JOAQUIM BRIGHENTI NETO
MARIO ROBERTO BASTOS

MODELAGEM DE UM SERVIDOR DE COMUNICAÇÃO PARA SUBESTAÇÃO

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para conclusão do Curso de Especialização em Tecnologia da Informação

São Paulo
2002

JOAQUIM BRIGHENTI NETO
MARIO ROBERTO BASTOS

MODELAGEM DE UM SERVIDOR DE COMUNICAÇÃO PARA SUBESTAÇÃO

Monografia apresentada à Escola Politécnica da Universidade de São Paulo para conclusão do Curso de Especialização em Tecnologia da Informação

Área de Concentração
Computação e Sistemas Digitais

Orientador:
Prof. Doutor
Jorge Luis Risco Becerra

São Paulo
2002

AGRADECIMENTOS

Ao nosso orientador Prof. Dr. Jorge Luis Risco Becerra pela competência e entusiasmo com que nos guiou durante a elaboração deste trabalho.

Ao Prof. Dr. José Sidnei Martini Colombo pela oportunidade ímpar de aprendizado que nos propiciou.

Aos nossos Gerentes pela confiança e compreensão que nos permitiram levar a bom termo este trabalho.

RESUMO

Este trabalho apresenta o modelo orientado a objetos de um Servidor de Comunicação para subestação projetado para integrar uma Unidade Terminal Remota legada com um equipamento moderno de seqüência de eventos. O enfoque adotado foi o de integração através do encapsulamento da interface. O texto preocupa-se basicamente com um problema concreto da Companhia de Transmissão de Energia Elétrica Paulista, mas o conceito é suficientemente geral para ser aplicado a outros casos.

ABSTRACT

This report presents the object oriented model of a Substation Communication Server designed to integrate a legacy Remote Terminal Unit with a modern sequence of events equipment. The approach used was to integrate through interface encapsulation. The text deals with an actual problem of Companhia de Transmissão de Energia Elétrica Paulista, but the concept is general enough to be applied to other cases.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ASCII	American Standard Code for Information Interchange
ASDU	Application Service Data Unit
CAG	Controle Automático de Geração
CESP	Companhia Energética de São Paulo
CNOS	Centro Nacional de Operação do Sistema
CORBA	Common Object Request Broker Architecture
COS	Centro de Operação do Sistema
CRC	Cyclic Redundancy Check
CRO	Centro Regional de Operação
CROB	Centro Regional de Operação de Bauru
CROC	Centro Regional de Operação de Cabreúva
CRO-SP	Centro Regional de Operação de São Paulo
CTEEP	Companhia de Transmissão de Energia Elétrica Paulista
DM	Disconnected Mode
DNP	Distributed Network Protocol
DTE	Data Terminal Equipment
EIA	Eletronics Industries Alliance
EMS	Energy Management System
FIFO	First in First Out
GPS	Global Positioning System
HDLC	High-Level Data Link Control
HTML	HyperText Markup Language
ICCP	Inter-Control Center Communications Protocol
IEC	International Eletrotechnical Commission
IEEE	Institute of Eletrical and Eletronics Engineers
IIOP	Internet Inter-ORB Protocol
IP	Internet Protocol
ISO	International Organization for Standardization
ITU-T	International Telecommunication Union Telecommunication Standardization Sector
LAN	Local Area Netowork
MMS	Manufacturing Message Specification
OMG	Objetct Management Group
ONS	Operador Nacional do Sistema
OSI	Open Systems Interconnection
SCADA	Supervisory Control and Data Acquisition
SE	Sub Estação
SNRM	Set Normal Response Mode
SRA	Sinalizador Registrador de Alarmes
SSC	Sistema de Supervisão e Controle
TCM	Telecomando Monoestável
TCP	Transmission Control Protocol
TMA	Telemedida Analógica
TMD	Telemedida Digital
TMT	Telemedida de Tap
TSI	TeleSinal com Indicação de Mudança Momentânea

TSS	TeleSinal Simples
UA	Unnumbered Acknowledge
UCA	Utility Communications Architecture
UML	Unified Modeling Language
URL	Uniform Resource Locator
UTR	Unidade Terminal Remota

1. INTRODUÇÃO

Em meados do ano 2001, a Diretoria da Companhia de Transmissão de Energia Elétrica Paulista (CTEEP) autorizou a aquisição de 64 Unidades Terminais Remotas para uso na Rede Básica (tensões acima de 230Kv), a fim de adequar a supervisão desta rede às exigências do Operador Nacional do Sistema (ONS).

O processo de aquisição não contemplou 33 subestações da Rede Conexão (tensões abaixo de 230Kv), em virtude de indefinições regulatórias. Estas subestações utilizam Unidades Terminais Remotas fabricadas na década de 1980 pela Microlab. Embora robustas e com excelentes índices de disponibilidade, possuem duas deficiências funcionais principais: comunicam-se com um único Sistema de Supervisão e não fazem seqüênciamento de eventos.

1.1 OBJETIVO

O objetivo deste trabalho é fornecer uma solução técnica para a modernização da supervisão das subestações da Rede Conexão da CTEEP, disponibilizando para estas instalações as funções de seqüênciamento de eventos e comunicação com mais de um Centro de Operação, sem que haja necessidade de substituir-se as atuais Unidades Terminais Remotas.

Será apresentado o modelo de análise orientada a objetos de um Servidor de Comunicação para Subestação capaz de integrar em uma mesma Base de Dados informações provenientes de Unidades Terminais Remotas Microlab ML 7810, equipamento da década de 1980 e do Sinalizador Registrador de Alarmes, equipamento moderno com função de seqüênciamento de eventos desenvolvido pela CTEEP. O enfoque de integração será o de encapsular o legado através da interface.

1.2 ORGANIZAÇÃO

Este trabalho está estruturado conforme segue.

O capítulo 2 apresenta a fundamentação teórica englobando os conceitos básicos de tecnologia orientada a objetos e de protocolos de comunicação.

No capítulo 3 é feita a caracterização do domínio do problema e são apresentados os requisitos e a modelagem do sistema.

No capítulo 4 o trabalho é encerrado com a apresentação das conclusões e recomendações de trabalhos futuros que poderão ser realizados a partir desta pesquisa.

O capítulo 5 apresenta as referências bibliográficas utilizadas.

