arquiteturas de monitoração: mínima interferência; configurabilidade da avaliação pelo usuário; e referência de tempo global.

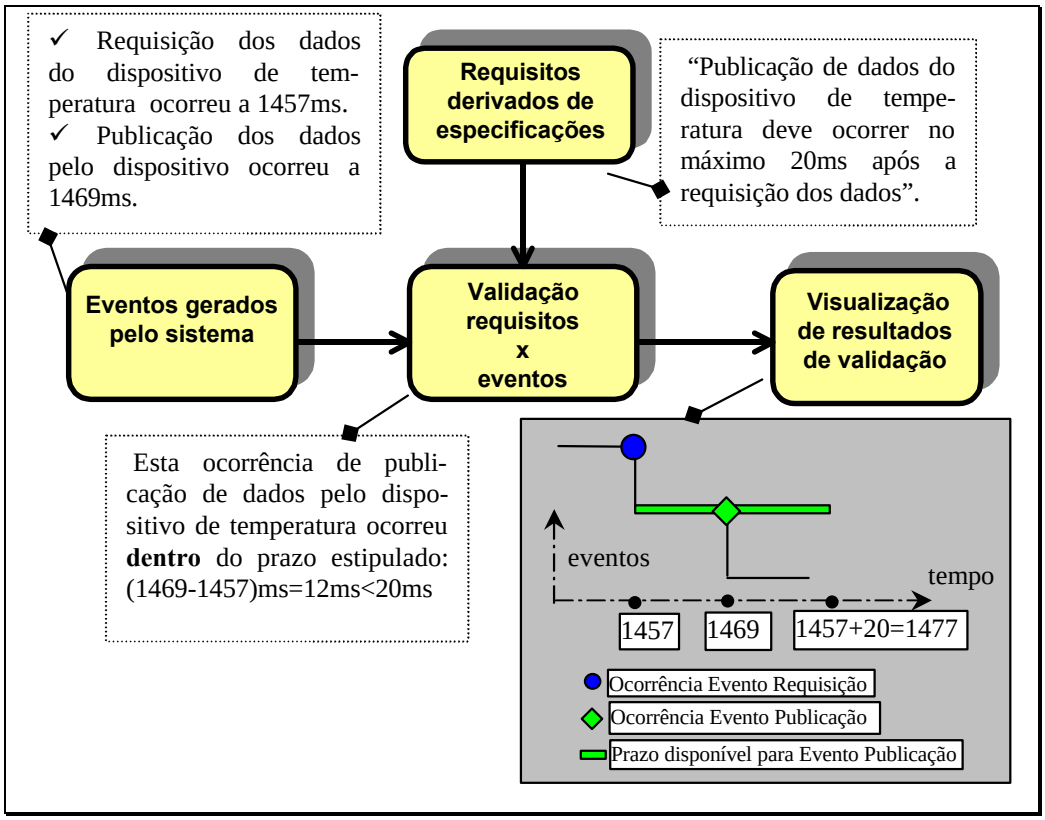

Figura 4.1 – Diagrama de Arquitetura

A proposta de ferramenta pode ser melhor compreendida através do diagrama de arquitetura e exemplo apresentados na Figura 4.1. É apresentada a validação de uma aplicação (estratégia de controle) apresentada no capítulo introdutório sobre *Foundation Fieldbus*, em que um dispositivo do tipo "transmissor de temperatura" deve publicar sua leitura da temperatura atual do processo, através de uma mensagem destinada ao dispositivo encarregado de calcular o valor de atuação no processo. A arquitetura proposta é constituída dos seguintes módulos:

- Especificação de requisitos
- Registro de eventos ocorridos no sistema alvo
- Validação de eventos em relação aos requisitos
- Visualização dos resultados

No contexto de validação, um conjunto de requisitos temporais deve ser gerado, relacionando eventos do sistema entre si e com intervalos de tempo. Os requisitos são elaborados a partir de necessidades da validação. Podem ser especificações oriundas de projeto de aplicação, como o exemplo da figura, em que o período da publicação de um

dado é ajustado em função da estratégia de controle escolhida. No exemplo apresentado, o requisito afirma que é necessário que, após requisitados, os dados referentes à temperatura atual do processo devem ser publicados em um prazo máximo de 20ms. Estes requisitos são especificados em uma notação própria para descrever restrições temporais, a qual será apresentada em detalhe posteriormente.

Os eventos do sistema, que são as mensagens trocadas pelos dispositivos e que trafegam no barramento, devem ser registrados juntamente com a informação de seu instante de ocorrência. Cada vez que uma mensagem relevante no sistema ocorre, como por exemplo a mensagem de requisição de dados no exemplo, é feito um registro da ocorrência e do instante em que ocorreu, valendo o mesmo para a mensagem de publicação de dados. O registro de eventos, neste caso, informa que os instantes de ocorrência verificados foram respectivamente 1457ms para a requisição e 1469ms para a publicação.

A validação compara o instante de ocorrência dos diversos eventos, informando se os requisitos relacionados foram atendidos ou não, e como foi atendido o requisito (por exemplo, a folga para o prazo de execução, ou a variação na periodicidade observada). Na figura observamos a validação do requisito relacionando a requisição e a publicação de dados do dispositivo de temperatura. O requisito exige, conforme mencionado, que a publicação ocorra no máximo 20ms após a requisição. A validação calcula o atraso da publicação (=1457ms) em relação a requisição (=1469ms), que é igual a 12ms (=1469-1457), e verifica que este é menor que o máximo estipulado (20ms), ou seja, o requisito foi atendido. Os resultados obtidos com a validação são passados para uma apresentação, de característica gráfica, de maneira a que possam compreendidos com mais clareza, aumentando o conhecimento sobre o sistema e mesmo possibilitando adaptações da implementação do mesmo. Esta arquitetura é baseada em arquiteturas clássicas de monitoração com validação de comportamento, tais como [CHO91], com o acréscimo de uma apresentação dos dados de validação.

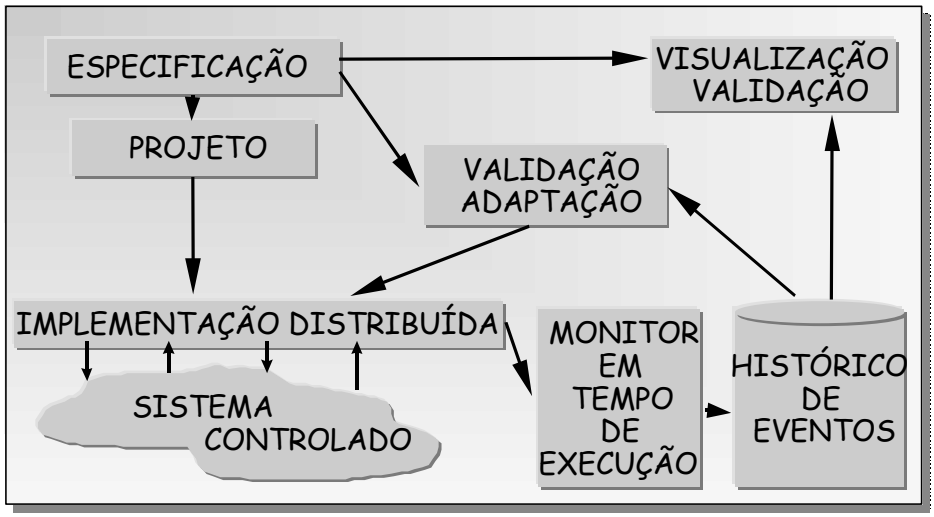

Figura 4.2 – Ciclo de desenvolvimento de projeto

O papel da arquitetura descrita, dentro do ciclo de desenvolvimento de projeto, pode ser compreendido a partir do diagrama da Figura 4.2. Em um ciclo tradicional de desenvolvimento de projeto, o ponto de partida é um conjunto de especificações, que descreve o problema a ser resolvido e as características da solução (controle de um determinado sistema) que deve ser encontrada. Em função das especificações, um projeto de solução é gerado com a missão de atender às especificações iniciais. A fase seguinte é a implementação: corresponde construtivamente ao projeto, e obedece às funcionalidades descritas na especificação, ou seja, agir sobre o sistema a ser controlado de forma a obter determinado comportamento. No caso de os elementos de controle e processamento serem diversos, e estarem espacialmente dispersos, a implementação é dita distribuída, como é o caso da tecnologia (*Foundation Fieldbus*) abordada por este trabalho.

Estes passos de projeto tradicional são estendidos pela presente proposta. Seguindo o diagrama da Figura 4.2, um monitor que armazena informações sobre eventos do sistema em um histórico (correspondendo ao bloco *Eventos* na Figura 4.1), e as especificações que deram origem ao projeto (ver bloco *Requisitos* na Figura 4.1), são base para uma validação e visualização (respectivamente blocos *Validação* e *Apresentação*, ainda na Figura 4.1) do comportamento temporal do sistema. Uma extensão deste tipo ao ciclo tradicional de desenvolvimento de projeto torna possível uma compreensão maior do comportamento temporal da implementação, abrindo caminho para uma alteração e adaptação da implementação para torná-la mais adequada às especificações iniciais.

Serão descritos nas próximas seções o registro de eventos, a especificação de requisitos, a validação de requisitos e eventos, e a visualização dos dados.

4.3 Registro de Eventos

O objetivo desta pesquisa é validar a ocorrência de eventos da comunicação *fieldbus* em relação a requisitos temporais. **Eventos** serão informalmente referidos como os acontecimentos relevantes do sistema; cada vez que se verificar efetivamente um acontecimento, haverá uma **ocorrência** do evento (ver Figura 4.3a e Figura 4.3b). Em um sistema distribuído e tempo-real, como por exemplo os sistemas *fieldbus* abordados neste trabalho, a comunicação, e seu comportamento temporal, têm importância central para desempenho conjunto de suas respectivas funções pelos dispositivos individuais. A comunicação neste contexto tem como seus eventos as diferentes mensagens utilizadas pelos dispositivos para trocar informações; as ocorrências serão a efetiva publicação destas mensagens no barramento.

A validação proposta constitui-se na investigação de intervalos de tempo, descritos por requisitos, entre determinadas ocorrências de certos eventos. Por este motivo, exige que estas ocorrências que serão validadas sejam registradas juntamente com o instante de tempo em que ocorreram. Também é necessário que os registros das ocorrências permaneçam armazenados até o momento de sua validação, e enquanto forem necessários para a validação de algum requisito. Por exemplo, um requisito pode estar definido como: “o intervalo entre a penúltima ocorrência da mensagem <<sincronização de dispositivos>> e a última ocorrência desta mensagem não deve ultrapassar 1500ms”. No caso deste requisito, deve-se armazenar o registro das duas últimas ocorrências da mensagem, e não somente da última. Para este requisito, cada vez que ocorrer novamente este evento, o registro mais antigo (ou seja, o antepenúltimo) pode ser descartado, pois não será mais utilizado na validação.

Os registros de ocorrências devem conter informações que serão utilizadas na validação e também outras informações que possam ser úteis no momento da apresentação e visualização dos resultados. Cada ocorrência deve identificar o evento a que corresponde, o instante de tempo, e ainda sua ordenação absoluta (estabelecendo o número de ocorrências do evento desde o início da atividade de registro). Outros dados, tais como duração da transmissão da mensagem, seu endereços fonte e destinatário, parâmetros e campo de dados podem ser utilizados no momento da apresentação dos

resultados da validação, para auxiliar na identificação e compreensão de comportamentos do sistema.

As ocorrências dos eventos são organizadas em estruturas de dados denominadas de histórico de eventos (Figura 4.3c), que são consultadas para efetuar a validação do comportamento temporal do sistema (ver [MCD89]). Um histórico contém um conjunto de registros de ocorrências de um determinado evento, e o conjunto de históricos contém o registro de todos os acontecimentos relevantes do sistema (Figura 4.3d). As ocorrências podem ser acessadas de duas formas: a primeira é a partir de sua ordenação absoluta, ou seja, a ordem de registros iniciando na primeira ocorrência percebida pelo sistema. Esta forma é indicada por Evento@x em que x é o ordinal da ocorrência referida, como por exemplo a primeira ocorrência (EventoA@1), ou a décima-quarta (EventoA@14) do evento EventoA. Já a segunda forma de acesso é a ordenação relativa à última ocorrência; é indicada por @-x, em que x é ordinal que, iniciando na última ocorrência, conta retroativamente as ocorrências mais recentes. Ou seja, Alarme@-1 indica a última ocorrência, e Alarme@-2 a penúltima ocorrência de um evento do tipo Alarme.

Deve ser considerado o número de ocorrências a ser armazenado em cada histórico de eventos. Armazenar todas as ocorrências de todos os eventos permite validar requisitos que envolvam, à mesma vez, acontecimentos em qualquer instante do ciclo de monitoração do sistema. Entretanto, um histórico tão abrangente pode não ser necessário, além de esbarrar em limitações de memória de equipamento. Requisitos normalmente referem-se a um determinado número de ocorrências indicadas pela ordenação absoluta, e a um determinado número indicadas pela ordenação relativa. Por exemplo, um requisito pode expressar que “ a ocorrência @-2 da mensagem <<reconhecimento de alarme>> deve ocorrer no máximo 4s após a ocorrência @3 da mensagem <<estado de alarme>>”. No caso deste requisito, o histórico precisa que sejam armazenadas, para que a validação possa ser efetuada, apenas a terceira ocorrência do evento <<estado de alarme>>, entre todas que sejam registradas; e apenas as duas últimas ocorrências da mensagem <<reconhecimento de alarme>>. Uma solução seria procurar identificar, no conjunto de requisitos, as referências a eventos e criar os históricos de forma a armazenar todas as ocorrências necessárias, e apenas estas. No presente trabalho optou-se por simplificar esta restrição, e armazenar um número fixo de ocorrências, como na Figura 4.3c.

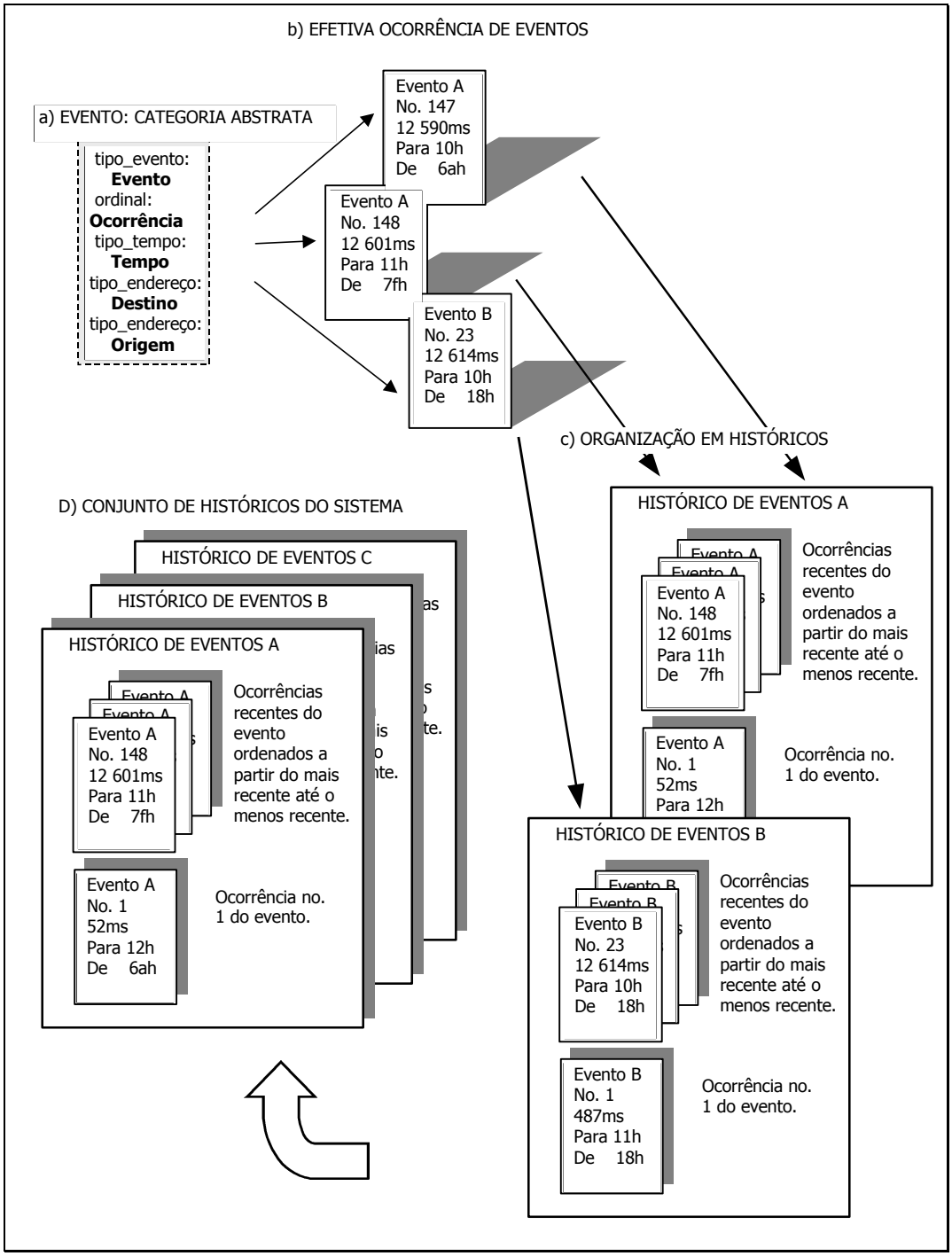

Figura 4.3 – Estruturas de dados

Como já mencionado, os eventos relevantes na abordagem deste trabalho são as mensagens que os dispositivos trocam entre si e que são publicadas no barramento comum de comunicação. Estas mensagens possuem várias funções, e sua classificação em eventos será baseada nestas diversas funções. Existem por exemplo mensagens com a função de passagem de permissão do tipo periódico para publicação de dados; permissão do tipo aperiódico; devolução de permissão; mensagens resposta à passagem de permissão que contém os dados a serem publicados; teste de dispositivo presente na rede; a resposta de dispositivo a este teste; publicação de tempo global com objetivo de sincronizar os dispositivos; e outros tipos ainda. Para aumentar a especificidade

da validação, se necessário, um evento pode ser definido, além de sua função, através dos endereços destino ou origem, ou ambos, da mensagem.

Na estrutura das mensagens, as características que são utilizadas para identificar os eventos são, portanto:

- função da mensagem: está contida em uma parte da mensagem conhecida por **campo de controle**. Para uma revisão das principais funções de mensagens *fieldbus,* como por exemplo, PT -- "*passs token*" — ou CD — "*compel data*", ver tabela no capítulo que contém revisão sobre *Foundation Fieldbus*;
- destino: informa dispositivo e entidade de enlace de dados que é destinatário da mensagem. Auxilia na identificação das mensagens pertencentes a uma conexão de dados, e está contido no **campo de destino**;
- origem: algumas mensagens possuem um **campo de origem** (dependendo do tipo definido pela função da mensagem). Assim como o campo de destinatário, auxilia na identificação das mensagens pertencentes a uma conexão de dados;

O subsistema de coleta de dados consiste de um monitor dedicado de barramento, que registra as mensagens circuladas. Isto significa um dispositivo especial, que é capaz de receber todas as mensagens presentes no barramento, diferentemente dos dispositivos normais, que recebem apenas aquelas destinadas a seu endereço. Também diferente é o comportamento deste dispositivo especial dentro do barramento: o monitor não responde a requisições de presença de dispositivo, nem a qualquer outra, nem tampouco publica qualquer tipo de mensagem, tornando-se invisível e não interagindo com os outros dispositivos presentes. Desta forma, sua presença não altera o comportamento dos dispositivos em rede. Isto torna possível uma validação temporal do sistema sem que sua significação seja alterada pela interferência do monitor. Adicionalmente, este dispositivo tem a capacidade de registrar as mensagens de forma a disponibilizá-las para uso posterior, acrescentando ainda ao registro o instante de tempo da ocorrência da mensagem. A referência temporal é única, e está ligada ao relógio do dispositivo. Todas as mensagens no barramento recebem o registro de tempo relativo a esta mesma referência.

Para a finalidade de tal monitor, é utilizado um microcomputador PC compatível com uma placa de interface *Foundation Fieldbus* e um programa de captura de mensagens, que as grava em um arquivo de formato binário, juntamente com seu

registro de tempo. O presente trabalho utilizou componentes comercialmente disponíveis para realizar este subsistema: a placa de interface padrão ISA modelo PCI, e o programa de captura de mensagens FBView (ver referência), ambos do fabricante Smar Equipamentos Industriais (Figura 4.4).