1.3 METODOLOGIA

O desenvolvimento desta monografia foi feito em três etapas logicamente encadeadas: definição do assunto, anteprojeto e projeto final.

Na definição do assunto procurou-se escolher um tema que correspondesse a área de interesse dos autores e que pudesse resolver um problema relevante para a empresa. Uma vez escolhido o tema, procurou-se delimitar a extensão do trabalho de modo que fosse possível de ser executado no prazo previamente definido.

Passou-se então a fase de anteprojeto com a elaboração de um sumário tentativo que foi apresentado ao orientador, juntamente com o primeiro capítulo da monografia.

Com as observações e a aprovação do orientador, deu-se inicio a elaboração do projeto final. Foram levantadas as fontes bibliográficas pertinentes, e definidos os requisitos, métodos e processos a serem utilizados na modelagem do sistema.

1.4 JUSTIFICATIVA

A maior parte das 33 subestações da Rede Conexão da CTEEP são total ou parcialmente desassistidas. As remotas destas instalações, por não possuírem seqüênciamento de eventos, não supervisionam as proteções. Quando da ocorrência de perturbações ou desligamentos, há necessidade de deslocar-se

operadores para diagnosticar a ocorrência. O processo todo é lento e provoca atrasos no restabelecimento do fornecimento de energia, prejudicando os consumidores.

Duas possíveis soluções para estes problemas foram estudadas: modernização das Unidades Terminais Remotas e adição de equipamentos que complementassem as funções das remotas Microlab de forma a eliminar suas deficiências funcionais. Pelas razões expostas no início deste capítulo, a troca das Unidades Terminais Remotas não foi considerada.

A primeira alternativa, modernização das Unidades Terminais Remotas, exigiria a troca das Unidades Centrais de Processamento baseadas em processador Intel 8080, tendo em vista que as limitações deste processador inviabilizam a implementação da função de seqüência de eventos. Em conseqüência, todo o software da remota teria que ser refeito. Haveria também problemas para a ampliação da capacidade das remotas, em função da falta de espaço nos gabinetes. Estes problemas tornaram esta alternativa pouco atraente.

A segunda alternativa, adição de equipamentos que suprissem as deficiências existentes, acabou mostrando-se mais interessante, em função da disponibilidade de um equipamento com capacidade de seqüênciamento de eventos desenvolvido pela CTEEP.

Este equipamento, Sinalizador Registrador de Alarmes (SRA) foi desenvolvido pela equipe do Laboratório de Eletrônica de Bauru. Baseado em tecnologia de microcontroladores da família 8051, tinha como objetivo inicial atender às necessidades urgentes de algumas instalações, cujos anunciadores não tinham mais condições de continuar operando.

Suas características de projeto permitem que seja operado tanto localmente através de um painel frontal, como remotamente através de porta serial de comunicação. Possui resolução e capacidade de discriminação de eventos de 1 milisegundo e pode ter seu relógio interno sincronizado via GPS.

Surgiu então a idéia de desenvolver-se um Servidor de Comunicação para Subestações, capaz de integrar informações do Sinalizador Registrador de Alarmes e da remota Microlab.

Nesta arquitetura, a supervisão dos disjuntores e das proteções, por necessitarem de seqüência de eventos seria feita pelo SRA, com a remota Microlab continuando a supervisionar e controlar os demais equipamentos. O Servidor deveria suportar comunicação com mais de um Centro de Operação e as informações adquiridas seriam enviadas como se tivessem sido adquiridas por um único equipamento.

Procedeu-se então a uma avaliação de viabilidade técnica e econômica. Verificou-se que a solução proposta atende a todas as necessidades de supervisão e controle da Rede Conexão e que pode ser implementada a um custo não superior a 20% do custo de uma Unidade Terminal Remota nova. Projetada para 33 subestações e considerando-se o custo atual de uma remota nova, pode-se estimar que a solução proposta permitirá uma economia de US$924.000,00.

Deve-se ressaltar que a substituição das 64 Unidades Terminais Remotas da Rede Básica irá disponibilizar grande quantidade de peças de reserva, tornando possível manter em operação as remotas Microlab por um período estimado em pelo menos 10 anos.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 A NECESSIDADE DO USO DA ENGENHARIA DE SOFTWARE

Em seus primórdios, o processo de desenvolvimento de software era muito mais uma arte do que uma ciência. Donald E. Knuth em sua a monumental obra de Ciência da Computação enfatizou este aspecto ao denomina-la The Art of Computer Programming. Enquanto esteve restrito às universidades e laboratórios este enfoque não causou grandes problemas.

Todavia, conforme o uso do software se generalizava e penetrava de forma cada mais profunda na vida das empresas e das pessoas, verificou-se que os enormes volumes de recursos gastos em seu desenvolvimento raramente produziam os resultados esperados. E quando isto ocorria era mais fruto do talento das pessoas envolvidas do que da existência de um sólido processo de desenvolvimento. Estouros de orçamento e de prazos eram (ou continuam sendo) a regra.

Tornou-se claro que o desenvolvimento de software necessitava de uma disciplina capaz de garantir que o produto final não apenas atendesse aos usuários como também fosse feito dentro do prazo e do orçamento.

Procurou-se então inspiração na experiência centenária da engenharia em busca da disciplina que se fazia necessária à área de software. Afinal, sabe-se de obras civis que apresentaram problemas, mas nunca se soube de um projeto de ponte que acabou resultando em um edifício. Ocorrências semelhantes na área de software são mais comuns do que gostaríamos de admitir.

Nos últimos vinte anos, a Engenharia de Software evoluiu de um assunto obscuro para se tornar matéria de alta relevância. A importância crescente do software para as empresas e pessoas deverá aprofundar e disseminar cada vez mais este processo. Profissionais que antes denominavam programadores, hoje procuram se intitular Engenheiros de Software.

Apesar de muito software ainda ser desenvolvido de maneira empírica, a Engenharia de Software já mostrou o caminho que deve ser trilhado. Acreditamos que somente com a utilização de sólidos princípios de Engenharia de Software, teremos condições de desenvolver produtos e sistemas que atendam às expectativas dos usuários e sejam feitos dentro do prazo e do orçamento estipulado.

2.2 LINGUAGENS DE PROGRAMAÇÃO

As linguagens de programação nasceram junto com os computadores e por esta razão cada computador tinha sua própria linguagem, em geral escrita na forma binária. Por serem específicas de uma máquina recebiam o nome de "Linguagem de Máquina". Programar em linguagem de máquina não era uma tarefa propriamente agradável. A simples inserção de nova instrução poderia exigir mudanças nos endereços de todas as demais.

Um primeiro avanço foi o desenvolvimento das Linguagens Simbólicas. Estas linguagens têm a mesma estrutura da linguagem de máquina, mas as instruções passaram a ser escritas em código mnemônico e os endereços definidos através de rótulos. A transformação das instruções simbólicas em instruções de máquina e a alocação dos endereços são feitas por um programa montador. Inclusões e alterações são simplificadas porque o programa montador recalcula automaticamente todos os endereços. Estas linguagens ainda são bastante utilizadas em programas que exigem acesso aos registradores internos do processador e em aplicações que exigem alto nível de otimização em ambientes com restrições de recursos.

Uma melhoria decisiva ocorreu com o desenvolvimento das linguagens de alto nível. Voltadas para o problema e não para a máquina, deram impulso decisivo no uso em larga escala dos computadores. Programas chamados de compiladores fazem a conversão das instruções em linguagem de alto nível para a linguagem de máquina. Um mesmo programa, atendidas certas restrições, pode ser utilizado em

diferentes computadores. Para tanto, basta que cada um deles disponha do respectivo compilador para a linguagem em que o programa está escrito.

2.3 CONCEITOS DE ORIENTAÇÃO A OBJETOS

As primeiras linguagens de alto nível suportavam apenas tipos de dados pré-definidos. O desenvolvimento das linguagens orientadas a objeto resolveu este problema ao permitir que o usuário adicione novos tipos de dados à aqueles pré definidos pela linguagem. Isto é feito através do mecanismo de classes. Um objeto é uma instância de uma classe. Objetos encapsulam dados (o valor dos atributos que definem o objeto) e operações (ações que são aplicadas para mudar os atributos do objeto).

Na programação tradicional, com enfoque orientado a procedimentos, um programa descreve uma série de passos a serem executados; isto é, um algoritmo. Na programação orientada a objetos um programa descreve um sistema de objetos interagindo.

A programação orientada a objetos envolve alguns conceitos chave: abstratação, que permite simplificar a construção de grandes programas; encapsulamento, que torna mais simples a manutenção de programas e hierarquia, que torna um programa facilmente e extensível.

2.3.1 ABSTRAÇÃO

Abstração é o processo de ignorar detalhes a fim de se concentrar nas características essenciais do problema. Uma linguagem de programação é tradicionalmente considerada de "alto nível" se suportar um alto grau de abstração. Por exemplo, se considerarmos dois programas que executam a mesma tarefa, um escrito em linguagem simbólica (assembly), e um em C. O programa em linguagem simbólica conterá uma descrição extremamente detalhada do que o computador faz para executar a tarefa. Programadores, usualmente, não estão preocupados com o que acontece neste nível. O programa em C fornece uma

descrição muito mais abstrata do que o computador faz, e esta abstração torna o programa mais claro e fácil de entender.

Linguagens tradicionais suportam abstração, mas as linguagens orientadas a objeto possuem mecanismos muito mais poderosos de abstração. Para entender como, deve-se considerar os diferentes tipos de abstração.

2.3.2 ABSTRAÇÃO FUNCIONAL

A forma mais comum de abstração é a abstração funcional, que permite que detalhes a respeito do processo sejam ignorados.

Existem muitos níveis de abstração funcional. Por exemplo, é possível descrever o que um programa faz com detalhes ainda maiores do que a feita na linguagem simbólica, listando-se cada um dos passos executados pelo processador para completar uma instrução em linguagem simbólica. Por outro lado, um programa escrito na linguagem macro de uma aplicação pode descrever uma dada tarefa com um nível de abstração muito mais alto que um programa em C.

Quando um programador escreve um programa em uma dada linguagem, ele não está restrito ao nível de abstração que a linguagem fornece. Muitas linguagens permitem escrever programas a um nível mais alto de abstração funcional através do suporte a funções (também conhecidas como sub-rotinas ou procedimentos) definidas pelo usuário. Escrevendo suas própria funções, um programador pode definir novos termos para expressar o que o programa faz.

Por exemplo, vamos supor um programa que freqüentemente tem que comparar duas cadeias de caracteres, c1 e c2. Uma possibilidade seria inserir a cada comparação as instruções necessárias (aproximadamente 15 linhas de código em C) à execução desta tarefa. Uma maneira alternativa seria colocar as instruções em uma função. Cada vez que fosse necessária a comparação bastaria chamar a função com os parâmetros correspondentes, em nosso caso strncmp (c1, c2).

O uso de strcmp faz mais do que apenas diminuir a quantidade de código a ser digitado. Ela torna o programa muito mais fácil de entender, porque os detalhes de

implementação ficam escondidos. Os passos executados pela função não são importantes. O importante é que a comparação seja executada.

Funções tornam mais fáceis a implementação de programas por permitirem ao programador pensar em termos de operações lógicas, ao invés de declarações específicas da linguagem de programação.

2.3.3 ABSTRAÇÃO DE DADOS

Um outro tipo de abstração é a "abstração de dados", que permite ignorar detalhes de como um determinado tipo de dado é representado.

Por exemplo, todos os dados armazenados em um computador podem ser visualizados com números, hexadecimais ou binários. Todavia, desde que as pessoas preferem pensar em termos de números decimais, muitas linguagens suportam números dos tipos, inteiro e de ponto flutuante. Pode-se simplesmente digitar "3.1416" ao invés de alguns bytes em hexadecimal. De forma similar, Basic suporta dados do tipo cadeia de caracteres, tornando possível a execução intuitiva de operações sobre este tipo de dado, sem necessidade de conhecer-se detalhes de como ele é representado. Por outro lado, C não suporta a abstração de cadeias de caracteres, o que obriga o programador a manipular cadeias como uma série de caracteres que ocupam posições consecutivas na memória.

Abstração de dados sempre envolve algum grau de abstração funcional. Quando se executa operações sobre variáveis de um tipo sobre o qual não se conhece a representação, pode-se ignorar também os detalhes de como as operações são executadas. A forma como as operações de ponto flutuante são executadas em binário é algo com o qual os programadores C não precisam se preocupar.