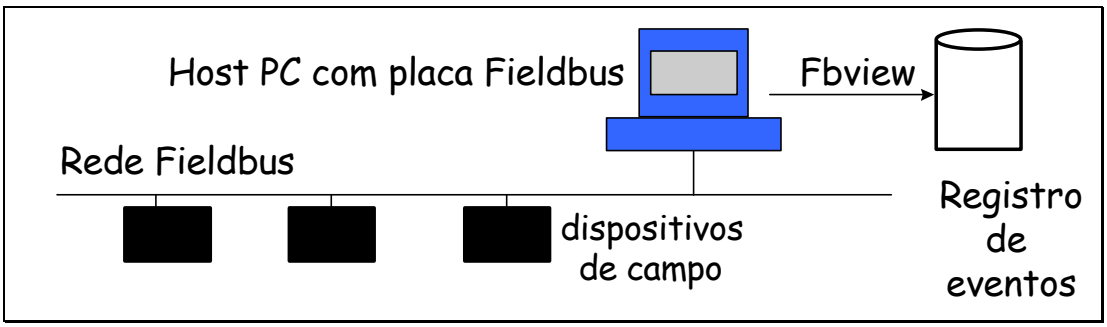

Figura 4.4 – Subsistema de coleta de dados

O registro de eventos gerado por esta arquitetura é um arquivo binário, contendo as mensagens, com seu registro de tempo e uma numeração de ordem associada. Este arquivo será utilizado como entrada pela ferramenta de validação, juntamente com os requisitos temporais especificados.

4.4 Especificação de requisitos

Sistemas tempo-real são caracterizados pela existência de requisitos temporais, relacionando instantes de ocorrência de eventos do sistema entre si. Isto ocorre por que, ao ter como função primordial o controle de processos e operações no mundo real, a reação do sistema controlador deve ser correta, não somente do ponto de vista lógico, mas também do ponto de vista de tempo. Por exemplo, ao ser controlado um sistema em movimento, como um braço robótico, existem as atividades de aquisição dos dados de posição atual, de cálculo da atuação, e de atuação sobre os motores. Neste caso, a atuação sobre os motores deve ser feita dentro de um prazo limite, antes que os dados de aquisição que basearam o cálculo deixem de ser atuais. Mesmo que o algoritmo utilizado fosse extremamente refinado, uma eventual demora no processo de cálculo tornaria inválidos seus resultados.

No contexto deste trabalho, os requisitos irão expressar relações de tempo entre a ocorrência de mensagens no barramento. Validar o comportamento temporal da comunicação é importante porque, em sistemas distribuídos, depende-se, entre outros fatores, da correção temporal na transmissão e recepção de mensagens para garantir a correção temporal do comportamento do sistema como um todo.

Os requisitos que este trabalho aborda podem provir de aspectos funcionais (ligados à aplicação) ou estruturais (do próprio sistema *fieldbus*). Requisitos derivados de aspectos funcionais são resultado de exigências do processo sob controle, mapeadas para o subsistema de comunicação. Por exemplo, se uma aplicação de controle exigir que os dados de uma variável de entrada sejam atualizados no mínimo a cada 10ms, e estes dados precisarem ser comunicados de um dispositivo para outro, é de se esperar que a comunicação ocorra, pelo menos, a cada 10ms. Já requisitos estruturais decorrem de necessidades do próprio sistema controlador que devem ser satisfeitas para que funcione corretamente. Um exemplo no *fieldbus* é a mensagem de distribuição de tempo (TD, "*time distribution*"): para garantir a sincronização de todos os dispositivos dentro de um limite de erro especificado, esta mensagem deve ocorrer a intervalos mínimos determinados (que dependem da taxa de deriva do relógio dos dispositivos e do limite de erro especificado). Em última análise, requisitos estruturais decorrem de aspectos funcionais generalizados, ou seja, para que a aplicação funcione corretamente, o sistema controlador deve também funcionar corretamente. Na verdade, do ponto de vista da validação, exige-se que ambos os tipos de requisitos sejam cumpridos pela comunicação; desta forma, a ferramenta desenvolvida não faz distinção, na especificação nem em outra etapa, entre requisitos funcionais e estruturais.

Uma questão importante relativa a requisitos é sua representação adequada. O esquema de representação deve ser flexível para permitir a expressão de todos os requisitos que forem necessários validar. Deve ser também adequado para que não ocorram ambigüidades, ou seja, uma expressão não deve permitir dúvidas entre mais de um possível significado.

No presente contexto de validação de comportamento, emergem outras características importantes para o esquema a ser utilizado. Uma característica é a possibilidade de uma representação quantitativa (com a especificação de intervalos em unidades de tempo) e não apenas qualitativa (em que apenas a ordenação relativa dos eventos pode ser especificada). Também importante é a possibilidade de alguma tolerância na representação do requisito. Eventos no mundo real, com suas incertezas na medição de um instante exato, e a existência na prática de um intervalo de valores válidos na determinação da solução de um requisito, tornam interessante que o esquema de representação suporte esta característica. Adicionalmente, é desejável a representação em alto nível, ou seja, de uma forma próxima à linguagem natural.

Representações muito distantes da linguagem natural, embora possam ser corretas e expressar uma extensa gama de requisitos, dificultam a compreensão e a construção de expressões na prática de uso, e podem até mesmo limitar a sua utilização por não especialistas.

O sistema de notação utilizado é apresentado em [PER95] . Trata-se de um sistema baseado na lógica temporal RTL [JAH86]. Esta notação mostra-se adequada à expressão dos requisitos abordados por este trabalho (ver também o levantamento feito por [DAS85] sobre capacidades necessárias para sistemas de notação). Expressa de forma quantitativa uma série de relações temporais entre eventos puntuais, isto é, que ocorrem em um instante de tempo, através de predicados temporais. As relações expressas são em alto nível, isto é, seu significado é similar ao de construções em linguagem natural (utilizando, por exemplo, predicados como “antes” ou “ciclicamente”). As relações podem ser expressas de forma a incluir uma tolerância para a solução, ou seja, a relação será satisfeita por um evento que ocorra dentro de um intervalo de tempo, e não somente em um instante preciso de tempo. As relações podem ser expressas de maneira absoluta (contagem de eventos e tempo a partir de um instante inicial) ou de maneira relativa (contagem a partir do tempo ou evento atual). Esta notação permite a expressão de requisitos de forma simples com a finalidade de validação pela ferramenta, sem que complexidades desnecessárias (para esta determinada finalidade) tornem-se um empecilho à sua utilização. O sistema é utilizado também em outros trabalhos no contexto do Laboratório de Automação do Departamento de Engenharia Elétrica, contando portanto com a vantagem adicional de permitir uma integração dentro de um conjunto de trabalhos relacionados a sistemas tempo-real distribuídos.

Relação Temporal	Expressão da Relação	Exemplo
Seqüência temporal	(<ev1>AFTER<ev2>) <intervalo>	(Peça_nova AFTER Peça_anterior) <1s, 2s>
	(<ev1>BEFORE<ev2>) <intervalo>	(Peça_anterior BEFORE Peça_nova) <1s, 2s>
Sincronização	(SYNCHRON<lista_ev>) <duração>	(SYNCHRON sinal1, sinal2, sinal3) <500ms>
Periodicidade	(CYCLIC<ev1>, <ciclo>)	(CYCLIC AtualizaAtuador, 75ms)

Tabela 4.1– Notação temporal utilizada

Nesta notação, as relações básicas são as de seqüência. O predicado <ev1>AFTER<ev2> significa que o evento de sistema, identificado pela denominação <ev1>, deve ocorrer após o evento <ev2>. O intervalo especifica quantitativamente a solução para este requisito: o evento <ev1> deve ocorrer pelo menos tanto tempo como o extremo inferior do intervalo, e no máximo tanto tempo como o extremo superior do intervalo, após <ev2>. Por exemplo (Figura 4.5), o requisito "(Peça_nova AFTER Peça_anterior) <1s, 2s>" significa que o evento de sistema denominado "Peça_nova" deve ocorrer após o evento "Peça_anterior"; é exigido que o evento "Peça_nova" ocorra pelo menos 1s após, e no máximo 2s após, "Peça_anterior". A relação de seqüência BEFORE é construída similarmente, significando que o primeiro evento deve ocorrer antes do segundo, dentro do intervalo especificado.

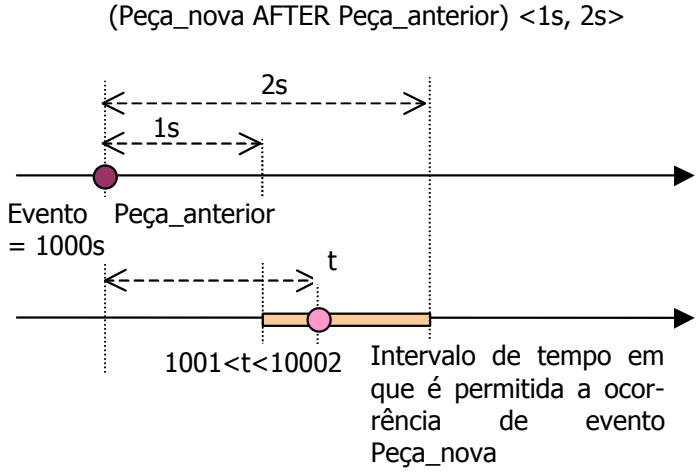

Figura 4.5 - Exemplo de Predicado AFTER

As outras relações apresentadas são derivadas das relações básicas, e foram criadas para facilitar a expressão de requisitos comumente utilizados em sistemas tempo-real. A expressão “(SYNCHRON<lista_ev>)<duração>” exige que todos os eventos da lista ocorram sincronizadamente, com no máximo a duração expressa em <duração> decorrendo entre o primeiro evento a ocorrer e o último. Ver o exemplo na Figura 4.6a). Já o predicado “(CYCLIC<ev1>, <ciclo>)” expressa eventos periódicos, com período <ciclo>, conforme exemplo na Figura 4.6b).

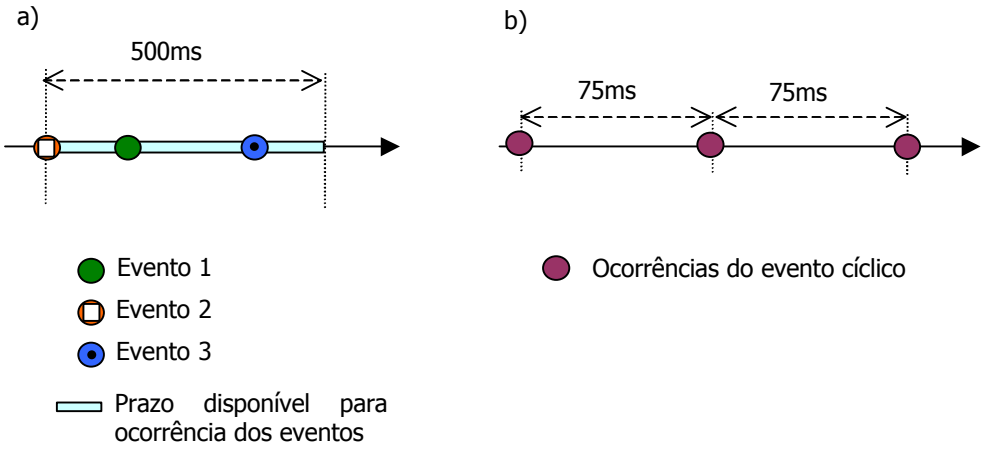

Figura 4.6– Exemplo de predicados SYNCHRON e CYCLIC

Nesta notação, as diversas ocorrências do mesmo evento são distinguidas por um índice, que no presente trabalho será aplicado ao nome do evento através do operador @. O significado do índice é semelhante ao da notação em RTL: evento@*i* significa a *i*-ésima ocorrência do evento se *i* for maior que zero, e a *i*-ésima mais recente ocorrência do evento se *i* for negativo. Este formato de referência a ocorrências já foi discutido acima, na seção **Registro de Eventos**.

A notação inclui a possibilidade de conectar dois predicados temporais através de operadores lógicos. Os operadores utilizados são: “ou” lógico (OR) e “e” lógico (AND).

A ferramenta desenvolvida identifica os diversos eventos com os nomes E*n*, em que *n* varia de 0 a um número máximo. A correspondência entre o evento e a mensagem é dada em uma tabela, que contém entradas como: classe=2, controle=0x33, destino=0x10. A “classe” identifica o evento; neste caso, portanto, o evento em questão é denominado E2. O identificador “controle” refere-se ao campo de controle dentro da mensagem, que contém um código indicando seu tipo. Consultando-se a tabela de

mensagens *Foundation Fieldbus*, no capítulo Estado da Arte, verifica-se que o código de controle 0x33 corresponde a mensagens do tipo PT (*pass token*), isto é, passagem de permissão aperiódica. Por fim, “destino” identifica o dispositivo ao qual a mensagem é dirigida, neste caso, o dispositivo com endereço 0x10. Ou seja: atribui-se ao nome de evento E2, as mensagens de tipo PT direcionadas ao dispositivo de endereço 0x10. Existe ainda a possibilidade de especificação de uma “máscara” para que dois valores de controle diferenciados por determinados bits sejam considerados o mesmo evento. Por exemplo, PT com prioridades diferentes no barramento, 0x33 (prioridade normal) e 0x32 (prioridade alta) são unificados utilizando-se um valor de “máscara” igual a 0x1.

Com esta notação, pode-se expressar requisitos temporais necessários à validação do comportamento da comunicação em *fieldbus*. Alguns de tais requisitos, e seu significado, serão apresentados, relacionados ao exemplo transmissor de temperatura e controle de válvula já apresentado. Serão assumidas algumas correspondências para tornar mais claros os requisitos. Para tanto, o evento E0 será considerado a requisição de publicação de dados periódicos para o dispositivo de temperatura; o evento E1, publicação de dados periódicos pelo dispositivo de temperatura; evento E2, requisição de dados periódicos para um outro dispositivo no barramento, por exemplo transmissor de pressão; E3, requisição de dados periódicos para um outro dispositivo no barramento, por exemplo transmissor de fluxo. Portanto:

- um requisito “(E1@-1 AFTER E0@-1) [5, 15] OR (E1@-1 BEFORE E0@-1) [0]” significa que a publicação dos dados periódicos pelo dispositivo de temperatura é esperada entre 5 e 15 unidades de tempo após a respectiva requisição. O segundo predicado torna válido o caso em que os dados foram publicados, e não ocorreu ainda uma nova requisição.
- um requisito “(SYNCHRON E1@-1, E2@-1, E3@-1) [50]” pode significar que é importante para a estratégia de controle implementada que as variáveis do processo sejam adquiridas e comunicadas com um intervalo mínimo entre uma e outra, de maneira que o conjunto de dados possa ser considerado uma descrição precisa do sistema naquele instante. No presente exemplo, a separação máxima entre os instantes de comunicação dos dados é admitida como 50 unidades de tempo.

4.5 Validação de Eventos e Requisitos

A validação proposta, no presente contexto, significa avaliar o instante de ocorrência dos eventos do sistema, a cada vez que ocorre um evento, e compará-los com intervalos de tempo determinados por requisitos temporais especificados. Se o evento ocorreu dentro do intervalo de tempo determinado pelos requisitos, a restrição imposta foi cumprida; caso contrário, ocorreu uma violação de algum requisito.

A validação, no presente contexto, propõe ainda a informação dos intervalos de tempo determinado pelos predicados dos requisitos, dentro dos quais devem ocorrer o eventos. Com esta informação, é possível também avaliar qualitativamente a ocorrência do evento em relação ao requisito. Por exemplo, pode-se observar se o evento ocorre freqüentemente próximo ao último instante possível para que o requisito seja atendido. Este caso indica provavelmente que o sistema está subdimensionado, trabalhando próximo de seus limites, e que alguma situação inesperada pode fazer com que o requisito deixe de ser atendido. Se, ao contrário, o evento ocorrer sempre com muita folga para atendimento ao requisito, o intervalo destinado ao evento possivelmente está superdimensionado, e o requisito pode ser otimizado para melhor aproveitamento do tempo disponível. Quando o requisito não é atendido, é possível avaliar a magnitude do atraso (ou adiantamento) do evento em relação ao intervalo permitido, e também a freqüência com que esta situação ocorre, orientando com estas informações a tarefa de depuração do sistema. A informação de intervalo não é rigorosa no sentido de que sua validade pode ser alterada pela conexão de predicados por proposições lógicas; entretanto, é uma facilidade que não é abordada por nenhuma das propostas pesquisadas neste trabalho, e se usada de forma inteligente, concomitantemente com a escolha apropriada da expressão dos requisitos, pode fornecer dados interessantes e úteis, não disponíveis de outra forma.

A ferramenta apresentada neste trabalho efetua a validação com um arquivo de mensagens coletadas. Isto significa que a validação é obtida após o tempo de execução do sistema alvo da monitoração. Neste sentido, a informação obtida pode ser utilizada para uma análise ou adaptação *a posteriori* do sistema.

A arquitetura proposta para a ferramenta (ver Figura 4.7) integra o registro de eventos, a consulta a requisitos temporais, e a validação e visualização dos resultados. Um módulo de entrada de eventos varre o arquivo de mensagens e gera eventos para a

ferramenta, um por vez. Estes eventos são registrados em históricos de eventos, de acordo com seu tipo. A cada evento ocorrido, o validador de requisitos é invocado para verificar a situação atual. Os resultados da validação são, por fim, enviados a um módulo de visualização de resultados.

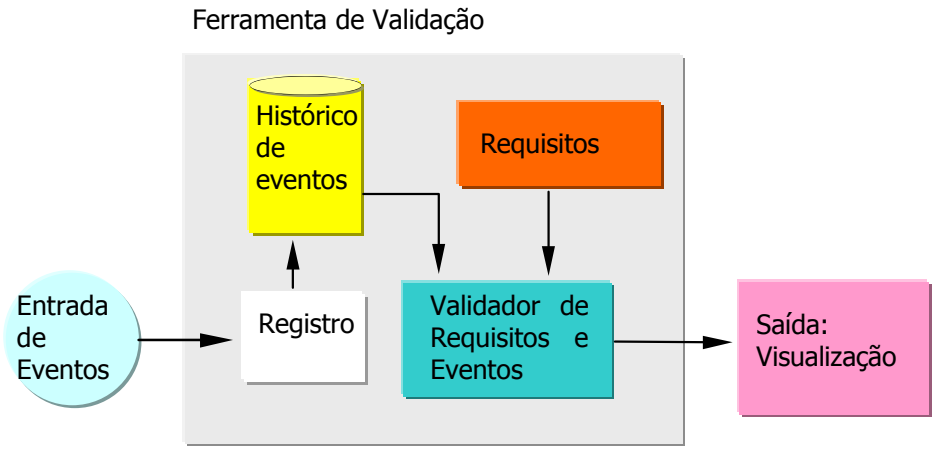

Figura 4.7– Diagrama de blocos da ferramenta

4.6 Mecanismo da Validação

A validação proposta através da ferramenta desenvolvida baseia-se na criação de uma lista de estruturas de dados, chamadas de *tuplas evento-requisito*, para cada evento e requisito que for validado. Estas tuplas constituem-se no resultado da validação, e são as estruturas cujo conjunto é apresentado pelo módulo de visualização.

Quando um evento é gerado a partir do arquivo de mensagens capturadas, sua ocorrência é registrada, e em seguida é passado a um validador, que é o núcleo da ferramenta desenvolvida. O validador então busca os requisitos que são afetados diretamente pelo evento ocorrido(ou seja, aqueles requisitos que referenciam o evento em sua expressão). Para cada requisito que mencione este evento, será gerada uma estrutura de dados *tupla evento-requisito*. Por exemplo, se um evento ocorrido é mencionado em 2 requisitos, serão geradas duas tuplas para esta ocorrência, cada uma como resultado da validação de um dos requisitos. Estas tuplas então são sucessivamente passadas ao módulo de visualização, que é responsável por apresentar graficamente o conjunto de tuplas como resultado da validação.

Para a avaliação de cada um dos requisitos individualmente, em primeiro lugar avaliam-se cada um dos predicados temporais especificados, contidos no requisito, como “verdadeiro” ou “falso”. A segunda etapa, se houver mais de um predicado, é a

conexão lógica dos resultados dos predicados, de acordo com os operadores lógicos especificados: “ou” ou “e”.