Comparadas com a capacidade de abstração funcional, muitas linguagens de programação têm suporte muito limitado para a criação de novos níveis de abstração de dados. A linguagem C suporta tipos definidos pelo usuário através dos comandos struct e typedef. Este tipo de recurso é conveniente porque permite a manipulação de diversas unidades de informação como uma unidade ao invés de individualmente. Todavia, esta solução não provê qualquer vantagem conceitual.

A linguagem C trata a abstração funcional e a abstração de dados como duas técnicas distintas. Não há qualquer vantagem em termos de abstração em criar-se um novo tipo de dados sem as funções necessárias à sua manipulação.

2.3.4 CLASSES

Linguagens orientadas a objetos combinam abstração funcional com abstração de dados na forma de classes. Quando uma classe é definida são incluídas, tanto a descrição dos dados, quanto das funções que os manipulam. Uma classe é uma entidade de alto nível que pode ser utilizada sem que se conheçam detalhes dos dados ou dos procedimentos que os manipulam. Para serem utilizadas em um programa, as classes precisam ser instanciadas. Uma instância de uma classe é chamada de objeto, daí o nome de Linguagens Orientadas a Objetos. Um objeto pode ser comparado a um componente eletrônico. Conhece-se apenas a interface externa (as funções de cada pino do componente), o que está dentro pode ser ignorado.

Classes também podem suportar entidades que normalmente não seriam consideradas tipos de dados. Por exemplo, uma classe pode representar uma arvore binária e possuir todas as operações executadas neste tipo de estrutura de dados, permitindo a pesquisa, adição e remoção de valores. A utilização de objetos desta classe não exigirá do programador qualquer conhecimento dos algoritmos ou qual estrutura de dados foi utilizada na implementação.

2.3.5 ENCAPSULAMENTO

Encapsulamento é o processo de esconder os detalhes internos de uma classe para suportar o uso de abstração. Isto requer uma clara distinção entre a "interface" da classe, que deve ser pública, e sua "implementação", que deve permanecer privada. A interface de uma classe descreve o que ela pode fazer, enquanto sua implementação descreve como ela faz. Esta distinção suporta a abstração expondo apenas as propriedades relevantes da classe: o usuário enxerga um objeto em termos das operações que ele pode executar, não em termos de sua estrutura de dados. A figura 2.1 abaixo mostra a representação de uma classe.

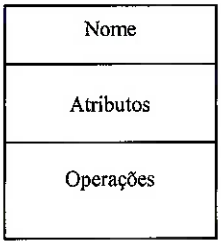

Figura 2-1 Representação de uma Classe

Os atributos que descrevem a classe ficam confinados e só podem ser acessados através da interface pública da classe. Desta forma, mudanças nos atributos de uma classe não são propagadas para outros módulos, desde que a interface pública permaneça a mesma.

2.3.6 HIERARQUIA DE CLASSES

Uma característica da programação orientada a objetos é a possibilidade de definir-se uma hierarquia de classes. Uma classe pode ser derivada de uma outra classe, tornando-se um subtipo, ou uma categoria especial, da classe inicial. Pode-se também expressar similaridades entre classes, ou defini-las como subcategorias de uma categoria mais ampla, derivando-as de uma mesma classe básica.

Identificar uma classe básica comum a diversas classes é uma forma de abstração. Uma classe básica é uma visão de alto nível de todas as classes para as quais ela serve de base. Ela especifica o que as classes derivadas têm em comum. Esta técnica de abstração permite a visão de umas poucas entidades ao invés de um grande número, o que facilita o entendimento do problema. Um exemplo é a classificação usada no reino animal: podemos nos referir à "mamíferos" ao invés de, por exemplo, cavalos, vacas e tigres.

Uma classe base é chamada de *generalização* de um grupo de classes. Uma classe derivada é chamada de *especialização* de uma outra classe. Uma classe derivada identifica um subtipo de um tipo previamente definido, e o descreve em termos de suas características adicionais. Por exemplo, o cavalo é um mamífero, mas eles têm certas características não encontradas em outros mamíferos.

Existem dois benefícios práticos de definir-se uma hierarquia de classes: a classe derivada pode compartilhar o código da classe base, ou pode compartilhar a interface da classe base. Estes dois benefícios não são mutuamente exclusivos, embora uma hierarquia projetada para reutilização de código apresente freqüentemente características diferentes de uma projetada para fornecer uma interface comum.

2.3.7 HERANÇA E POLIMORFISMO

Herança é o processo pelo qual uma classe derivada herda todos os atributos e métodos da classe base. Este processo é automático e permite a reutilização direta de código, constituindo-se numa das principais diferenças das linguagens orientadas a objeto em relação a programação convencional. Consideremos, por exemplo, uma aplicação de folha de pagamento de uma empresa com os tipos de funcionários: Mensalista e Horista. Poderia ser criada uma classe Empregado que teria todos os atributos e métodos comuns a todos os empregados. Desta classe base, poderiam ser derivadas as classes Mensalista e Horista, que herdariam todos os atributos e métodos da classe Empregado e adicionariam os atributos e métodos específicos às classes derivadas.

Polimorfismo é a habilidade de se efetuar a chamada de um método de uma classe sem especificar o tipo exato do objeto utilizado na chamada. A palavra "polimorfismo" significa "a habilidade de assumir muitas formas", e se refere a uma característica das linguagens orientadas a objeto que permitem que uma única declaração efetue a chamada de diferentes métodos. Pressman (2001) apresenta o exemplo abaixo de uma aplicação para desenhar quatro diferentes tipos de gráficos: linha, pizza, histogramas e diagramas de Kiviat. Em uma aplicação convencional, uma vez coletados os dados, seriam feitas chamadas a módulos específicos para cada tipo de gráfico:

```
case of graphtype:
        if graphtype = linegraph then DrawLineGraph(data);
        if graphtype = piechart then DrawPieChart(data);
        if graphtype = histogram then DrawHisto(data);
        if graphtype = kiviat then DrawKiviat(data);
```

end case;

A adição de um novo tipo de gráfico exigirá alteração na lógica de controle.