Os predicados temporais são semanticamente equivalentes a desigualdades relacionadas a instantes de tempo. São avaliados através da substituição das ocorrências especificadas no requisito, pelos respectivos registros de tempo das ocorrências, e verificando-se se a inequação resultante é válida. Para o caso do predicado BEFORE:

$(E_a \text{ BEFORE } E_b)\ [I_1, I_2]$

onde E_a e E_b são os eventos envolvidos, e I_1 e I_2 são os extremos do intervalo relativo especificado. As referências E_a e E_b são substituídas pelo registro de tempo das respectivas ocorrências; e então avaliam-se as desigualdades:

$Eb - I_2 \leq Ea$ e $Eb - I1 \geq Ea$ (ver Figura 4.8**Erro! A origem da referência não foi encontrada.**)

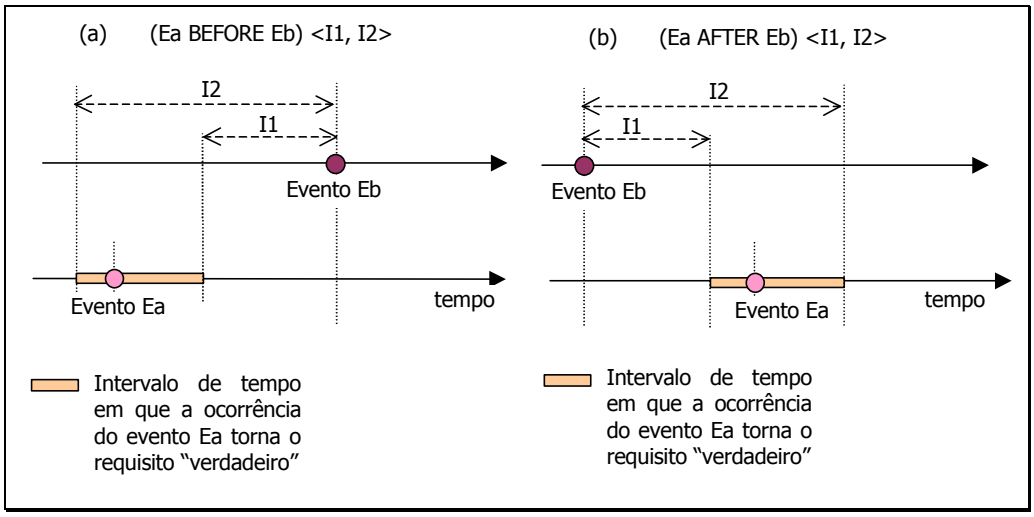

Figura 4.8 – Validação dos predicados AFTER e BEFORE

Para o caso do predicado AFTER:

$(E_a \text{ AFTER } E_b)\ [I_1, I_2]$

avalia-se as desigualdades:

$E_a \leq E_b + I_2$ e $E_a \geq E_b + I_1$ (ver Figura8b)

Para os dois casos, ambas as desigualdades apresentadas devem avaliar “verdadeiras” para que o predicado avalie “verdadeiro”. A Figura 4.5 apresenta um exemplo do predicado e sua avaliação. Havendo dois predicados conectados por um operador lógico, o requisito é avaliado pelo resultado da operação indicada entre as duas

avaliações de predicado. Isto é, se houver um requisito como “PREDICADO1 OU PREDICADO2”, e PREDICADO1 avaliar “verdadeiro” e PREDICADO2 avaliar “falso”, então o requisito avalia “verdadeiro”.

Conforme mencionado, o validador retorna, além da avaliação booleana (“verdadeiro”/”falso”), um intervalo descrevendo um “intervalo de validade” permitido pelo predicado para o evento especificado. Esta informação tem a função de aumentar o conhecimento sobre o comportamento temporal dos eventos envolvidos em relação aos requisitos sendo avaliados. É uma informação correta, porém seu significado não rigorosamente determinado, uma vez que os operadores que conectam predicados podem alterar seu significado e relacionamento. Entretanto, se usada de maneira conveniente, esta informação pode revelar aspectos importantes do comportamento temporal do sistema investigado.

Voltando ao caso do predicado BEFORE,

$(E_a \text{ BEFORE } E_b)\ [I_1, I_2]$

o intervalo I que é retornado para este predicado é, portanto, aquele dentro do qual a ocorrência do evento Ea torna sua avaliação “verdadeira”, ou seja,

$I = [Eb - I2, Eb - I1]$

E, para o predicado AFTER,

$(E_a \text{ AFTER } E_b)\ [I_1, I_2]$

o intervalo retornado é:

$I = [Eb + I1, Eb + I2]$

Tomando novamente como exemplo a situação da Figura 4.5, temos que, dado o requisito “(Peça_nova AFTER Peça_anterior) <1s, 2s>”, e dada a ocorrência do evento “Peça_anterior” no instante t_0 = 1000s, o intervalo de validade retornado pelo requisito é I = [1001, 1002]. Se o evento “Peça_nova” ocorrer, por exemplo, a 1001,3s, este predicado retorna “verdadeiro”. Já se o evento “Peça_nova” ocorrer no instante 1002,5s este predicado será avaliado como “falso”.

O predicado SYNCHRON,

$(\text{SYNCHRON } E_0, ..., E_n)\ [\ D]$

avalia se os eventos da lista ocorreram próximos no tempo, ou seja, se a diferença entre o instante de ocorrência do evento mais antigo e do mais recente é menor do que a duração expressa D. A desigualdade

$$\max(E_0, ..., E_n) - \min(E_0, ..., E_n) \leq D$$

deve ser avaliada verdadeira para que o requisito seja avaliado verdadeiro (ver Figura 4.9).

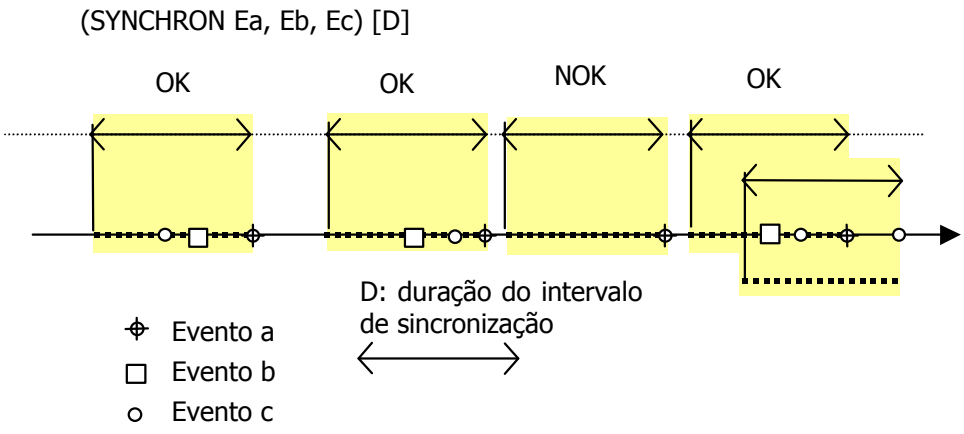

Figura 4.9 – Validação do predicado SYNCHRON

Por fim, o predicado CYCLIC

(CYCLIC E) [C, j]

que descreve um evento E de tipo periódico, com período C, e máxima variação do período j, é avaliado através da desigualdade

$$|E@-1 - E@-2 - C| \leq j$$

ou seja, a diferença entre o intervalo entre as duas últimas ocorrências do evento e o ciclo especificado é a variação de periodicidade calculada, que deve ser menor do que a variação especificada (ver Figura 4.10). É importante observar que, da forma como é proposto o seu cálculo, a variação pode significar tanto diferença eventual entre o ciclo especificado e o intervalo entre duas ocorrências do evento, como também uma diferença entre o ciclo especificado e o período efetivo das ocorrências do evento.

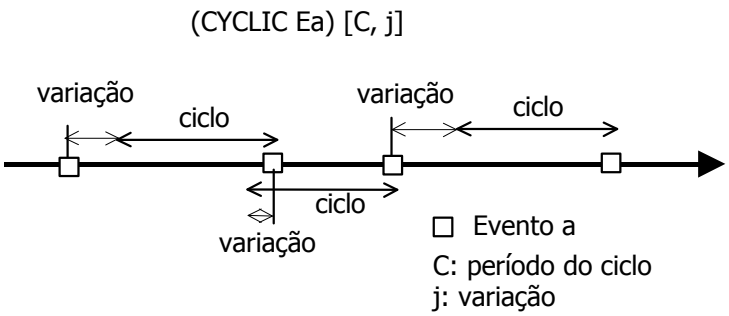

Figura 4.10 – Validação do predicado CYCLIC

O intervalo que é retornado para este requisito é, novamente, aquele dentro do qual a ocorrência do evento é avaliada verdadeira, ou seja,

$$I = [\ E@-2 - C - j,\ E@-2 - C + j\ ]$$

Em resumo, requisitos são avaliados para ocorrências de eventos do sistema, retornando um valor booleano que indica se foram atendidos ou não, e também um intervalo temporal de validade, relacionando a ocorrência ao predicado especificado no requisito.

4.7 Visualização dos Dados

A apresentação dos dados de monitoração e validação é fundamental para que a ferramenta possa prestar auxílio na compreensão e depuração do comportamento temporal do sistema investigado. A possibilidade de diversas representações aborda diferentes visões de comportamento temporal, permitindo avançar na direção da compreensão do sistema como um todo, e da visualização de interrelações temporais entre componentes do sistema e sua comunicação ([TSA96], [MCD89]). Benefícios auferidos por estratégias de visualização adequadas são apoio a depuração e identificação de pontos críticos de desempenho temporal, assim como comportamento errôneo detectado e ressaltado [TSA95]. Exige-se de tais estratégias de visualização que sejam corretas e eficientes, no sentido de abstrair dados significativos dentre o volume de informação disponível e permitir a identificação de interações e interrelações entre componentes do sistema e sua comunicação.

A partir destas premissas de importância da visualização para os dados resultantes da validação, foram propostos algumas estratégias de visualização dos dados, abordando algumas visões estáticas do comportamento temporal do sistema

investigado. Foram propostas a apresentação textual da validação, e gráficos de área, histogramas de validação e diagramas de Gantt validados (ver Figura 4.11 até Figura 4.17).

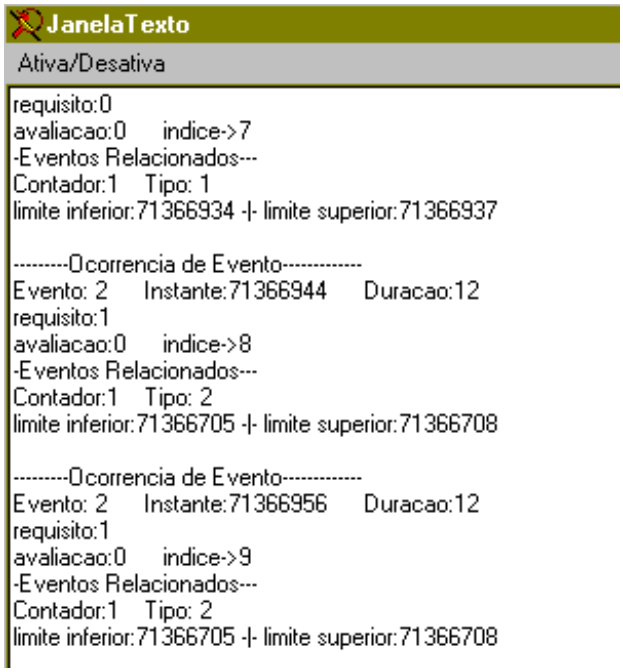

Figura 4.11 – Visualização textual

A apresentação textual (Figura 4.11) expõe diretamente a situação da validação para cada ocorrência de evento. Apresenta as seguintes informações:

- o evento que ocorreu,
- o instante de ocorrência,
- a duração da mensagem correspondente,
- o requisito ao qual este evento está relacionado,
- a avaliação deste requisito,
- o índice absoluto de ocorrência deste evento, isto é, quantos eventos deste tipo ocorreram no sistema até então.

Apresenta ainda eventos relacionados a este por requisitos, isto é, os eventos para os quais os requisitos que contém o evento ocorrido geram intervalos de validação. São apresentados:

- quais são estes eventos,
- o intervalo gerado pela validação, e

- um “contador”, para que seja acompanhado o número de requisitos que se referem ao evento ocorrido e que geram intervalos para o mesmo.

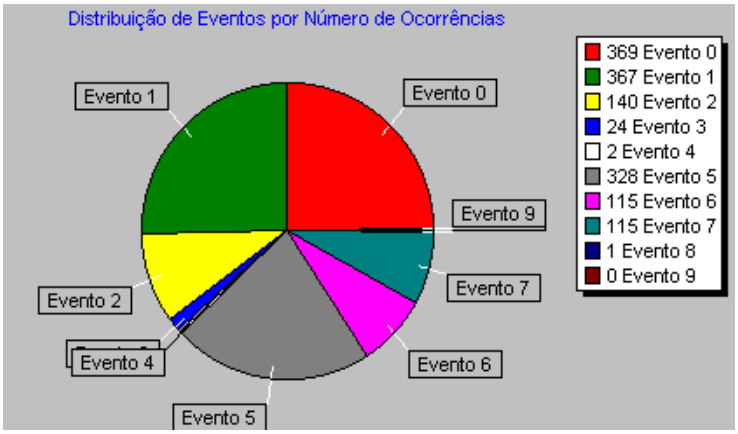

Figura 4.12 – Área circular, número de ocorrências

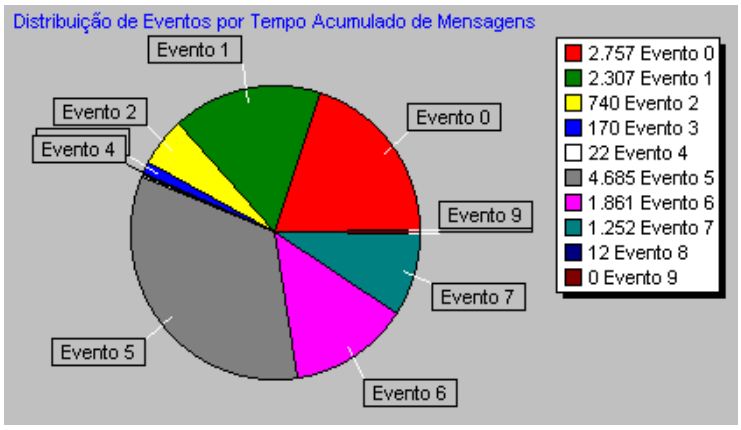

Figura 4.13– Área circular, tempo total

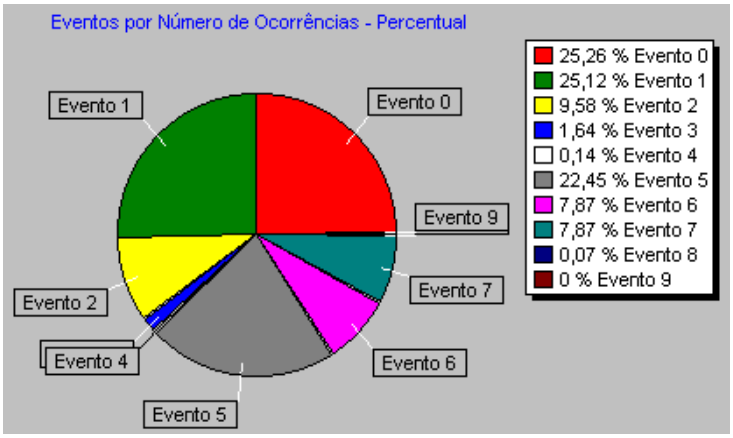

Figura 4.14 – Área circular, quantidade percentual

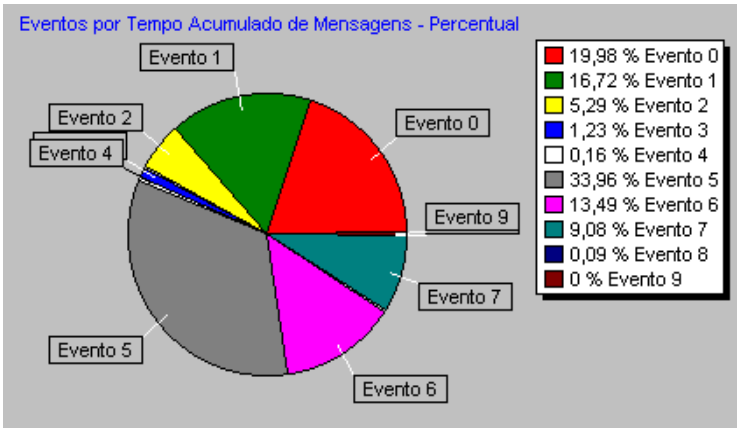

Figura 4.15 – Área circular, tempo percentual

O diagrama tipo "área" expõe de maneira visual a participação relativa dos diversos eventos na comunicação: um setor de círculo, cuja área é proporcional à quantidade ou ao tempo total de eventos de cada tipo. São disponibilizados quatro visualizações: uma com as áreas proporcionais ao número de ocorrências e uma lista com os números totais de mensagens (Figura 4.12), uma com as áreas proporcionais ao tempo relativo de cada evento e uma lista com o somatório de tempos de cada evento (Figura 4.13), uma com as áreas proporcionais ao número de ocorrências e uma lista com o percentual de número de ocorrências de cada evento (Figura 4.14), e uma com as áreas proporcionais ao tempo relativo e uma lista com o percentual de tempo total despendido com cada evento (Figura 4.15**Erro! A origem da referência não foi encontrada.**).

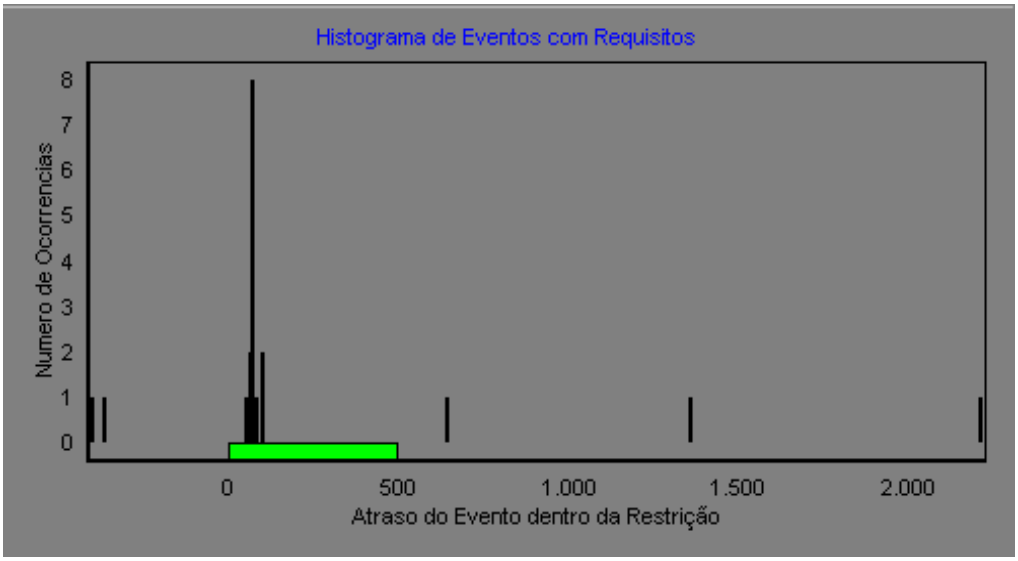

Figura 4.16 – Histograma de Eventos e Requisitos

O histograma de evento e requisito (Figura 4.16) expõe a situação temporal das ocorrências de um evento em relação ao intervalo de tempo que representa o requisito. Neste diagrama, o instante temporal “0” é o início do intervalo permitido pelo requisito para a ocorrência; o intervalo permitido é marcado com uma barra verde, e a altura da barra do histograma representa o número de ocorrências com um determinado atraso em relação ao intervalo permitido. Por exemplo, se a barra verde vai até o valor “5ms” e existe uma barra vertical, de altura 8 na abcissa 3ms, isto significa que o intervalo de tempo que o requisito permite para ocorrências deste evento tem 5ms de extensão, e que 8 ocorrências deste evento tiveram lugar dentro do intervalo permitido, 3ms após o primeiro instante permitido e 2ms antes do último prazo para este evento.

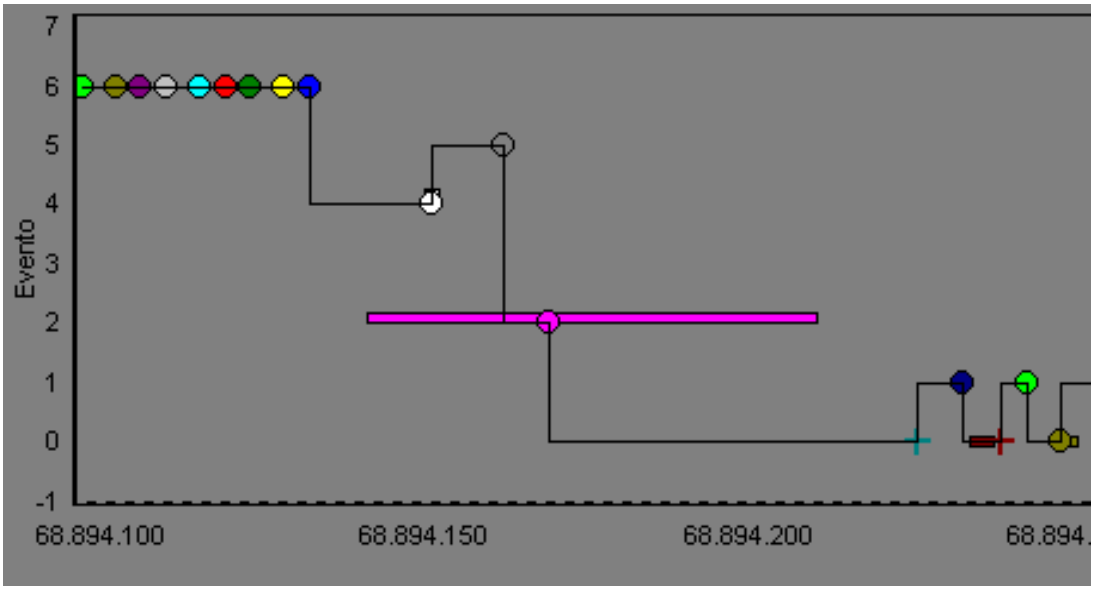

Figura 4.17 – Diagrama de Gantt validado

O diagrama de Gantt (Figura 4.17) é uma representação temporal tradicional, e muito expressivo visualmente para compreensão de ordenação de diversos eventos entre si, sendo por estes motivos muito utilizado (ver por exemplo [BAK91], [SHA89]). Este diagrama reserva para cada evento uma ordenada; o eixo horizontal representa o instante de tempo medido. Para cada ocorrência de um evento é marcado um pequeno círculo; se existe algum requisito associado a este evento, uma barra horizontal da mesma cor do círculo indica o intervalo de tempo permitido por este requisito; se porventura a temporização do evento desrespeitar o requisito (isto é, se acontecer “fora da barra”), o círculo é substituído por uma cruz. A linha indica o fluxo de controle, isto

é, qual mensagem está sendo transmitida no momento, e conduz para a linha da próxima mensagem.