Este problema pode ser resolvido com o uso de programação orientada a objetos. Neste caso, pode-se definir uma classe geral chamada Graph e partir desta derivar classes específicas para cada tipo de gráfico. Cada classe derivada teria uma operação chamada Draw que seria chamada dependendo do tipo de gráfico desejado. Neste caso a implementação seria:

Graphtype Draw;

Se um novo tipo de gráfico for adicionado, uma nova classe derivada seria criada com sua própria operação Draw. A implementação permaneceria a mesma.

2.4 ENGENHARIA DE SOFTWARE ORIENTADA A OBJETOS

2.4.1 PROCESSOS DE SOFTWARE

Processos e metodologias formam o arcabouço em que se baseia a Engenharia de Software. Metodologias (estudo de métodos) descrevem um conjunto independente de atividades. Processos conectam estas atividades de forma a que objetivos predefinidos sejam atingidos.

O uso de tecnologias orientadas a objetos, da mesma forma que nos métodos tradicionais, só é eficaz quando suportada por processos e métodos de Engenharia de Software. Escrever programas em uma linguagem orientada a objetos não garante que os conceitos fundamentais da orientação a objetos estão sendo utilizados. Pode-se, por exemplo, escrever um programa em C++ sem usar os conceitos de encapsulamento, herança e polimorfismo. A este respeito Berard (1993) apud Pressman (2001) escreveu:

"Os benefícios da tecnologia orientada a objetos são obtidos apenas quando sua utilização for feita no começo e durante todo o processo de engenharia de software. Aqueles que estão considerando a utilização de tecnologia orientada a objetos devem avaliar seu impacto em todo o processo de engenharia de software. Simplesmente empregar programação orientada a objetos (OOP) não irá produzir os melhores resultados. Engenheiros de software e seus gerentes devem considerar itens como análise de requisitos orientada a objetos (OORA), projeto orientado a objetos (OOD), análise de domínio orientado a objetos (OODA), sistemas de base de dados orientados a objetos (OODBMS) e engenharia de software auxiliada por computador orientada a objetos (OOCASE)".

Alguns dos aspectos da análise e projeto orientados a objetos diferem dramaticamente dos utilizados na metodologia tradicional. Um destes aspectos refere-se ao modelo do processo de engenharia de software que deve ser utilizado.

A natureza das tecnologias orientadas a objetos é tal que torna muito difícil a identificação em uma única etapa de todas as classes necessárias a um sistema ou aplicação. O processo ocorre por iterações, conforme a evolução do processo. Assim, para ser eficaz, o processo de Engenharia de Software deve reconhecer a natureza iterativa do desenvolvimento orientado a objetos.

A escolha de um processo de engenharia de software adequado à orientação a objetos é o primeiro passo na direção da obtenção de produtos e sistemas de alta qualidade construídos dentro do prazo e do orçamento. Deve-se cuidar para que qualquer que seja o processo utilizado, cada iteração produza executáveis que possam ser testados e verificados.

Rowlett (2001) propõe que um processo robusto de software deve ser:

1. Completo

   "Um processo é completo quando inclui todas as etapas necessárias para progredir de uma especificação mínima até o produto final. Todas as etapas devem ter seu início e fim, claramente identificados. A equipe de projeto deve sempre saber identificar em qual etapa do projeto está trabalhando".

2. Consistente

   "Um processo é consistente quando, exceto pela primeira atividade, cada entrada para uma atividade do processo deve ter sido produzida pela etapa precedente, e nenhuma etapa deve produzir resultados que não sejam entradas para a etapa posterior".

3. Verificável

   "Todas os produtos de cada etapa do processo devem ser verificados em relação aos requisitos do usuário. A verificação não deve ser deixada para a fase de teste. Testar para verificar a correção assume incorretamente que todos os defeitos serão descobertos durante o teste; que haverá tempo para refazer e testar novamente; e que os testes verificarão os requisitos".

4. Rastreável

   "Um processo é rastreável quando todos os produtos podem ser associados com um requisito e para cada requisito é possível identificar o produto que o implementa"

5. Incrementável

   "O processo deve permitir que os requisitos sejam particionados de forma que o esforço para a construção da aplicação em partes seja aproximadamente o mesmo de construi-lo como uma única unidade. Este enfoque permite grande flexibilidade na distribuição do trabalho".

6. Testável

   "Para ser testável, um processo precisa permitir que os casos de teste sejam desenvolvidos a partir dos requisitos. Esta condição exige que, qualquer mudança nos requisitos precisa ser refletida não só no produto final como também em todos os produtos intermediários".

2.4.2 MÉTODOS DE ANALISE ORIENTADA A OBJETOS

A definição de uma metodologia é um dos fatores críticos para o sucesso no desenvolvimento de sistemas orientados a objeto. A crescente popularidade destas tecnologias fez surgir uma grande quantidade de métodos orientados a objeto. Profissionais de tecnologia de informação não podem reclamar de escassez nesta área. Método de Coad e Yourdon (um dos mais antigos e conforme Pressman (2001), já abandonado pelos autores), método de Booch, método de Rumbaugh e método de Jacobson são apenas alguns dos mais conhecidos.

De acordo com Pressman (2001), embora diferindo na terminologia e nos passos, todos os métodos de Análise Orientada a Objetos são bastante similares. Para executar análise orientada a objetos, um engenheiro de software deve executar os seguintes passos genéricos:

1. Obter do usuário os requisitos do sistema;
2. Identificar os cenários ou casos de uso;
3. Selecionar as classes e objetos usando os requisitos básicos como guia;
4. Identificar os atributos e operações de cada classe;
5. Construir um modelo de relacionamento de objetos;
6. Construir um modelo comportamental;
7. Revisar o modelo de análise contra os cenários ou casos de uso.

2.4.3 UNIFIED MODELING LANGUAGE

Com o objetivo de obter uma forma padronizada de representação de sistemas, Grady Booch, James Rumbaugh e Ivar Jacobson combinaram seus esforços para criar uma linguagem que ficou conhecida como Unified Modeling Language (UML). A UML é detalhadamente descrita em Booch (1999)

A UML é uma linguagem de modelagem. Não é um método. É uma linguagem que pode ser usada por qualquer método para representar um sistema, sendo neutra do ponto de vista metodológico.

A UML define nove tipos de diagramas. Cada diagrama permite a visualização do sistema de diferentes perspectivas.

A UML é uma linguagem padronizada pela Object Management Group (OMG), consorcio sem fins lucrativos dedicado ao desenvolvimento de padrões para interoperabilidade entre aplicações.

2.5 PROTOCOLOS DE COMUNICAÇÃO

As tarefas de um sistema de comunicação incluem o envio de informações que sejam corretas, que estejam na ordem própria e que possam ser entendidas pelo receptor. Para executar estas funções, um subsistema de hardware/software precisa trabalhar de forma cooperativa em conformidade com um conjunto de regras chamadas de protocolos.

Dicionários definem "protocolo" como sendo um conjunto de regras e cerimonias através das quais diplomatas e chefes de estado se comunicam. As regras do protocolo diplomático garantem que a comunicação seja completa e corretamente entendida, por ambas as partes. Em comunicação de dados, os protocolos executam uma função similar, e seu uso é quase tão complexo quanto o uso dos protocolos diplomáticos.

Para apoiar o desenvolvimento de famílias de protocolos, a International Organization for Standardization (ISO) desenvolveu um modelo no qual cada protocolo que forma a família é uma "camada" que executa certas funções para o protocolo (camada) acima dele. Este modelo é chamado de Modelo de Referência para Interconexão de Sistemas Abertos ou simplesmente modelo OSI.

O modelo OSI não especifica os detalhes de cada protocolo, mas especifica uma forma de projetar famílias de protocolos, isto é, que camadas devem estar presentes e que funções cada camada deve executar. Duas famílias de protocolos compatíveis com o modelo OSI não conseguem necessariamente se comunicar um com o outro. Além disto, muitas famílias de protocolos são pouco aderentes ao modelo OSI.

Em adição ao modelo, na década de 1980 a ISO desenvolveu um conjunto completo de protocolos para cada camada do modelo OSI. Estes protocolos são usualmente conhecidos como pilha de protocolos ISO. O modelo OSI pode ser usado para descrever virtualmente qualquer pilha de protocolos, enquanto a pilha de protocolos ISO usualmente se refere a protocolos padronizados pela ISO.

2.6 MODELO DE REFERÊNCIA OSI

O modelo OSI define sete camadas independentes de software. Cada camada executa um conjunto diferente de funções e é independente de qualquer outra camada. As sete camadas do modelo OSI são as seguintes:

1. Física - Define as características físicas do meio de comunicação;

2. Enlace - Fornece serviços confiáveis de envio de dados através do meio físico;

3. Rede - Gerencia as conexões através da rede para os níveis superiores;

4. Transporte – Fornece conexões ponta a ponta com detecção e correção de erros;

5. Sessão – Gerencia sessões entre aplicações;

6. Apresentação – Padroniza a apresentação de dados para as aplicações;

7. Aplicação – Consiste dos programas de aplicação que usam a rede.

2.7 MODELO DE REFERÊNCIA EPA

Para atender as necessidades de Sistemas de Supervisão e Controle, que requerem tempos particularmente curtos de reação em redes de comunicação de baixa velocidade, a International Eletrotechnical Commission (IEC) desenvolveu um modelo de referência de três camadas que recebeu o nome de Enhanced Performance Architecture (EPA). Este modelo utiliza apenas os níveis físico (nível 1), enlace (nível 2) e aplicação (nível 3). Os protocolos da IEC baseados no modelo de referência EPA são definidos nas séries de normas internacionais IEC 60870-5

2.8 PROTOCOLO HDLC/AM

As unidades terminais remotas Microlab da CTEEP utilizam uma versão simplificada do protocolo HDLC, adaptado ao modo assíncrono (o sufixo AM) não balanceado. Este protocolo segue o modelo de referência EPA e sua camada de aplicação é otimizada para uso em canais de baixa velocidade. A aplicação consegue manter o tempo de varredura abaixo de 2 segundos, utilizando linhas de comunicação de 1200 bits por segundo.

Na estrutura do protocolo HDLC/AM a estação mestre possui um conjunto receptor primário/emissor primário para cada UTR. A estação remota possui um

conjunto emissor secundário/receptor secundário. A estação mestre emite ordens e as estações remotas emitem respostas. A taxa de antecipação é fixada em 1, o que implica que cada estação deverá esperar o recebimento do reconhecimento de uma mensagem anterior antes de emitir a seguinte. Toda mensagem emitida deve ser reconhecida por uma resposta.

Todas as mensagens emitidas pela estação mestre devem conter o endereço da estação remota. As mensagens emitidas pela estação remota em direção a estação mestre devem conter o endereço da estação remota.

Os autômatos das estações remotas asseguram o serviço de troca de informação. Os autômatos da estação mestre asseguram o serviço de troca de informação e o serviço de iniciação. O serviço de iniciação deve preceder qualquer troca de informação.

A estação mestre pedindo iniciação transmite uma mensagem SNRM (Set Normal Response Mode). A estação remota, recebendo SNRM inicia suas variáveis e responde com uma mensagem UA (Unnumbered Acknowledge). A estação remota só transmite informação quando interrogada. Se interrogada antes de ser iniciada, responderá com uma mensagem DM (Disconnected Mode).

O protocolo suporta três tipos de mensagens:

- Mensagem de informação (I)

  Este tipo de mensagem é usado para troca de informações entre as estações. Toda mensagem I contém um número de seqüência NS e um número quitação NR, usado para reconhecimento da mensagem I recebida. Nas mensagens com erro de seqüência, o campo de dados é desprezado.

- Mensagem de supervisão (S)

  Estas mensagens são para reconhecer uma mensagem I. Contêm um número NR de quitação, tendo mesmo significado que na mensagem I.

- Mensagem de Controle não Seqüencial (U)

São utilizadas para mensagens de iniciação SNRM, para a mensagem UA de resposta e para o pedido de iniciação DM.