Como a natureza estrutural da ferramenta desenvolvida é essencialmente modular, outras propostas de visualização podem facilmente ser incorporadas. As várias interpretações gráficas da validação temporal levam o projetista a uma maior compreensão do funcionamento do sistema, assim como podem servir de guia para adaptar ou modificar a implementação das estratégias de controle.

5. Implementação

5.1 Descrição Estrutural

A ferramenta proposta foi implementada como um programa de computador, possibilitando ao usuário efetuar as análises e obter resultados conforme abordagem exposta no capítulo anterior. O presente capítulo apresenta alguns detalhes sobre esta implementação e suas estruturas de dados.

A implementação da ferramenta é composta de duas partes (ver Figura 5.1). Um programa executável contém a interface com o usuário, através do qual este interage com a ferramenta, e o conjunto de módulos de visualização, através do qual os resultados da validação são expressos. Uma biblioteca de funções, implementada como uma biblioteca de ligação dinâmica (*dynamic link library*, DLL), contém o núcleo de estruturas de dados e funcionalidades da validação propriamente dita. A biblioteca de funções, quando acionada pelo programa executável, passa a este estruturas de dados contendo a validação efetuada; estas estruturas são passadas a um módulo de visualização, que as utiliza na construção de sua apresentação.

A ferramenta, constituída pelas duas partes descritas, relaciona-se com uma série de arquivos que contém dados necessários à validação (ver novamente Figura 5.1). Deve existir um arquivo de inicialização, identificado pelo nome *vcat.ini*, que contém uma associação entre eventos e mensagens. A associação é estabelecida com base em características da mensagem: valor do campo de controle e endereço destino da mensagem. Este arquivo é lido uma vez pela ferramenta quando de sua inicialização, sendo então as associações estabelecidas. A validação é efetuada de acordo com estas associações, durante a execução da ferramenta. Além deste arquivo de inicialização, são

requeridos um arquivo com os eventos amostrados e seus instantes de ocorrência (identificado pela extensão *.fbv*) e um arquivo com os requisitos a serem validados (identificado pela extensão *.rst*). O usuário indica, através da interface proporcionada pela ferramenta, qual arquivo de registro e de requisitos deseja validar (Figura 5.2).

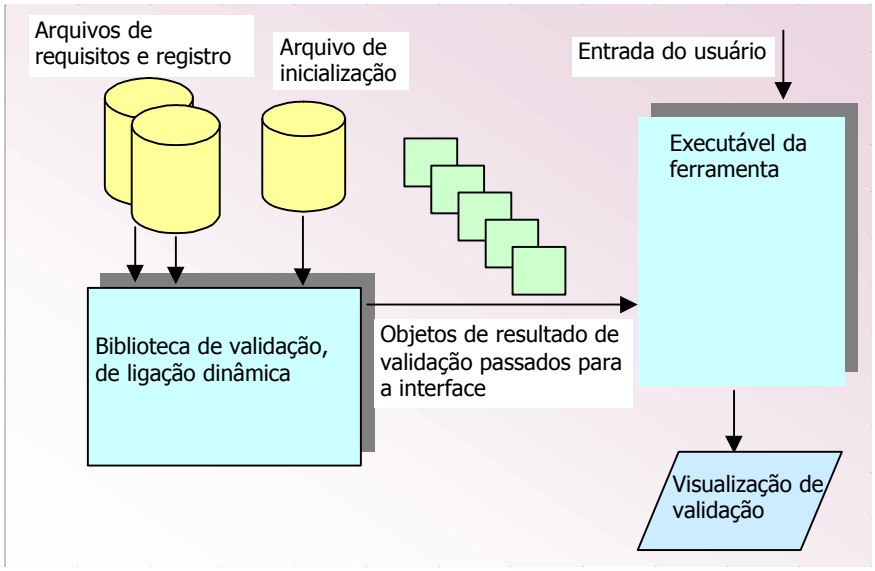

Figura 5.1 – Estruturas de dados da ferramenta

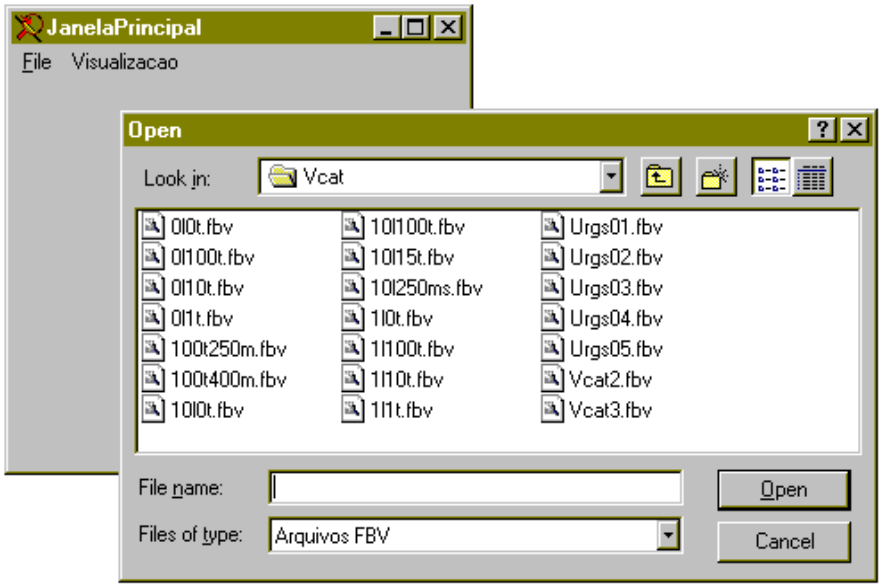

Figura 5.2 – Interface de seleção de arquivos

Conforme foi apresentado no capítulo anterior, a ferramenta desenvolvida possui uma arquitetura funcional constituída por uma entrada de eventos, um histórico para eventos, um validador de requisitos e uma saída de resultados na forma de visualização gráfica. Construtivamente, a ferramenta foi desenvolvida orientada a objetos: estruturas

de dados e blocos funcionais foram definidos como classes, e interfaces definem a interação dos objetos entre si. Por fim, a aplicação como um todo foi desenvolvida a partir das definições de classes.

A ferramenta foi implementada como uma aplicação Windows 32 bits utilizando-se dois ambientes de programação distintos. Os objetos relacionados a eventos, requisitos e validação, foram implementados em C++ e compilados como uma DLL (biblioteca de ligação dinâmica). A interface com o usuário e os módulos de visualização foram construídos no ambiente Delphi (linguagem Object Pascal), utilizando os objetos gráficos disponíveis para desenvolvimento. A interface entre os dois blocos da aplicação é feita através de funções da biblioteca e do compartilhamento de objetos (os objetos validação).

5.2 Objetos que constituem a ferramenta

A Figura 5.3 apresenta o diagrama de arquitetura abordado no capítulo . Esta arquitetura representa a funcionalidade desejada para uma implementação da proposta da ferramenta. O diagrama relaciona os dados a serem investigados pela ferramenta (requisitos e registro de eventos) como entradas do sistema, e a saída do sistema, que é a visualização dos dados resultantes da validação. A proposta descrita foi modelada e implementada com orientação a objetos, e as estruturas de dados modeladas e implementadas (os objetos computacionais) serão descritos a partir de sua inserção no diagrama de arquitetura.

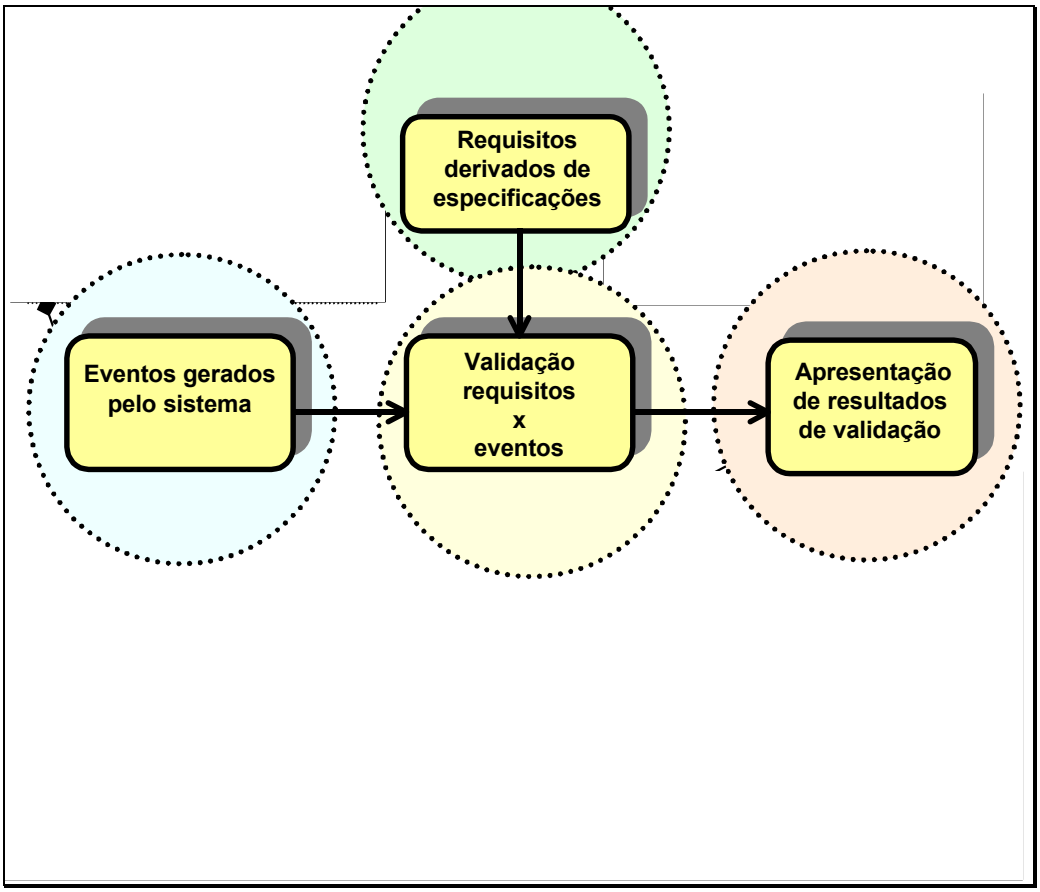

Figura 5.3 – Arquitetura do sistema

O componente de arquitetura "*Eventos gerados pelo sistema*" (Figura 5.4) consiste de uma série de objetos destinados a repassar à validação as ocorrências de eventos registrados em um arquivo. As ocorrências de eventos são modelados pelo objeto *C_Evento*, que tem atributos de tipo de evento a que pertence a instância do objeto, o registro de instante de tempo de ocorrência do evento, e os endereços de origem e destino da mensagem. O objeto *C_Monitor* é capaz de varrer um arquivo de registros de eventos, gerar instâncias de objeto *C_Evento* e passá-las para a validação. Este objeto encapsula a funcionalidade necessária para interpretar o arquivo de registros de mensagens, em termos de protocolo de mensagens e de formato de arquivo. Ou seja, este objeto deve ser adaptado para que a ferramenta seja capaz de validar mensagens em outro protocolo qualquer (por exemplo, Profibus).

O objeto *C_Historico* contém agregado uma fila circular (de tamanho finito) de objetos *C_Evento*, e representa a coleção de um determinado número de ocorrências mais recentes de um certo evento. A ferramenta possui um conjunto de instâncias de *C_Historico*, uma instância para cada tipo de evento que deve ser armazenado. As

instâncias de objeto *C_Evento* são passadas a instâncias de *C_Historico*, para que passem a fazer parte da fila de ocorrências de um determinado evento. A instância de *C_Historico*, se consultada, pode informar a ocorrência de evento presente em sua fila, de acordo com um índice regressivo a partir da ocorrência atual, até o evento mais antigo presente na fila (e que depende do tamanho da fila).

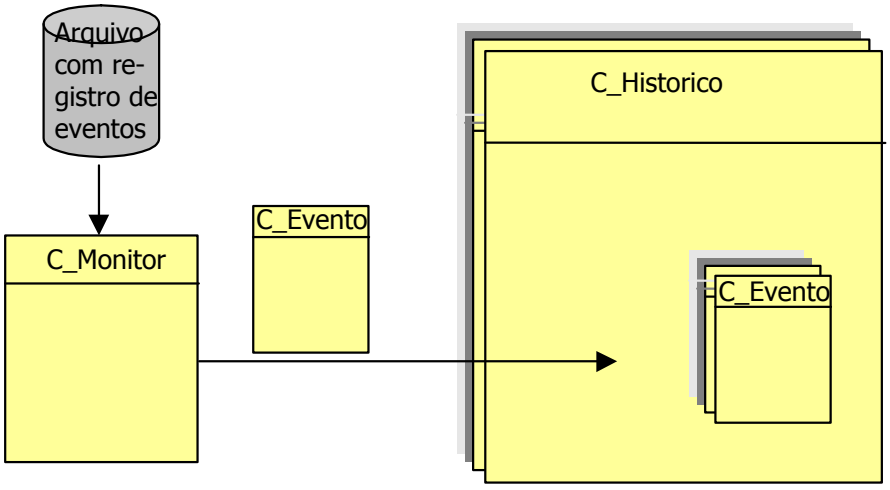

Figura 5.4 – Estruturas de dados associadas ao registro de eventos

O componente da arquitetura "*Requisitos derivados de especificações*" corresponde a uma série de instâncias de um objeto *C_Requisito* (ver Figura 5.5). Cada uma destas instâncias contém um requisito temporal, que é lido de um arquivo gerado pelo usuário e que expressa os requisitos para a validação. O objeto tem a capacidade de, conhecendo o conjunto de instâncias de objeto *C_Historico* contendo o registro de ocorrências de eventos, efetuar a validação do requisito que expressa. Esta funcionalidade é uma das mais relevantes para a proposta da ferramenta.

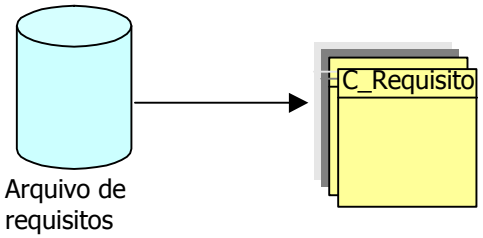

Figura 5.5 – Estruturas de dados associadas a especificação de requisitos

O componente da arquitetura "*Validação de requisitos e eventos*" consiste basicamente de um objeto validador, *C_Validador*. Este objeto contém a estrutura necessária para a validação de eventos e requisitos e geração das estruturas de dados que contêm os resultados. O objeto *C_Validador* contém a coleção de históricos de

ocorrências de eventos, e também o conjunto de instâncias de objetos requisito. A funcionalidade principal associada a este objeto é invocar a validação de requisitos (nos objetos *C_Requisito*) relacionados a ocorrências registradas no histórico. O objeto validador, a partir do resultado da validação, cria uma instância de um objeto *C_TRE* (tupla requisito-evento), que contém o resultado da validação, e referências para a ocorrência do evento e para o requisito que foram validados, além de um intervalo que descreve a validade de evento descrito no requisito (ver capítulo seção). É importante notar que para uma ocorrência pode ser gerado mais de um objeto TRE, uma vez que a ocorrência pode estar relacionada a (ou seja, ser mencionada em) mais de um requisito. É gerada, desta forma, uma validação para cada par evento-requisito. O objeto TRE é então introduzido em uma fila encadeada, que contém uma série de validações. São os objetos desta fila que são passados para o módulo que gera a visualização dos resultados da validação.

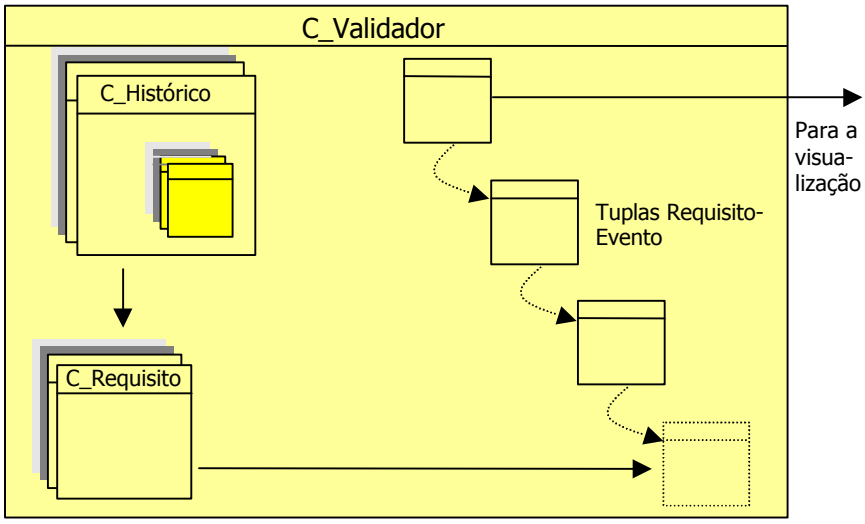

Figura 5.6 – Estruturas de dados associadas à validação

Por fim, o componente da arquitetura "*Apresentação dos resultados da validação*" contém um conjunto de objetos que constituem a interface com o usuário e um conjunto de objetos responsáveis pela visualização da validação (ou seja, os módulos de visualização), conforme apresentado na Figura 5.7. A interface com o usuário é construída com a utilização de módulos padrão fornecidos pelo ambiente de desenvolvimento (tais como janelas, menus e botões). Implementa com o auxílio destes módulos padrão as funcionalidades de invocação da ferramenta, seleção de arquivos fonte de dados, e de invocação dos diversos módulos de visualização. Este conjunto tem

a função também de passar os dados referentes a escolhas do usuário para os objetos de validação e visualização, configurando assim a atuação destes objetos.

A visualização também é construída a partir de módulos disponibilizados pelo ambiente de desenvolvimento utilizado. Cada módulo de visualização, ao ser invocado, passa a acionar o objeto validador e a receber uma seqüência de instâncias de objetos TRE, correspondendo à validação efetuada. De acordo com a forma como a validação será visualizada, o módulo extrai as informações contidas nos TRE, e efetua algum tipo de manipulação adicional sobre estas quando necessário (por exemplo, calculando a diferença temporal entre o início do intervalo de validade do requisito e a ocorrência do evento especificado). A seguir, estes dados, tanto aqueles provenientes diretamente dos TRE quanto aqueles eventualmente manipulados, são armazenados em um objeto *container* série cuja função é exatamente conter dados seriados. Um objeto especializado utiliza o *container* série como fonte de dados na construção de uma representação gráfica – cujo formato depende da especialização do objeto, podendo ser, por exemplo, um gráfico de área circular, ou de barras, ou de linhas.

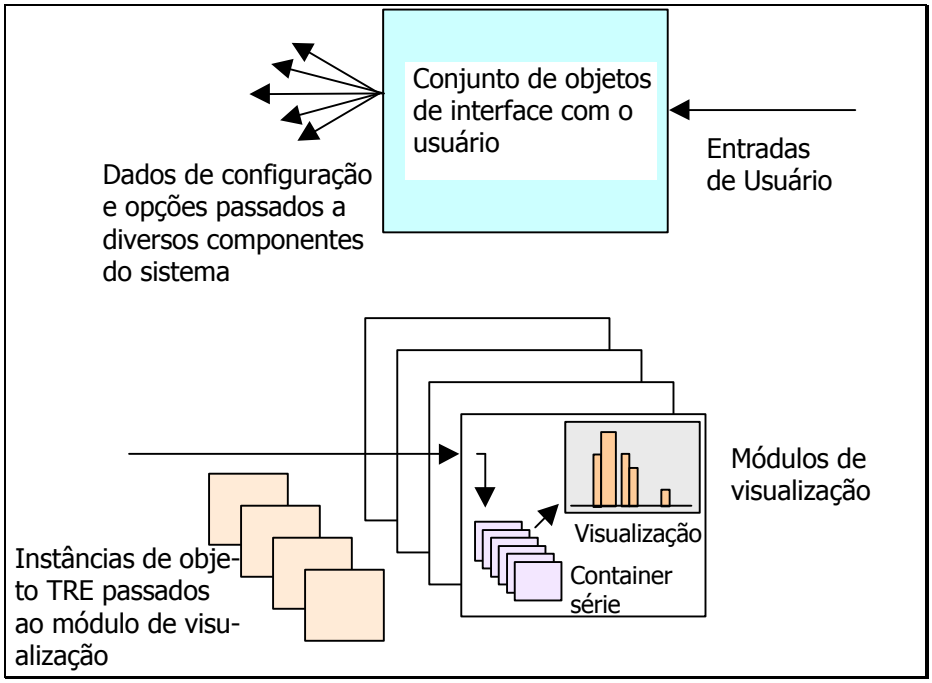

Figura 5.7 – Estruturas de dados associadas à visualização e interface de usuário

6. Estudos de Caso

A ferramenta proposta e implementada foi aplicada à análise de estudos de caso que serão detalhados neste capítulo. Estas análises permitiram avaliar a proposta com relação à capacidade de esclarecer o comportamento temporal dos sistemas avaliados e de gerar informação útil ao projetista de sistema (implementação de protocolo) e de aplicação. Os estudos de caso consistem em aplicações *Foundation Fieldbus* para controle de uma planta industrial. A configuração da aplicação dos estudos de caso varia desde um simples controle de nível até um sistema mais complexo, permitindo acompanhar as alterações no comportamento temporal do barramento em função da complexidade da aplicação e da quantidade de dados que devem ser comunicados. Serão estudados casos com diferentes quantidades de dados periódicos, correspondendo ao número de ligações entre blocos funcionais, e também com diferentes quantidades de dados aperiódicos, relacionados principalmente a dados de supervisão do processo da aplicação. Serão abordados principalmente a distribuição proporcional entre os diferentes tipos de mensagens, a avaliação da periodicidade das mensagens periódicas e o comportamento temporal das mensagens aperiódicas. Serão apresentadas também conclusões sobre a análise pela ferramenta proposta e o comportamento temporal dos sistemas investigados.

6.1 Estudo de Caso 1 – Controle de Nível

Inicialmente a ferramenta foi utilizada na validação de uma arquitetura simples, constituída por um laço de controle de nível de líquido em um tanque. Dois dispositivos implementam esta aplicação: um transmissor de pressão absoluta (manométrica), calibrado para medição de nível, e um posicionador de válvula, com a missão de controlar o fluxo de líquido (ver Figura 6.1). A válvula com controle manual constitui-

se no fluxo de entrada para o tanque. Para que o nível seja mantido no ponto de operação, o sistema deve controlar o fluxo de saída através da válvula controlada V.

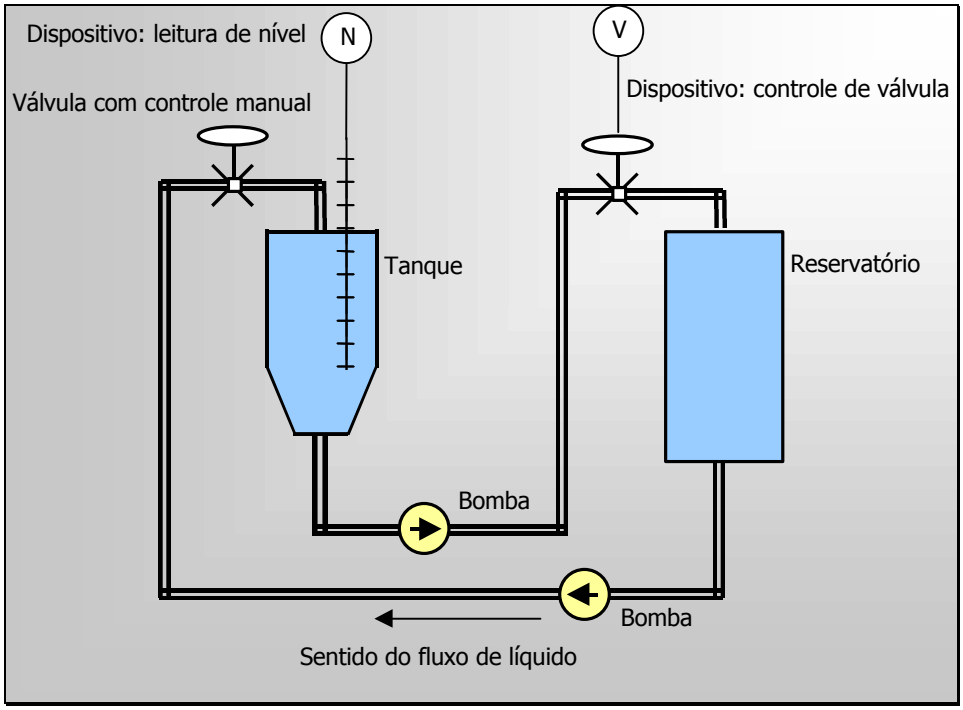

Figura 6.1 – Sistema a ser controlado pela aplicação

A aplicação laço de controle foi implementado com o auxílio de três blocos funcionais: uma Entrada Analógica (EA), que disponibiliza para o sistema a leitura do sensor do dispositivo de pressão, um controlador PID, e uma saída analógica (SA), que transfere para o atuador o valor de atuação adequado. Estes blocos foram conectados da maneira usual para compor um controle PID realimentado (Figura 6.2), em que a entrada do bloco PID é originada no bloco EA, e a sua saída (valor de atuação) é enviada para o bloco SA. Dados em um dispositivo que devem ser comunicados para outro dispositivo são publicados como mensagens periódicas no barramento. O bloco funcional EA foi configurado para rodar no dispositivo transmissor, enquanto que o controle PID e o bloco SA foram alocados no posicionador de válvula. É importante observar que o bloco PID poderia também ser alocado para rodar no transmissor de temperatura. Esta configuração alternativa, apesar de não interferir na funcionalidade da aplicação, pode influenciar o comportamento temporal da comunicação, pelo fato de alterar as mensagens intercambiadas no barramento. O escalonamento das mensagens foi gerado automaticamente através de uma ferramenta de configuração, a qual define o ciclo da comunicação periódica entre os blocos funcionais EA e PID.

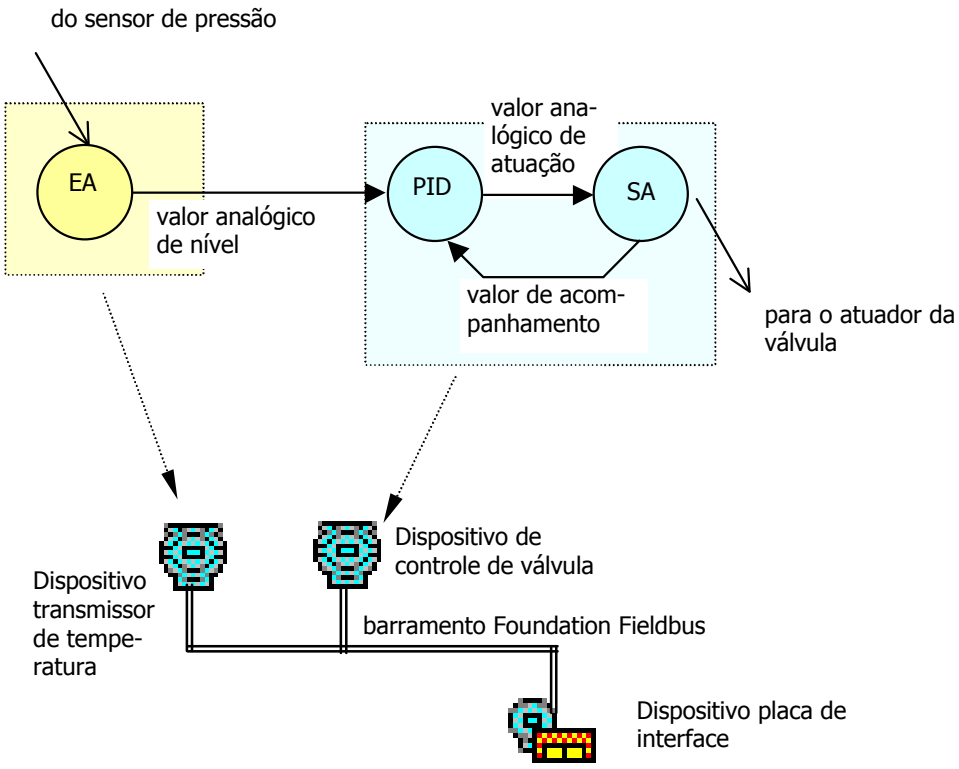

Figura 6.2 – Exemplo de implementação da aplicação proposta

O valor de temperatura, disponibilizado pelo bloco EA, é transmitido ao bloco PID de acordo com o seguinte procedimento, comum a todos os dados periódicos de blocos funcionais: (i) o dispositivo LAS (Escalonador Ativo do Enlace, o mestre da rede), ao chegar o momento configurado de transmissão, envia ao transmissor a mensagem *Invoca Dados* (*Compel Data*, CD); (ii) em resposta a esta mensagem o transmissor publica o valor de saída do bloco EA no barramento, utilizando a mensagem *Publicação de Dados* (*Data*, DT). Conforme já explicado no capítulo introdutório sobre *Foundation Fieldbus*, além das mensagens periódicas, escalonadas a priori e adequadas a comunicação de dados com restrições temporais, existem mensagens que comunicam dados sem restrições temporais (aperiódicos) da aplicação, que nos casos apresentados constituem-se principalmente em variáveis (parâmetros) do sistema monitoradas por um supervisório. Estas mensagens aperiódicas são comunicadas de acordo com o seguinte procedimento: (i) o dispositivo LAS, quando há um intervalo de tempo sem mensagens periódicas escalonadas, envia ao transmissor a mensagem *Passagem de Permissão* (Pass Token, PT); (ii) em resposta a esta mensagem o transmissor publica os dados configurados para transmissão, utilizando uma ou mais mensagens *Publicação de Dados* (DT); (iii) após a mensagem DT, ou se não houver nenhum dado a ser

transmitido, o transmissor envia ao LAS a mensagem *Devolução de Permissão (Return Token*, RT).

A fim de avaliar o comportamento temporal do sistema, as mensagens acima, juntamente com outras mensagens relacionadas ao gerenciamento da comunicação e utilizadas pelo dispositivo LAS, foram associadas a nomes de evento para serem identificadas pela ferramenta. A associação consta da Tabela 6.1, em que estão os mnemônicos das mensagens conforme *Foundation Fieldbus*, os tipos de mensagem a que se referem, e o nome de evento atribuído. É interessante observar que alguns tipos de mensagem com função semelhante compartilham mnemônicos diferenciados por um número (como é o caso de DT1, pela norma "Publicação de Dados Aperiódicos" e DT3, "Publicação de Dados Periódicos).

Mnemônico FF	***Tipo de mensagem***	***Nome de evento***
PT	Passagem de permissão aperiódica	E0
RT	Retorno de permissão aperiódica	E1
Idle	Barramento ocioso	E2
PN	Teste de endereço (para ativação de estação)	E3
TD	Distribuição de tempo (sincronização do barramento)	E4
DT1	Publicação de dados aperiódicos	E5
CD2	Invoca dados periódicos	E6
DT3	Publicação de dados periódicos	E7

Tabela 6.1 – Associação de mensagens a nomes de evento

6.1.1 Configuração sem ligações entre blocos

a) Sem comunicação de dados para supervisório

Com o objetivo de investigar a comunicação entre os dispositivos constituída pelas mensagens de gerenciamento de sistema e de dados aperiódicos, a aplicação com seus blocos funcionais foi configurada nos dispositivos, entretanto a princípio sem a ligação entre blocos. O estudo deste caso permite observar como as mensagens periódicas serão inseridas na comunicação pelo escalonamento do LAS. Neste caso não ocorrem as mensagens CD2, nem os respectivos DT3, correspondentes a transmissão de

dados periódicos. A análise de um registro de mensagens revela que a maior parte das mensagens circuladas referem-se a passagem e retorno de permissão (sem dados a serem transmitidos) - eventos E0 e E1 - e mensagens de barramento ocioso – evento E2. A Figura 6.3 mostra também que nesta situação ocorrem as mensagens do tipo teste de endereço – E3 – e outras, em menor número, tais como sincronização de barramento – E4 – e publicação de dados aperiódicos – E5.

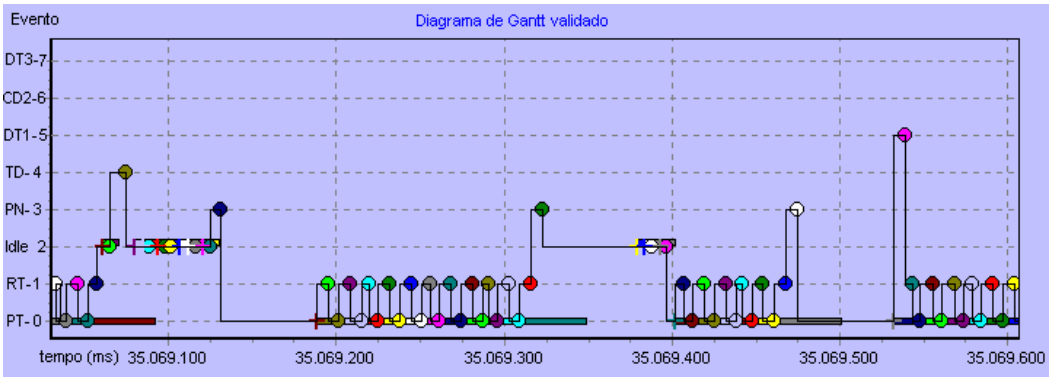

Figura 6.3 – Diagrama de Gantt, caso 1.1a

Uma análise percentual resulta na distribuição de mensagens apresentado na Figura 6.4. Nota-se a predominância das mensagens tipo PT, RT e Idle – E0, E1, E2 – que é observada no diagrama de Gantt da Figura 6.3. Como não estão configuradas as ligações entre blocos, as mensagens CD e DT3 – E6, E7 – estão completamente ausentes deste registro. Além disto, o evento que se refere a publicação de dados aperiódicos é pouco presente, restringindo-se a parâmetros para gerenciamento da aplicação e da comunicação (DT1, E5) – ou seja, os dispositivos retornam a permissão logo após a receber.

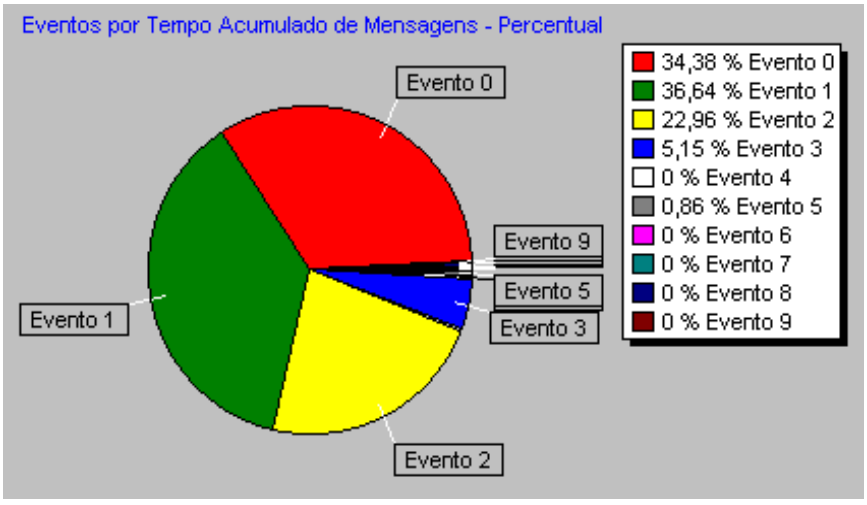

Figura 6.4 – Tempo percentual, caso 1.1a

b) Com comunicação de dados para supervisório

Após esta análise, foi configurado um supervisório ligado ao barramento, para ler uma determinada quantidade de parâmetros da aplicação e criar tráfego de mensagens aperiódicas. Foram configurados 100 parâmetros diferentes para serem atualizados pelo sistema. A nova distribuição de mensagens pode ser observada na Figura 6.5. A ocupação do barramento pela mensagem DT1 (E5), utilizada para comunicar os dados aperiódicos requisitados, torna-se significativa. Entretanto, mesmo com o aumento do tráfego de dados no barramento, a mensagem Idle (E2) continua ocupando praticamente na mesma proporção o tempo total de comunicação.

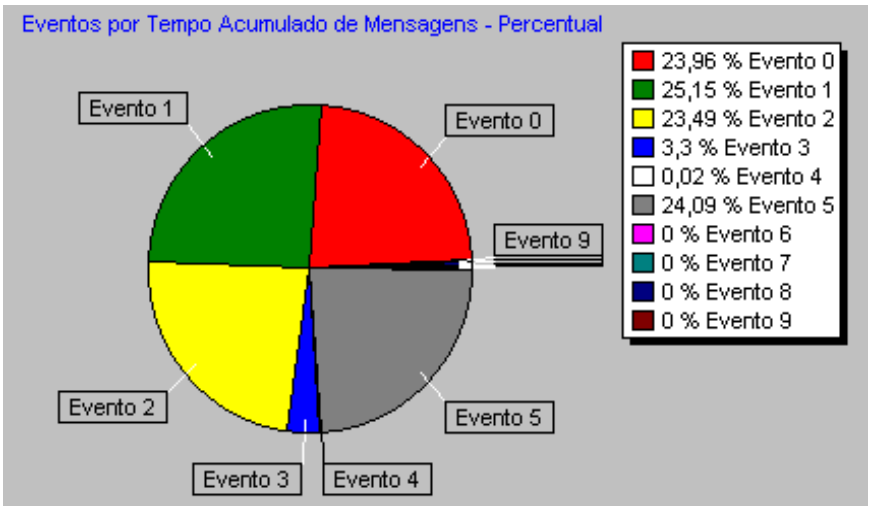

Figura 6.5 – Tempo percentual, caso 1.1b

6.1.2 Configuração com ligação entre blocos

a) Sem passagem de dados para supervisório

Em seguida a primeira experiência sem ligações entre blocos, a aplicação foi configurada em sua forma definitiva, incluindo-se a ligação entre o bloco EA no dispositivo transmissor de nível e o bloco PID no controlador da válvula. A distribuição de mensagens em um registro de uma situação sem parâmetros monitorados por supervisório pode ser observada na Figura 6.6. A diferença mais relevante em relação à situação sem ligação entre blocos é a presença de mensagens CD e DT3 (E6 e E7).

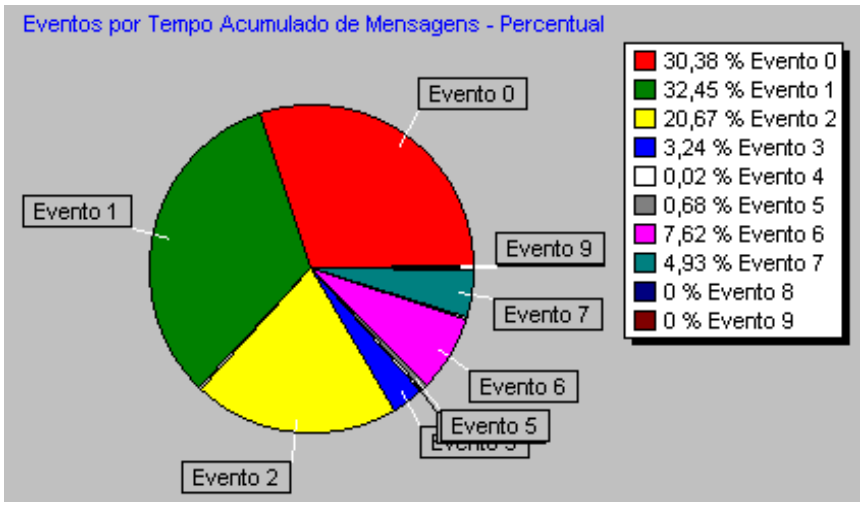

Figura 6.6 – Tempo percentual, caso 1.2a

É interessante ressaltar o fato de que o tempo de duração da mensagem é computado desde o fim da última mensagem, incluindo assim o período de silêncio que porventura ocorrer antes da efetiva transmissão de sinal no barramento. A duração deste período de silêncio é utilizado pelo LAS para ajustar o instante correto de ocorrência (término) da mensagem, e é por este motivo que, em duração de mensagens, o evento CD ultrapassa DT3, mesmo que este último evento contenha uma maior quantidade de dados. A comparação entre a Figura 6.7 e a Figura 6.6 permite observar este fato; esta última demonstra a efetiva paridade entre eventos E6 e E7 em termos de número de mensagens transmitidas.