As figuras 2-2 e 2-3 mostram, respectivamente, o formato das mensagens de informação e de supervisão/controle não seqüencial, enquanto que a figura 2-4, a estrutura do byte de controle:

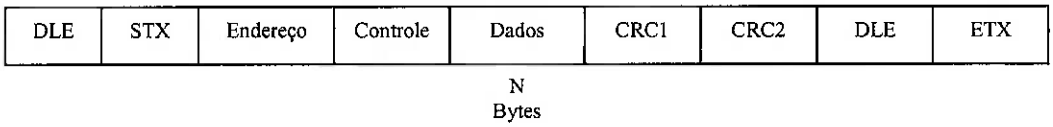

Figura 2-2 Mensagem de Informação

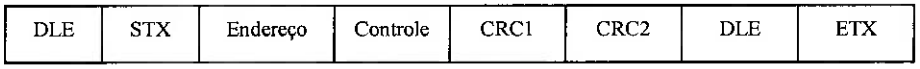

Figura 2-3 Mensagem de Supervisão/Controle não Numerado

Tipo de Mensagem	Ordens	Respostas	Codificação Binária 7 6 5 4 3 2 1 0			
Informação	Informação	Informação	NR	1	NS	1
Supervisão	RR	RR	NR	1	0 0 0 1	
Controle	SNRM	-	1 0 0	1	0 0 1 1	
		UA	0 1 1	1	0 0 1 1	
		DM	0 0 0	1	0 0 1 1	

Figura 2-4 Estrutura do Byte de Controle

Os quadros são delimitados por pela seqüência de caracteres de controle ASCII DLE-STX e DLE-ETX. Quando o emissor encontrar no campo de dados uma informação igual a DLE, um DLE adicional será inserido. O receptor fará a operação inversa quando encontrar dois caracteres DLE em seqüência

A verificação da mensagem é feita pelo método do Caracter de Redundância Cíclica, conforme mostrado em McNamara (1988).

A camada de aplicação do protocolo HDLCAM utiliza o conceito de Código de Função para implementar as unidades de dados dos serviços de aplicação. Diferentes estruturas de dados são utilizadas para cada função. O envio de informações pode ser feito por solicitação explícita da estação primária (envio por demanda) ou de forma espontânea pela estação secundária, quando alguma mudança for detectada (envio por exceção). O protocolo suporta variáveis binárias (indicações de estado), variáveis analógicas de 12 bits, variáveis digitais de 16 bits, telecomando do tipo "selecione antes de operar" e telecomando de execução direta. O protocolo de comunicação HDLC/AM está detalhadamente descrito em Microlab (1984).

2.9 Protocolo do Sinalizador Registrador de Alarmes

O Sinalizador Registrador de Alarmes é um equipamento desenvolvido pela CTEEP com funções de anunciador de alarmes e de seqüência de eventos. Cada unidade consegue supervisionar 32 entradas. Possui capacidade de sincronizar seu relógio interno com um sistema GPS com exatidão de 1 milisegundo. A resolução do relógio e a capacidade de separação de eventos também são de 1 milisegundo.

Os eventos são enviados utilizando-se o protocolo MODBUS no modo RTU. É utilizado o código de função 24 – Read FIFO Queue.

O MODBUS é um protocolo aberto e sua especificação pode ser encontrada em Modicon (1996).

2.10 Protocolo IEC 60870-5-101

O protocolo IEC 60870-5-101 é usado por equipamentos e sistemas de telecontrole que utilizam transmissão de dados serial para a monitoração e controle de processos geograficamente espalhados. A definição do protocolo utiliza padrões da série de documentos IEC 60870-5 para definir perfis funcionais para as tarefas básicas de telecontrole.

O protocolo IEC 60870-5 é baseado no modelo de referência "Enhanced Performance Architecture" (EPA), conforme especificado na cláusula 4 da norma IEC 60870-5-3.

O nível físico utiliza recomendações da ITU-T que suportam transmissão no meio requerido de forma binária, simétrica e sem armazenamento temporário, de forma a preservar a integridade do método de codificação do bloco do nível de enlace.

O nível de enlace consiste de uma série de procedimentos de enlace e de uma seleção de formatos de quadros de transmissão que fornecem o necessário suporte para a transmissão de forma confiável de Unidades de Dados do Serviço de Aplicação.

O nível de aplicação contém um número de "Funções de Aplicação" que envolve a transmissão de Unidades de Dados do Serviço de Aplicação entre a fonte e o destino.

A figura 2-5 mostra o modelo Enhanced Performance Architecture (EPA) e as seleções das definições dos padrões utilizadas no protocolo IEC 60870-5-101.

Seleção de Funções de Aplicação da norma IEC 60870-5-5	Processo do Usuário
Seleção de Unidades de Serviço de Dados da norma 60870-5-3	Nível de Aplicação (camada 7)
Seleção de Elementos de Informação da norma 60870-5-4	
Seleção de Procedimentos de Enlace da norma IEC 60870-5-2	Nível de Enlace (camada 2)
Seleção de Formatos de Quadros da norma IEC 60870-5-1	
Seleção de Recomendações da ITU-T	Nível Físico (camada 1)

Figura 2-5 Normas utilizadas na definição do protocolo IEC 60870-5-101

3. MODELAGEM DO SISTEMA

3.1 CARACTERIZAÇÃO DO DOMÍNIO DO PROBLEMA

Nos seus primórdios, a industria de energia elétrica era caracterizada por sistemas isolados cuja operação pouco exigia em termos de coordenação. O crescimento da industria com a construção de grandes usinas geradoras, em muitos casos distantes dos principais centros de carga, exigiu a construção de uma malha de transmissão de energia bastante complexa.

A operação desta malha foi inicialmente feita através de operadores nas subestações com a coordenação de despachantes tendo com recurso básico a comunicação através de telefone. Esta forma de operação era bastante deficiente, não sendo capaz de prover os despachantes com uma visão em tempo real da situação do sistema elétrico de potência.

A solução deste problema só se tornou possível com a evolução da tecnologia de informação, que permitiu o desenvolvimento de sistemas capazes de coletar, comandar e transmitir as informações do processo elétrico para Centros de Operação, onde o estado do processo poderia ser visualizado e controlado.

3.1.1 SISTEMAS DE SUPERVISÃO E CONTROLE

Desenvolvidos a partir da década de 1970, estes sistemas vêm passando por um processo de melhoria contínua, através da incorporação dos avanços tecnológicos obtidos nas áreas de micro eletrônica e tecnologia da informação.

Os primeiros sistemas possuíam uma arquitetura monolítica, com os aplicativos, gerenciadores de base de dados, interface homem máquina e unidades terminais remotas desenvolvidas por um único fabricante, sem qualquer possibilidade de interoperar com outros equipamentos e sistemas.

A evolução da tecnologia de software permitiu aos desenvolvedores de Sistemas de Supervisão e Controle a utilização de uma série de ferramentas de prateleira disponíveis comercialmente. Gerenciadores de base de dados, interfaces gráficas, historiadores, etc. passaram a ser adquiridos no mercado e incorporados aos produtos. Este processo permitiu que os desenvolvedores se concentrassem na aplicação propriamente dita, deixando para outros o desenvolvimento do software de suporte.