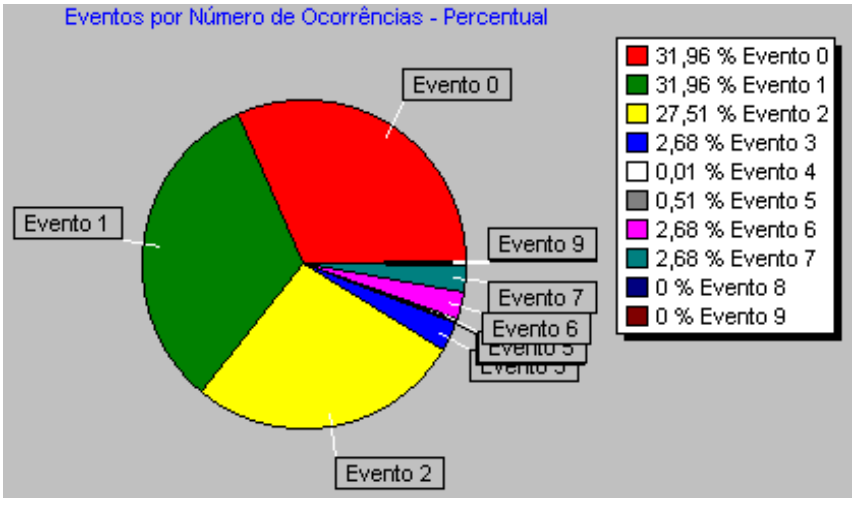

Figura 6.7 – Quantidade percentual, caso 1.2a

O macrociclo de publicação de dados periódicos da ligação entre os blocos EA e PID é definido pela ferramenta de configuração, levando em conta fatores tais como

ocupação do barramento, tempo de processamento de cada bloco e parâmetros de cada bloco, e não pode ser diretamente alterado pelo usuário. Para esta aplicação, o macrociclo definido e apresentado pela ferramenta de configuração é de 275ms. Para avaliar esta periodicidade, foi especificado o requisito

(CYCLIC E7@-1) [275, 1]

significando que para este evento é esperado um comportamento periódico, com período 275ms, e variação máxima de periodicidade de ±1ms. A Figura 6.8 apresenta o resultado da avaliação deste requisito. Neste histograma, períodos de 274ms (correspondendo a atraso 0) a 276ms (correspondendo a atraso 2) estão sobre a área verde de intervalo admissível para a ocorrência de eventos. Observa-se que na verdade, a periodicidade efetiva é de 276ms, com uma variação significativa entre períodos de 275ms a 277ms, e algumas ocorrências de desvios maiores, chegando a 282ms no máximo e a 270ms no mínimo.

Figura 6.8 – Histograma de mensagem periódica, caso 1.2a

Além do macrociclo, a ferramenta de configuração define e apresenta um tempo para ocupação de segundo plano ("*background*") o qual, entre outros fins, reserva uma parte do macrociclo para a passagem de dados aperiódicos que porventura existirem para publicação. Este tempo pode ser alterado pelo usuário. Entretanto, se este tempo for aumentado além de certo limite, a ferramenta de configuração redefine (aumenta) o tempo de macrociclo de forma a acomodar uma tal previsão de tráfego. Nos presentes casos de estudo, exceto quando mencionado, o tempo de ocupação de segundo plano foi deixado como definido pela ferramenta de configuração. No caso da presente aplicação,

para o macrociclo de 275ms foi definido um tempo de ocupação de segundo plano de 100ms, em torno de 36%. Este dado é mais importante quando existe uma grande quantidade de dados aperiódicos para ser transmitida, por que influencia o tempo disponível para a publicação de tais dados.

Para efeito de comparação com outras configurações, foi redefinido o evento E0 como a passagem de permissão para um dispositivo específico, no presente caso, o dispositivo transmissor de nível. Desta maneira é possível avaliar quanto tempo um determinado dispositivo, dispondo de uma informação a ser comunicada, precisará esperar para receber novamente permissão para publicá-la. Este evento foi validado para o requisito

(E0@-1 AFTER E0@-2) [0, 40]

resultando no histograma apresentado na Figura 6.9. No caso examinado, pode-se observar dois picos na distribuição do atraso entre duas ocorrências do evento. Um destes possui uma concentração elevada de ocorrências em torno de 36ms, motivando a escolha do intervalo de validação proposto. Um outro, menor, está em torno de 169ms. As ocorrências estão mais distribuídas (isto é, menos concentradas) do que no caso das mensagens periódicas, justamente pelo fato de serem aperiódicas, sem restrições de tempo impostas pelo sistema.

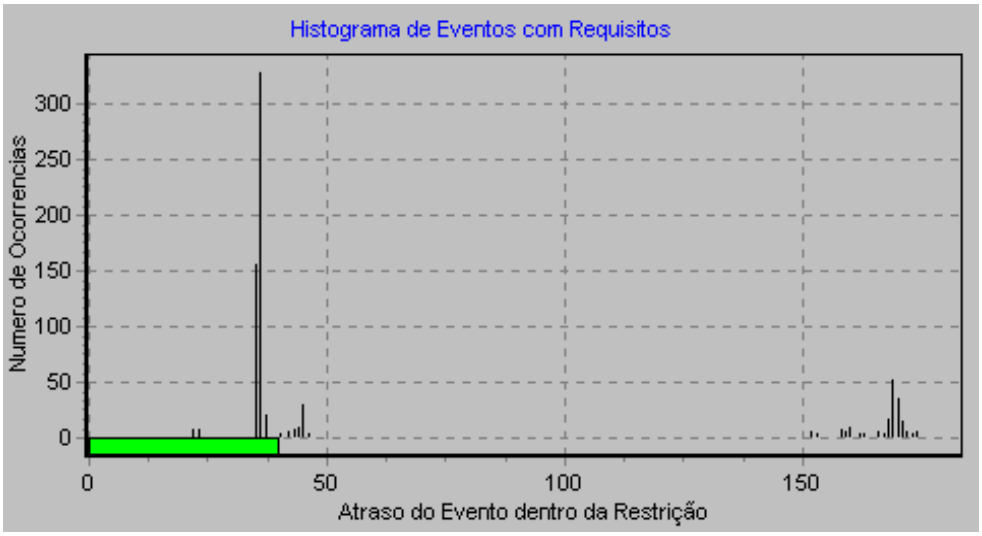

Figura 6.9 – Histograma de mensagem aperiódica, caso 1.2a

O primeiro pico na distribuição dos atrasos está relacionado à rotação das passagens de permissão entre os dispositivos presentes no barramento. No presente caso, além dos dois dispositivos mencionados (transmissor de nível e controlador de

válvula) está presente o dispositivo placa de interface (do microcomputador que roda o configurador do sistema). As mensagens de passagem e retorno de permissão têm uma duração de cerca de 6ms cada; para cada um dos três dispositivos ocorre um conjunto de passagem e retorno de permissão, levando a um tempo de aproximadamente

T = 3 x (6 + 6) = 36

36ms entre duas ocorrências sucessivas da passagem de permissão a um endereço específico, nesta configuração, e sem outras mensagens ocorrendo entre estas duas ocorrências. Já o segundo pico está relacionado à existência de mensagens periódicas que interrompem o ciclo de rotação de permissão aperiódica. A diferença entre os atrasos dos picos observados, em torno de 136ms, é constituída pela duração das mensagens periódicas (CD e DT3), mais um conjunto de mensagens Idle, mais uma mensagem de teste de endereço e sua respectiva espera por resposta, conforme pode ser observado no diagrama de Gantt da Figura 6.10 (no qual o evento E0 está definido como PT genérico). Também a partir da Figura 6.9 pode-se observar que, para este caso, o atraso máximo entre dois eventos PT para o mesmo dispositivo fica em torno de 180ms.

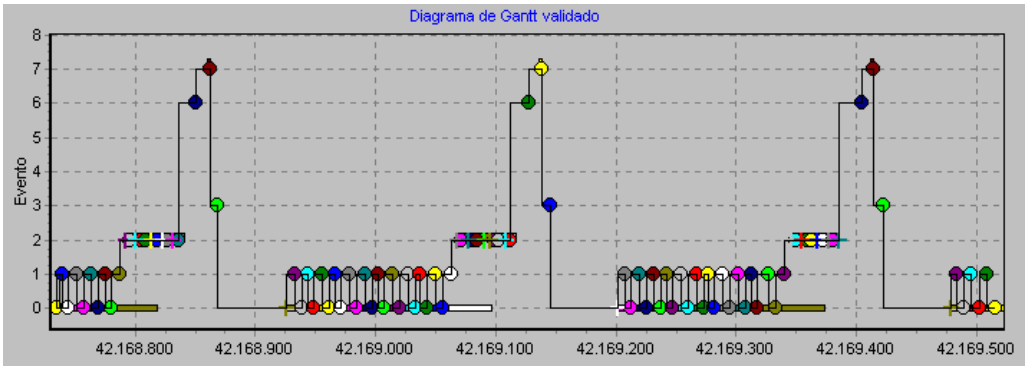

Figura 6.10 – Diagrama de Gantt, caso 1.2a)

b) Caso com comunicação de dados para supervisório

O mesmo sistema foi configurado para comunicar parâmetros a um sistema supervisório. Foram configurados , novamente, para comunicação, 100 parâmetros a serem atualizados. A distribuição de mensagens resultante está apresentada na Figura 6.11. Pode-se observar, em relação ao sistema sem monitoração de parâmetros (Figura 6.6), o aparecimento de uma proporção significativa de mensagens do tipo DT (E5), que contém os dados aperiódicos comunicados (parâmetros). Além disto, é interessante notar a diminuição no tempo destinado a mensagens de barramento ocioso (E2), embora

sua proporção permaneça relevante. A proporção das mensagens ligadas à comunicação periódica permanece praticamente inalterada (E6 e E7), como era de se esperar.

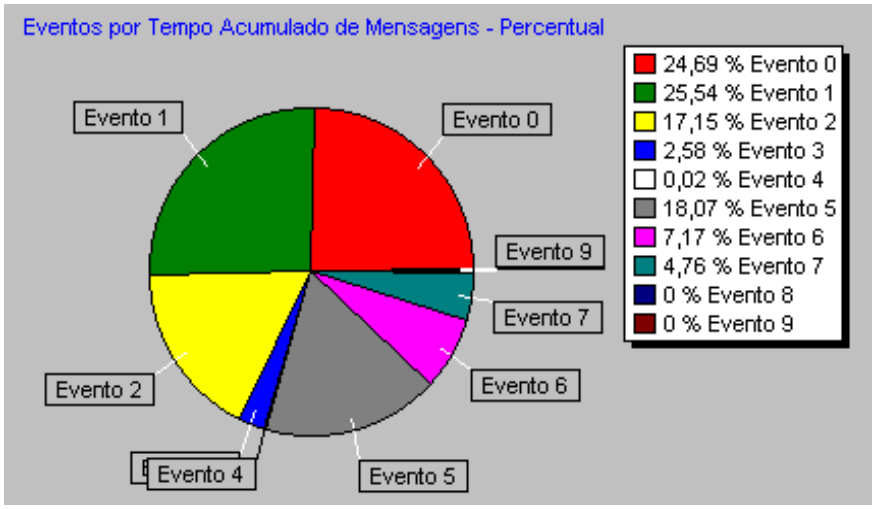

Figura 6.11 – Tempo percentual, caso 1.2b

Para este caso, também foi validada a periodicidade da mensagem DT3 (E7), novamente através do requisito

(CYCLIC E7@-1) [275, 1]

e o histograma resultante é apresentado na Figura 6.12. Comparando-se com o caso sem comunicação aperiódica com o supervisório – ver Figura 6.8 – o comportamento é semelhante, inclusive quanto ao período mais freqüente do evento: 276ms. Mesmo os desvios maiores não aumentaram significativamente, passando a 283ms e 269ms (comparando-se com 282ms e 270ms do caso da Figura 6.8).

Figura 6.12 – Histograma de mensagem periódica, caso 1.2b

A validação do atraso entre duas passagens sucessivas da permissão aperiódica para um mesmo dispositivo através do requisito

(E0@-1 AFTER E0@-2) [0, 40]

é apresentada no histograma da figura Figura 6.13, que deve ser comparado com a Figura 6.9. O pico principal continua o mesmo, devido aos mesmos fatores mencionados (o tempo total para a rotação da permissão entre os três dispositivos presentes no barramento). Entretanto, embora existam outros picos menores, estes não estão isolados, ou seja, existe uma distribuição nos tempos verificados. Esta distribuição pode ser relacionada a existência de um número maior de mensagens aperiódicas interferindo no ciclo de tempo disponível pelo LAS para acomodar todas as mensagens que devem ser comunicadas. É interessante comparar também o atraso máximo verificado, que é em torno de 240ms, em torno de 30% maior que o do caso da Figura 6.9. Apesar de maior que no caso anterior, este atraso ainda é menor que o macrociclo. Isto é possível neste caso porque existe apenas uma mensagem periódica reservando tempo no macrociclo e porque a rotação da passagem aperiódica entre os dispositivos da rede é rápida, ocorrendo uma passagem logo após a outra.

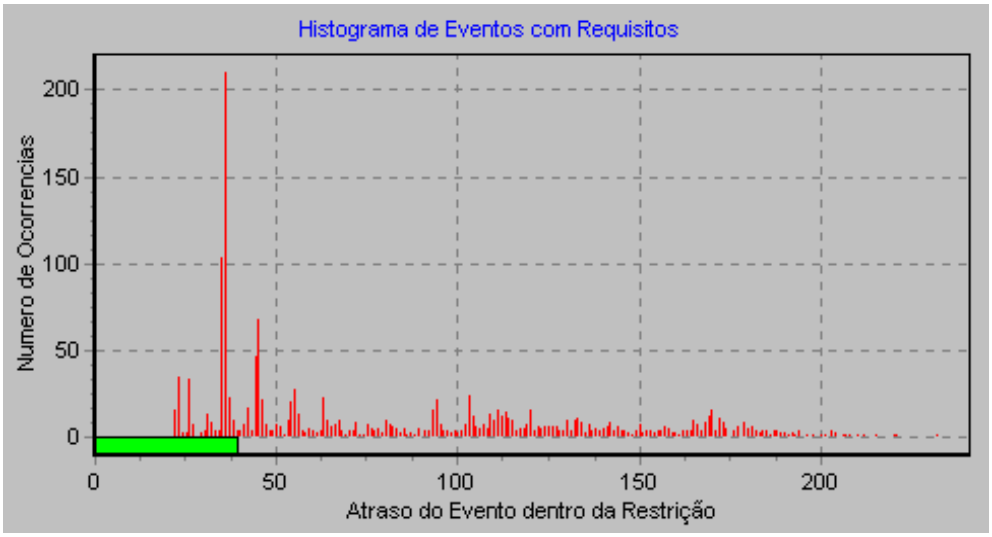

Figura 6.13 – Histograma de mensagem aperiódica, caso 1.2b

6.2 Estudo de Caso 2 – Controle de uma planta experimental

Uma aplicação para controle de uma planta experimental é o segundo estudo de caso para a ferramenta. A aplicação é proposta para o controle de uma planta conforme Figura 6.14. A aplicação possui laços de controle para o nível de cada um dos tanques, para a composição do tanque 3 (proporção entre insumos provenientes dos tanques 1 e 2), e ainda ajuste de ponto de operação para saída do tanque 3. Para tanto existem sensores de nível (1, 2 e 3) em cada tanque, sensores de fluxo (1 e 2) na saída dos tanques 1 e 2, e válvulas de controle (1 a 5) para a entrada e saída de cada tanque.

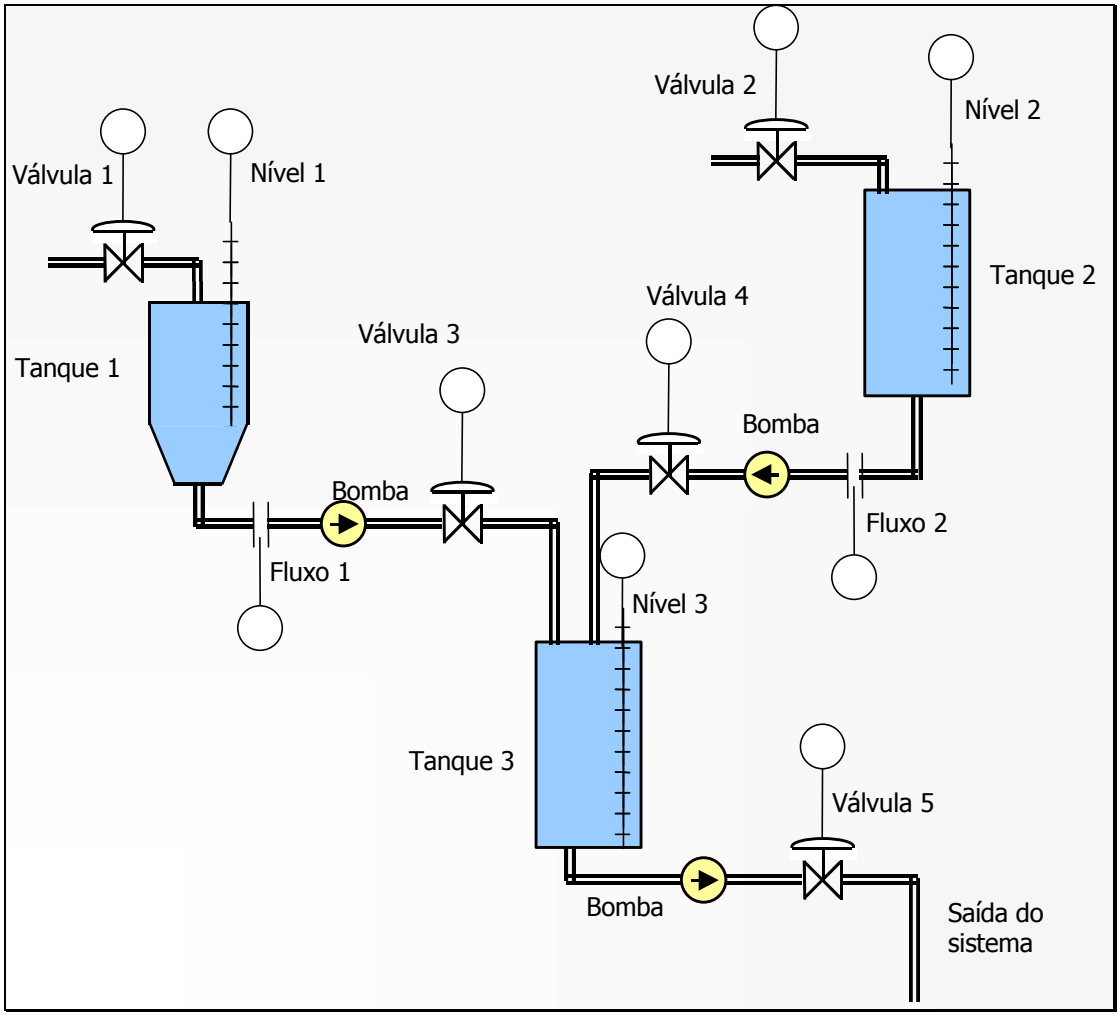

Figura 6.14 – Esquema da planta a ser controlada

A configuração resultante conta com 10 ligações entre blocos funcionais publicadas no barramento. O macrociclo proposto pela ferramenta de configuração é de 430ms. O tempo de ocupação de segundo plano foi definido, também pela ferramenta, como 100ms.

6.2.1 Caso sem comunicação de dados para supervisório

Analogamente a análise anterior, um primeiro passo abordou o comportamento temporal da aplicação rodando sem monitoração de parâmetros (dados para supervisão). A correspondência entre tipo de mensagens e nome de eventos, para identificação pela ferramenta, é a mesma utilizada para o estudo de caso anterior (ver Tabela 6.1). A distribuição percentual das mensagens, por tempo, é apresentada na Figura 6.15.

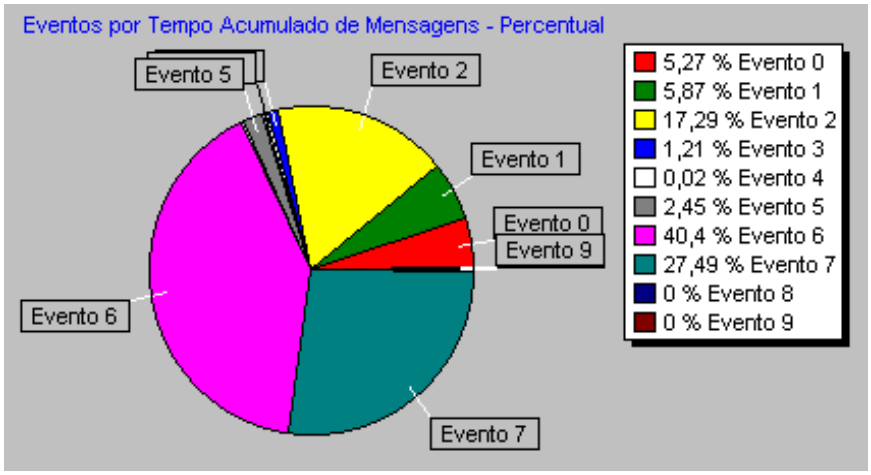

Figura 6.15 – Tempo percentual, caso 2.1

É possível observar na distribuição das mensagens por tempo a predominância dos eventos CD e DT3, referentes a publicação de mensagens periódicas, devido ao número significativo de ligações entre blocos funcionais em diferentes dispositivos, sendo publicadas no barramento (10 ligações). Novamente é interessante notar que a mensagem Invoca Dados (E6, CD) ocupa mais tempo no barramento do que a própria Publicação de Dados (E7, DT3). Conforme mencionado acima, a duração da mensagem é computada desde o término da última mensagem publicada, incluindo o período de silêncio na linha cuja duração pode ser utilizada pelo LAS para ajustar com exatidão o instante de ocorrência (término) da mensagem sendo publicada. Além das mensagens relacionadas a publicação periódica de dados, ocupam tempo significativo as mensagens de barramento ocioso.

6.2.2 Caso com comunicação de dados para supervisório

Em seguida, foi validada a situação em que dados de supervisão são configurados para comunicação. Com 15 parâmetros para supervisão, o resultado da distribuição pode ser visto na Figura 6.16. Um número maior de parâmetros leva a atrasos na comunicação de magnitude suficiente para disparar os alarmes de temporização do supervisório para praticamente todos os dados supervisionados. A distribuição é semelhante àquela do caso sem dados para supervisão visto na Figura 6.12; apenas o evento DT1 (E5) correspondente à publicação de dados aperiódicos aumentou em pequena proporção.

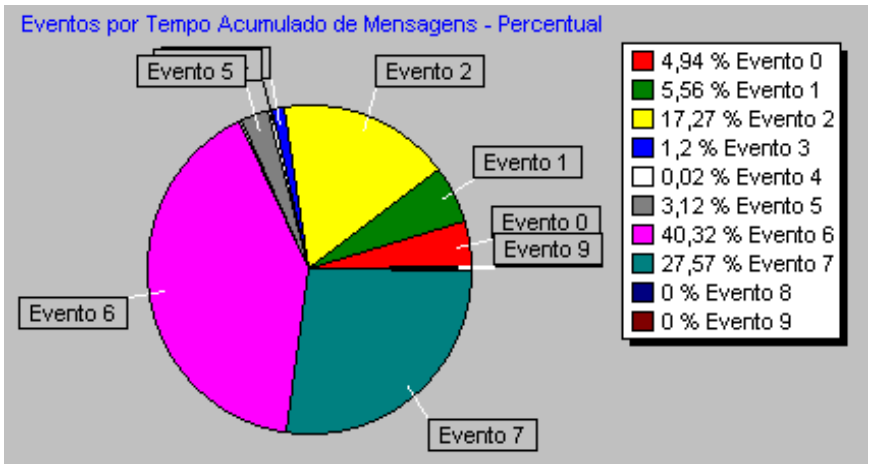

Figura 6.16 – Tempo percentual, caso 2.2

A Figura 6.17 contém o diagrama de Gantt para um intervalo do registro de mensagens desta situação. Nota-se a intensa ocupação do barramento pelas mensagens de publicação de dados periódicos (E6 e E7), e a pequena fração de tempo destinada à circulação de dados aperiódicos (entre os blocos de mensagens E6 e E7), que é função do tempo de ocupação de segundo plano definido (que é em torno de 23%). Ainda é interessante observar a presença de mensagens de barramento ocioso (E2). Estas últimas ocorrem em duas situações: entre um par CD/DT3 e outro, e ao fim do macrociclo, quando não há mais tempo disponível para circular a permissão aperiódica ou outra mensagem de gerenciamento da comunicação.

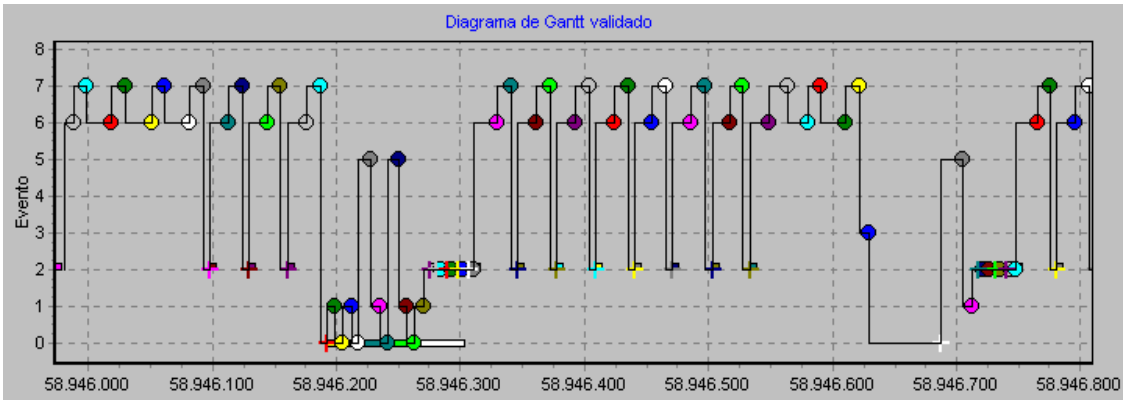

Figura 6.17 – Diagrama de Gantt, caso 2.2

O restrito tempo reservado para comunicação de dados aperiódicos e o número de dispositivos presentes no barramento pode levar a um tempo significativo para a rotação da passagem de permissão aperiódica entre todos os dispositivos. O histograma da Figura 6.18 refere-se ao evento passagem de permissão aperiódica para um endereço específico, definido como E0, validado para o requisito já apresentado

(E0@-1 AFTER E0@-2) [0, 40]

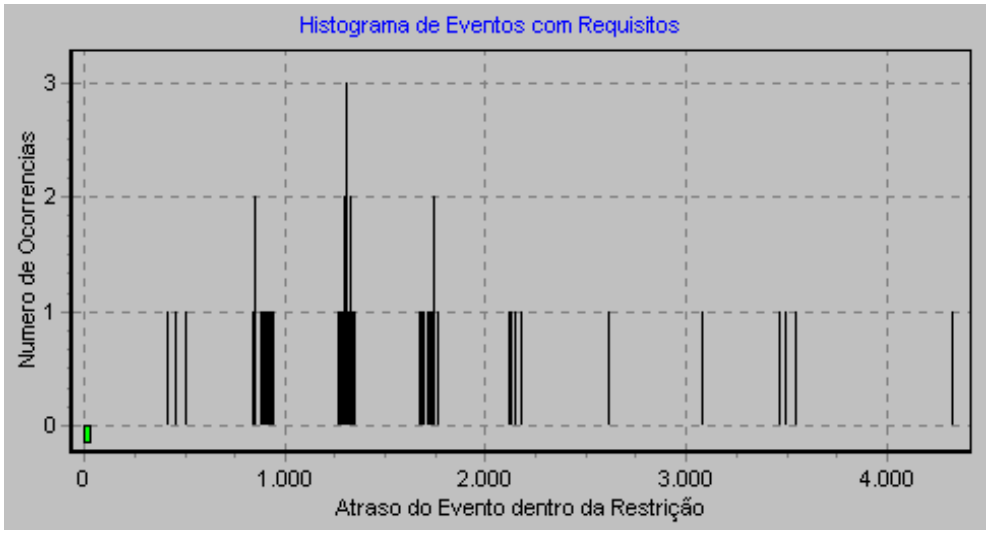

Figura 6.18 – Histograma de mensagem aperiódica, endereço específico, caso 2.2

O histograma confirma a suposição de tempos elevados entre duas passagens sucessivas de permissão ao mesmo endereço. Mesmo considerando que mensagens aperiódicas são reservadas a dados sem restrições temporais, um atraso desta magnitude pode ser considerado muito elevado, com o risco de comprometer a atualização de dados em supervisórios, ou o funcionamento satisfatório de uma interface homem-máquina.

A periodicidade do evento de invocação de dados periódicos para um determinado endereço também foi investigada para este caso. A seguir são apresentados os resultados da validação do requisito

(CYCLIC E6@-1) [430, 1]

Na Figura 6.19, E6 é o evento CD para um endereço específico que, dentro da seqüência de CD dentro do macrociclo, **não** ocupa a primeira posição (não é o primeiro CD da seqüência dentro do macrociclo). Como pode-se notar na Figura 6.17, os pares de mensagens CD2/DT3 ocorrem em seqüência dentro do macrociclo, e é importante fazer esta distinção, já que adiante será destacada a diferença entre o comportamento

das mensagens periódicas quando ocupam ou não o primeiro lugar na seqüência. Observa-se que a variação de periodicidade é semelhante ao de outros casos, já analisados – inclusive no fato de que o período efetivo não é exatamente igual ao macrociclo proposto pela ferramenta de configuração.

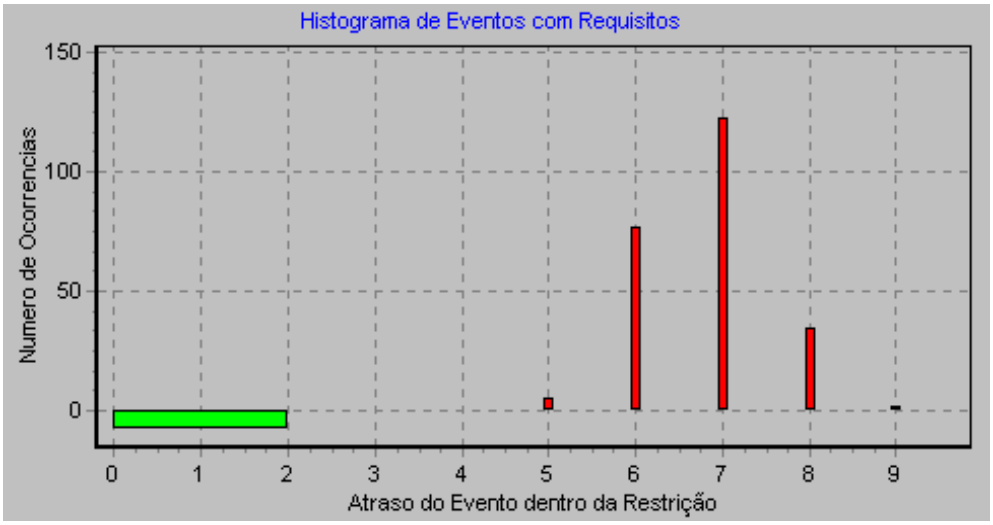

Figura 6.19 – Histograma de mensagem periódica, endereço específico, não ocupa 1° lugar na seqüência, caso 2.2

Já a Figura 6.20 mostra a mesma validação para um endereço específico que é o primeiro a ser publicado dentro do macrociclo. É possível observar que, embora o comportamento em linhas gerais seja o mesmo que aquele do evento na Figura 6.19, inclusive no que tange a período efetivamente verificado, a variação da periodicidade é maior.

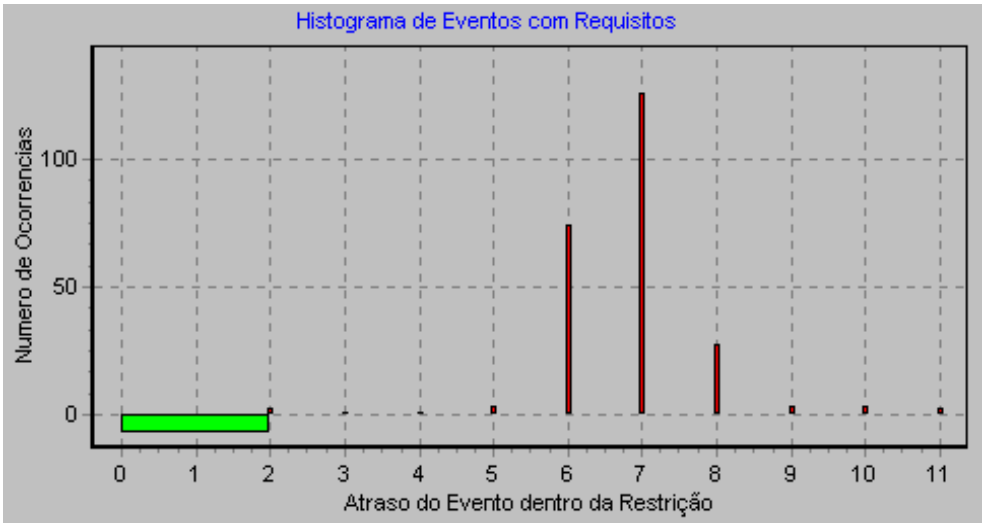

Figura 6.20 – Histograma de mensagem periódica, endereço específico, ocupa o 1° lugar na seqüência, caso 2.2

6.2.3 Caso com comunicação de dados para supervisório, com tempo de ocupação de segundo plano aumentado

Com o intuito de avaliar a influência do valor de tempo de ocupação de segundo plano, a mesma configuração de aplicação e dados de supervisão teve este valor alterado de 100ms para 250ms. A ferramenta reconfigurou para este valor de tempo de ocupação de segundo plano um macrociclo de 580ms. Desta maneira, o percentual relativo de tempo de ocupação de segundo plano passou a cerca de 43%.

A Figura 6.21 mostra o resultado desta alteração para o evento passagem de permissão aperiódica para um endereço específico. Tanto o atraso da maior parte das ocorrências como os maiores atrasos verificados são menores do que o caso anterior (23% de tempo de ocupação de segundo plano) – compare-se com a Figura 6.18. Como o escalonamento libera mais tempo para dados aperiódicos, a demora entre duas passagens sucessivas para o mesmo endereço diminui.

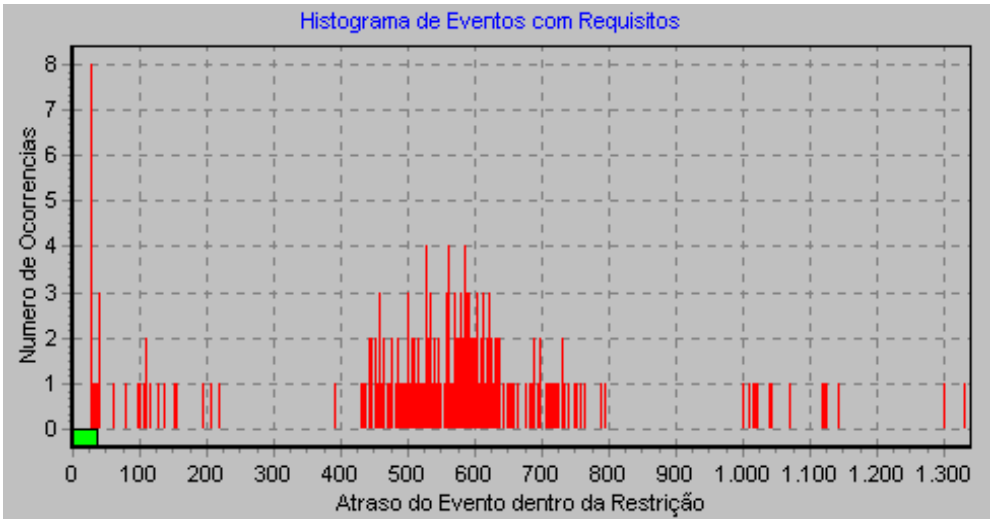

Figura 6.21 – Histograma de mensagem aperiódica, endereço específico, caso 2.3

A periodicidade dos eventos CD para este caso é apresentada nas figuras a seguir, através do requisito

(CYCLIC E6@-1) [580, 1]

O histograma na Figura 6.22 valida o evento CD para um endereço específico que não é o primeiro na seqüência de eventos CD dentro do macrociclo. A distribuição de periodicidade não difere significativamente daquela observada no caso anterior (ver Figura 6.19). Já a Figura 6.23, que apresenta a validação para o evento CD que é o primeiro na seqüência dentro do macrociclo, mostra uma variação muito maior, com

desvios de até ±12ms (cerca de 2% do período). Novamente, o período efetivamente observado, para ambos os casos, apresenta uma diferença em relação àquele proposto pela ferramenta de configuração.

Figura 6.22 – Histograma de mensagem periódica, endereço específico, não ocupa 1° lugar na seqüência, caso 2.3

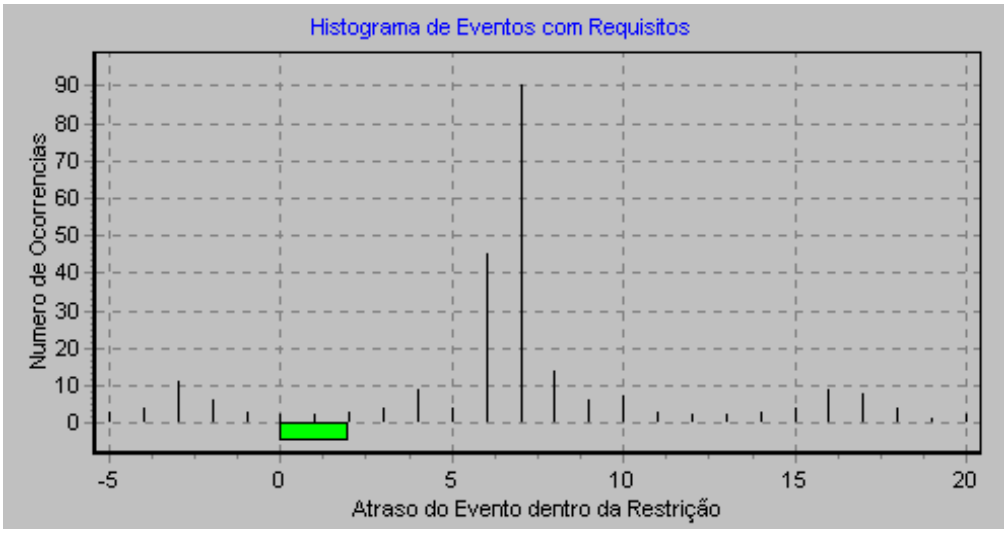

Figura 6.23 - Histograma de mensagem periódica, endereço específico, ocupa 1° lugar na seqüência, caso 2.3

Por fim, a Figura 6.24 mostra a distribuição por tempo das mensagens no registro feito do caso sob estudo. Deve-se destacar a maior participação do evento E5, publicação de dados aperiódicos, em relação ao caso anterior – ver Figura 6.16. Também ocorreu diminuição na participação percentual da publicação de dados periódicos, E6 e E7, e de mensagens de barramento ocioso, E2.

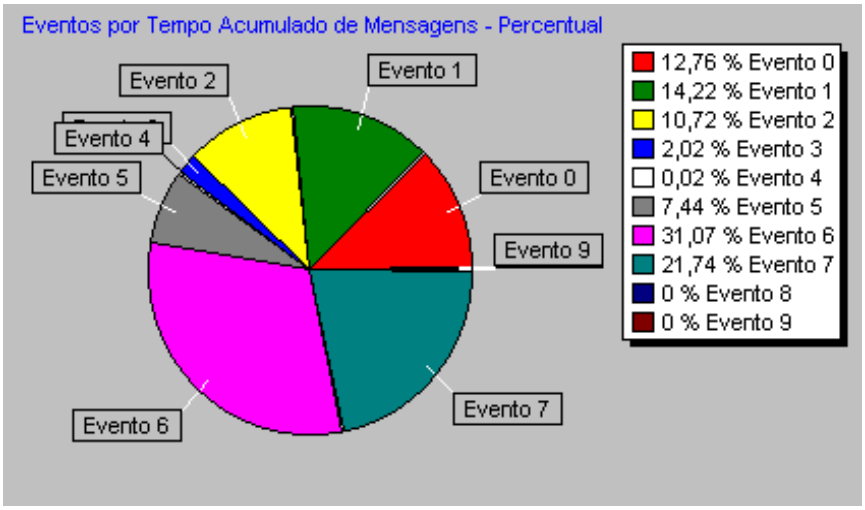

Figura 6.24 – Tempo percentual, caso 2.3

É interessante ressaltar que estes estudos de caso foram realizados a partir de aplicações rodando em uma planta experimental, existente no Laboratório de Automação Industrial deste Departamento, contando com diversos instrumentos e interação real com tanques, válvulas e tubulações de ligação.

6.3 Estudo de caso 3 – Aplicação Smar

Este estudo de caso foi efetuado a partir de registros de comunicação cedidos pelo fabricante de tecnologia e dispositivos *Foundation Fieldbus* Smar Equipamentos Industriais. Estes registros refletem o comportamento temporal de uma rede composta por um grande número de dispositivos, rodando uma aplicação com uma quantidade elevada tanto de ligações entre blocos funcionais, como de dados de supervisão. Tal rede foi configurada exatamente para investigar situações de saturação na comunicação. O estudo foi feito com base em um arquivo de registro de um caso em que os dados de supervisão foram agrupados, sendo colocados diversos dados dentro da mesma mensagem. Segundo o fabricante, trata-se de uma estratégia especial para diminuir o número de mensagens na rede.

A análise deste caso foi iniciada com a participação percentual das mensagens no tempo total de comunicação (Figura 6.25). Observa-se a grande participação da transmissão de dados aperiódicos (Evento 5) no tempo de comunicação, ainda maior que a dos casos estudados anteriormente. Também é notável a pequena quantidade de tempo de barramento ocioso (Evento 2), em torno de 5%, a metade do melhor caso

observado nos estudos anteriores. Frente à quantidade de mensagens aperiódicas, a percentagem de dados periódicos é menor que a dos outros casos estudados.

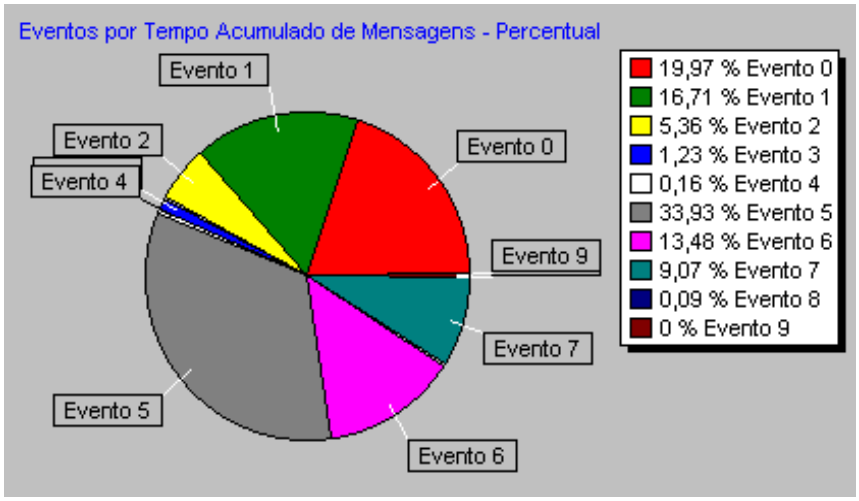

Figura 6.25 – Tempo percentual, caso 3

O comportamento das mensagens aperiódicas é apresentado graficamente na Figura 6.26, através do atraso entre duas passagens de permissão. Nota-se mais uma vez um pico em um valor pequeno de atraso, relativo aos intervalos em que a permissão é passada entre os dispositivos sem outras mensagens intervenientes. Quando outras mensagens ocorrem entre uma passagem de permissão e outra, o atraso torna-se maior, gerando a distribuição observada na figura.

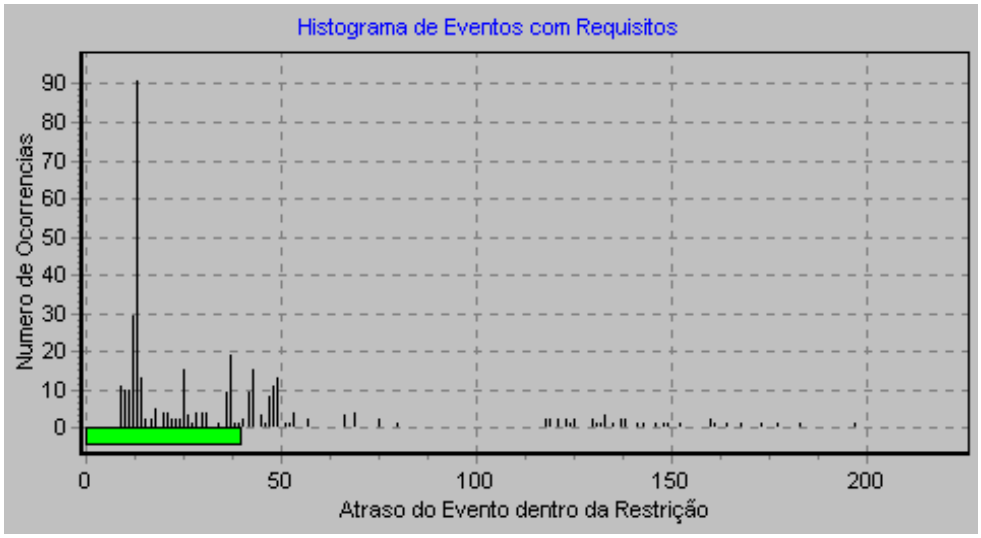

Figura 6.26 – Histograma de mensagens aperiódicas, caso 3

A avaliação do intervalo entre duas passagens de permissão ao mesmo endereço está na Figura 6.27. Embora não exista nenhuma ocorrência de atraso menor que 200ms,

ainda assim os valores estão mais agrupados em comparação aos casos estudados anteriormente, e nenhuma ocorrência observada supera 700ms. Possivelmente esta situação deve-se à estratégia de agrupamento de informações em mensagens, otimizando a utilização do barramento.

Figura 6.27 – Histograma de mensagens aperiódicas, endereço específico, caso 3

A Figura 6.28 contém o diagrama de Gantt validado para um intervalo de validação um pouco maior que o macrociclo configurado. Pode-se observar duas seqüências de mensagens periódicas (Eventos 6 e 7), e entre estas uma seqüência de mensagens aperiódicas (Eventos 0, 1 e 5), mostrando de outra forma o comportamento da Figura 6.26, com várias passagens de permissão uma após a outra, e intervalos em que a passagem de permissão deve esperar outros eventos antes de ocorrer.

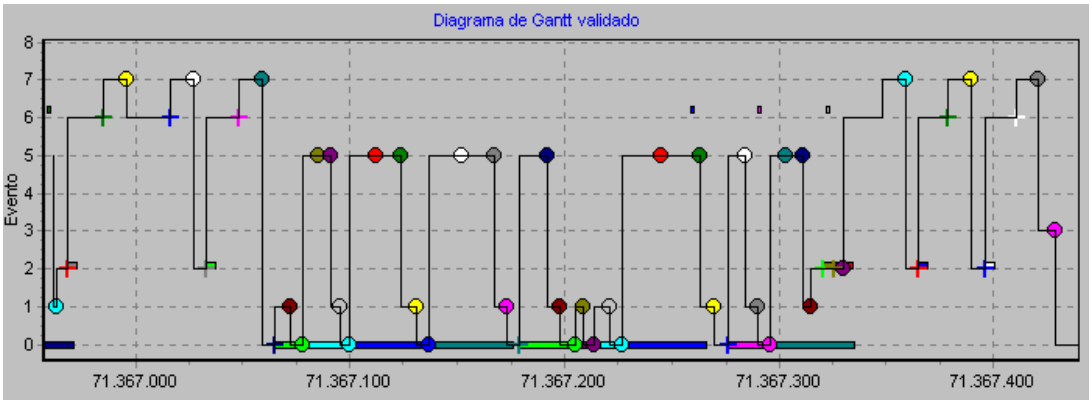

Figura 6.28 – Diagrama de Gantt, caso 3

Por fim, é interessante observar o comportamento de mensagens periódicas, conforme apresentado na Figura 6.29 e na Figura 6.30. A primeira mostra a ciclicidade da mensagem que é a primeira na seqüência de periódicas do macrociclo, e a segunda é

relativa a segunda (conforme já discutido em outro estudo acima). Para este caso repete-se a situação já observada antes: a primeira mensagem na seqüência de mensagens periódicas do macrociclo tem uma ciclicidade mais irregular do que as subseqüentes.

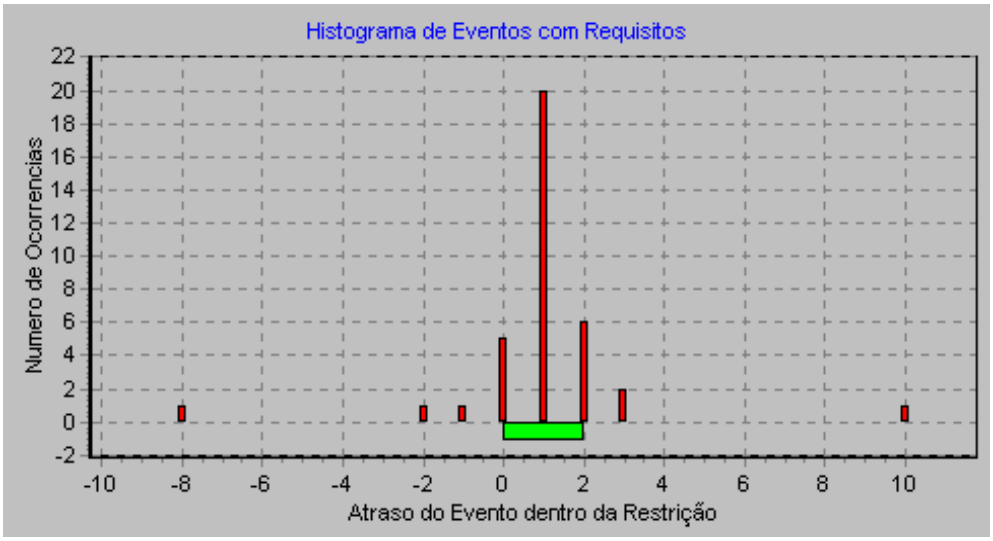

Figura 6.29 – Histograma de mensagem periódica, endereço específico, ocupa 1° lugar na seqüência, caso 3

Figura 6.30 – Histograma de mensagem periódica, endereço específico, não ocupa 1° lugar na seqüência, caso 3

6.4 Conclusões sobre os estudos de caso

A análise dos estudos de caso apresentados revela aspectos interessantes do comportamento temporal da comunicação no *Foundation Fieldbus*. O algoritmo utilizado pelo LAS para escalonar as mensagens no barramento e a forma como a ferramenta de configuração cria a tabela de escalonamento para o LAS combinam-se

para gerar as características da comunicação, em função de variáveis tais como número de ligações entre blocos funcionais e número de parâmetros da aplicação a serem supervisionados.

Em primeiro lugar, é possível observar que mesmo para aplicações simples, com pouca ocupação de barramento, existe uma variação de periodicidade, embora pequena, para a publicação de seus dados periódicos. No caso da aplicação mais complexa, observou-se uma variação considerável quando a mensagem em questão é a primeira mensagem do macrociclo.

Outras observações referem-se à publicação de dados aperiódicos. Em função da estratégia de passagem de permissão aperiódica para os dispositivos, situações de pouca ocupação do barramento levam a um comportamento “quase periódico”. Já situações com ocupação maior têm como característica interessante o tempo de retorno da permissão ao dispositivo. O tempo reservado pelo configurador, ao elaborar o escalonamento, para a ocupação de segundo plano, pode tornar-se muito pequeno em face da quantidade de dados aperiódicos a serem publicados. Nestes casos, a demora no retorno da permissão ao dispositivo pode elevar-se a valores inadequados para certas aplicações.

Também a proporção de mensagens de barramento ocioso é bastante significativa em todos os casos. Mesmo situações muito carregadas e com tempo de retorno de permissão aperiódica elevado contam uma proporção de mensagens de barramento ocioso semelhante a situações menos carregadas. Uma reelaboração da estratégia de escalonamento, em conjunto com alterações nos parâmetros temporais da comunicação utilizados pelo LAS, poderia auxiliar no melhor aproveitamento desta parcela de tempo da comunicação.

A ferramenta proposta e implementada auxilia efetivamente na visualização e compreensão do comportamento temporal dos sistemas investigados. A utilização da ferramenta permitiu chegar-se a conclusões as quais o simples exame do registro de eventos da comunicação não permitiria, ou tornaria muito difícil. Entretanto, embora as necessidades de validação e visualização consideradas mais importantes tenham sido implementadas, o aperfeiçoamento destas capacidades seria interessante para possibilitar análises com um maior grau de detalhe. A utilização futura da ferramenta, tanto com o protocolo *Foundation Fieldbus* como com outros protocolos de comunicação, poderá definir quais os aperfeiçoamentos mais necessários e úteis.

7. Conclusões

A tecnologia de barramentos de campo conquistou grande importância dentro da área de automação industrial, devido aos inúmeros benefícios oferecidos. Por tratar-se de sistemas tempo-real distribuídos, aplicações utilizando tal tecnologia exigem um suporte adequado tanto para seu projeto como para sua validação. Neste tipo de sistema, a comunicação desempenha um importante papel na garantia de requisitos temporais. O presente trabalho propôs uma ferramenta para monitoração, validação e visualização da comunicação no barramento.

O trabalho desenvolvido ao longo da elaboração desta dissertação mostrou que as premissas propostas para a elaboração eram fundamentalmente corretas. Tanto a necessidade de validação para o comportamento de sistemas tempo-real tais como barramentos de campo, quanto a carência de uma ferramenta para efetuar esta validação dentro do escopo apresentado, mostraram-se úteis balizas para orientar este estudo.

A ferramenta desenvolvida apresenta as funcionalidades básicas necessárias para efetuar a validação proposta, atingindo assim um dos objetivos deste trabalho. O validador desenvolvido tem por entrada requisitos temporais formalizados, e um histórico registrado de ocorrências de eventos do sistema, gerando a partir daí informação a respeito da comparação deste com aqueles: concordância e variações de temporização em tempo de execução. A verificação em princípio ocorre em tempo posterior à execução. São propostas ainda várias possibilidades de visualização dos resultados obtidos. É importante ressaltar que, apesar da presente implementação estar vinculada ao protocolo *Foundation* Fieldbus, a proposta é genérica, podendo ser adaptada a outros protocolos.

O resultado do trabalho, tanto em relação à própria ferramenta, como em relação às análises efetuadas, mostrou-se extremamente válido no âmbito da validação de sistemas de automação baseados em barramentos de campo. A presente abordagem é realmente efetiva para revelar aspectos importantes a respeito do comportamento temporal da comunicação. A aplicação da ferramenta a estudos de caso revelou informações interessantes sobre a comunicação no barramento de campo investigado, não acessíveis ao exame simples do registro de mensagens. As informações obtidas, tais como desvios de periodicidade e atrasos máximos verificados para mensagens, proporcionam uma realimentação importante para engenheiros de automação em *fieldbus*, tanto do ponto de vista da implementação de tecnologia quanto para o desenvolvimento de aplicações seguras e eficientes.

Cabe destacar que, apesar disso, em vários aspectos esta ferramenta merece aperfeiçoamentos, para ampliar suas capacidades e utilizabilidade. A utilização da ferramenta revelou alguns destes aperfeiçoamentos possíveis. Por exemplo, aumentar as capacidades de validação e representação da ferramenta, para torná-la mais poderosa em relação a requisitos temporais. Também é importante a extensão do tratamento de mensagens para outros protocolos - por exemplo, métodos periódicos em classes ativas (AOC++) (ver [PER94]) e mensagens Profibus. Uma extensão possível foi proposta e implementada por [BEC99] para exame do comportamento temporal de um protocolo experimental baseado em CSMA/CD [SOA95].

Além de aperfeiçoamentos da própria ferramenta, possíveis trabalhos futuros incluem a sua aplicação a situações diversas em que a validação temporal de uma solução seja importante. Uma tal situação é, por exemplo, a avaliação comparativa de estratégias de escalonamento para o LAS no *Foundation Fieldbus*, já que a possibilidade de se propor uma estratégia mais elaborada é aberta pela observação de algumas características no comportamento atual da comunicação no barramento investigado.

Bibliografia

[BAK91] BAKER, T. Stack-Based Scheduling of Realtime Processes. **J. Real-Time Systems**, v. 3, n. 1, pp. 67-99, Mar. 1991.

[BEC99] PEREIRA, C.; BECKER, L.; GOTZ, M.; WILD, R.; HÜSEMANN, R. Tool Support for Evaluating Temporal Characteristics of Industrial Protocols. Trabalho submetido ao IFAC WCFS 2000.

[BRA89] BRANTLEY, W.; MCAULIFFE, K.; NGO, T. RP3 Performance Monitoring Hardware. In: SIMMONS, M.; KOSKELA, R.; BUCHER, I. editores. **Instrumentation for Future Parallel Computing Systems.** New York: ACM Press, 1989.

[BUR90] BURNS, A.; WELLINGS, A. **Real-time Systems and Their Programming Languages**. Reading, USA: Addison-Wesley, 1990.

[CHO91] CHODROW, S.; JAHANIAN, F.; DONNER, M. Run-Time Monitoring of Real-Time Systems. In: **Proc. Real-time Systems Symp**. Los Alamitos, USA: IEEE CS Press, 1991, pp 74-83.

[DAS85] DASARATHY, B. Timing Constraints of Real-Time Systems: Constructs for Expressing Them, Methods of Validating Them. **IEEE Transactions on Software Engineering**, v. SE-11, n. 1, pp.80-86, Jan. 1985.

[DOD92] DODD, P.; RAVISHANKAR, C. Monitoring and Debugging Distributed Real-Time Programs. **Software - Practice and Experience**, v. 22, n. 10, pp. 863-877, Oct. 1992.

[FIE98] FIELDBUS FOUNDATION. **Foundation Fieldbus Technical Overview**: FD-043 versão 2.0, 1996 (rev. 1998). Disponível em http://www.fieldbus.org/pdf/fd-043s.pdf (mar. 1999)

[FIE99] FIELDBUS FOUNDATION. **Foundation Fieldbus Specification:** Versão 1.4. Fieldbus Foundation, 1999.

[GIR98] GIRARDIN, L.; BRODBECK, D. A Visual Approach for Monitoring Logs. In: SYSTEM ADMINISTRATION CONFERENCE, 12., 1998, Boston. **Proceedings...** pp. 299-308. Disponível em www.ubs.ch/e/index/about/ubilab/print-versions/pdf/gir98.pdf (mar. 99)

[HAB90] HABAN, D.; WYBRANIETZ, D. A Hybrid Monitor for Behavior and Performance Analysis of Distributed Systems. **IEEE Transactions on Software Engineering**, v. 16, n. 2, pp. 197-211, Feb. 1990.

[JAH86] JAHANIAN, F.; MOK, A. Safety Analysis of Timing Properties in Real-Time Systems. **IEEE Transactions on Software Engineering**, v. SE-12, n. 9, pp. 890-904, Sep. 1986.

[JAH95] JAHANIAN, Farnam. Run-Time Monitoring of Real-Time Systems. In: SON, Sang H. (editor). **Advances in Real-Time Systems**. Englewood Cliffs, USA: Prentice Hall, 1995.

[JOY87] JOYCE, J.; LOMOW, G.; SLIND, K.; UNGER, B. Monitoring Distributed Systems**. ACM Transactions on Computer Systems**, v. 5, n. 2, pp 121-150, 1987.

[LIU89] LIU, A.; PARTHASARATHI, R. Hardware Monitoring of a Multiprocessor System. **IEEE Micro**, v. 9, n. 5, pp. 44-51, Oct. 1989.

[MCD89] MCDOWELL, C.; HELMBOLD, D. Debugging Concurrent Programs. **ACM Computing Surveys**, v. 21, n. 4, pp. 593-622, Dec. 1989.

[MIL86] MILLER, B.; MACRANDER, C.; SECHREST, S. A Distributed Programs Monitor for Berkeley UNIX. **Software - Practice and Experience**, v. 16, n. 2, pp. 183-200, 1986.

[MIN90] MINK, A.; CARPENTER, R.; NACHT, G.; ROBERTS, J. Multiprocessor Performance-Measurement Instrumentation. **Computer**, v. 23, n. 9, pp. 63-75, Sept. 1990.

[PER94] PEREIRA, C. Real Time Active Objects in C++/Real-Time UNIX. In:ACM SIGPLAN WORKSHOP ON LANGUAGES, COMPILER, AND TOOL SUPPORT FOR REAL-TIME SYSTEMS, Orlando, EUA. **Proceedings...**, 1994.

[PER95] PEREIRA, C. Temporal Reasoning on Object-Oriented Analysis Methods. In: EUROMICRO WORKSHOP ON REAL-TIME PROGRAMMING, 6., Vaesteraas, Sweden. **Proceedings...**, 1994, pp. 104-109

[PLA84] PLATTNER, B. Real-Time Execution Monitoring. **IEEE Transactions on Software Engineering**, v. SE-10, n. 6, pp. 756-764, 1984.

[RAJ92] RAJU, S.; RAJKUMAR, R.; JAHANIAN, F. Timing Constraints Monitoring in Distributed Real-time Systems. In: REAL-TIME SYSTEMS SYMPOSIUM. **Proceedings...** Los Alamitos, USA: IEEE CS Press, 1992, pp. 57-67.

[RYU98] RYU, M.; HONG, S. End-to-End design of distributed real-time systems. **Control Engineering Practice**, n. 6, pp. 93-102, 1998.

[SHA89] SHA, L.; RAJKUMAR, R.; LOHOCZKY, J.; RAMAMRITHAM, K. Mode Change Protocols for Priority-Driven Preemptive Scheduling. **J. Real-Time Systems**, v. 1, pp. 243-264, 1989.

[SIL95] SILVA NETO, E.; HASLER, H.; SOUZA, L.; FRANCO, L.; ARAÚJO, O.; ROCHA, W. Comunicação em Chão de Fábrica: Solução Fieldbus. In: CONGRESSO BRASILEIRO DA ISA, 3/CONGRESSO INTERNACIONAL DA ISA, 1. **Anais...** 1995. pp. 337-345.

[SOA95] SOARES, L.; LEMOS, G.; COLCHER, S. **Redes de computadores**: das LANs, MANs e WANs às redes ATM. Rio de Janeiro: Campus, 1995.

[SON95] SON, Sang H. Preface. In: SON, Sang H. (editor). **Advances in Real-Time Systems**. Englewood Cliffs, USA: Prentice Hall, 1995.

[STA88] STANKOVIC, J. Misconceptions About Real-Time Computing: A Serious Problem for Next-Generation Systems. **Computer**, New York. p. 10-19, Oct. 1988.

[TIM98] TIMESYS CORPORATION. **TimeWiz User's Guide**. Versão 1.0. TimeSys Corporation, 1998.

[TOK88] TOKUDA, H.; KOTERA, M.; MERCER, C. A Real-Time Monitor for a Distributed Real-Time Operating System. In: ACM WORKSHOP PARALLEL AND DISTRIBUTED DEBUGGING. **Proceedings...** New York, ACM Press, 1988, pp. 68-77.

[TSA90] TSAI, J.; FANG, K.; CHEN, H. A Noninvasive Architecture to Monitor Real-Time Distributed Systems. **Computer**, v. 23, n. 3, pp. 11-23, Mar. 1990.

[TSA96] TSAI, J.; BI, Y; YANG, S.; SMITH, R. **Distributed Real-Time Systems**: Monitoring, Visualization, Debugging, and Analysis. Wiley-Interscience, 1996.