Atualmente, a utilização de Sistemas de Supervisão e Controle em empresas de serviços de eletricidade é um fato comum. A figura 3-1 apresenta a arquitetura típica destes sistemas.

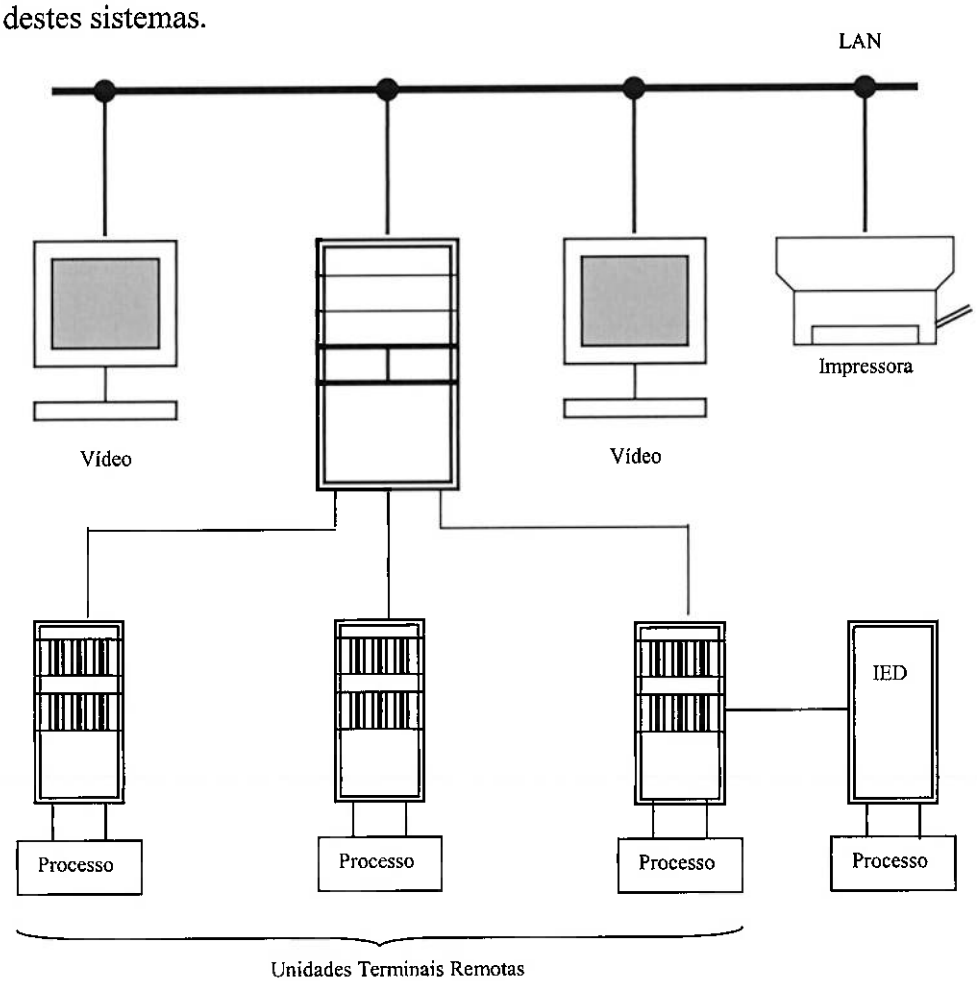

Figura 3-1 Arquitetura típica de um sistema SCADA

Medições e indicações de estado do processo são coletadas por Unidades Terminais Remotas e enviadas através de linhas de comunicação para servidores

instalados nos Centros de Operação. Os servidores se encarregam das funções de tratamento, registro, recuperação e apresentação dos dados adquiridos pelas Unidades Terminais Remotas. No sentido inverso, comandos gerados nos Centros de Operação são enviados às Unidades Terminais Remotas, que se encarregam da execução dos mesmos.

Algumas UTR possuem suporte para a aquisição de dados provenientes de Dispositivos Eletrônicos Inteligentes (IED). Transdutores Digitais, Reles de Proteção e Controladores Programáveis estão entre os principais tipos de IED utilizados.

3.1.2 UNIDADES TERMINAIS REMOTAS

Responsáveis pela interface com o processo, eram inicialmente estreitamente acopladas aos Sistemas de Supervisão e Controle. Gradualmente, passaram a ser fabricadas por fornecedores independentes. Fator importante neste sentido foi a padronização dos protocolos de comunicação para equipamentos de telesupervisão e telecontrole.

Protocolos como IEC 60870-5-101, DNP 3.0 e MODBUS, deram impulso ao processo de desverticalização do fornecimento de Sistemas de Supervisão e Controle.

Este processo apresentou excelentes resultados, fazendo surgir uma legião de produtores de Sistemas de Supervisão e Controle que não fabricam remotas e vice-versa. O aumento da competição permitiu ganhos significativos em termos de preços e de qualidade.

As UTR são encontradas em duas arquiteturas básicas: centralizada e distribuída.

As figuras 3-2 e 3-3 apresentam exemplos de cada uma destas arquiteturas.

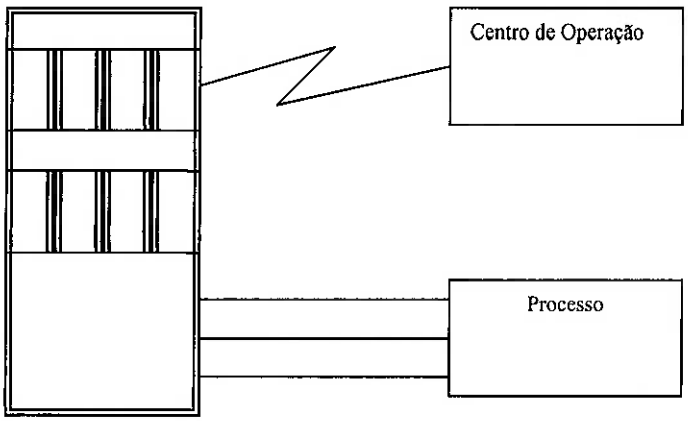

Figura 3-2 Arquitetura Centralizada

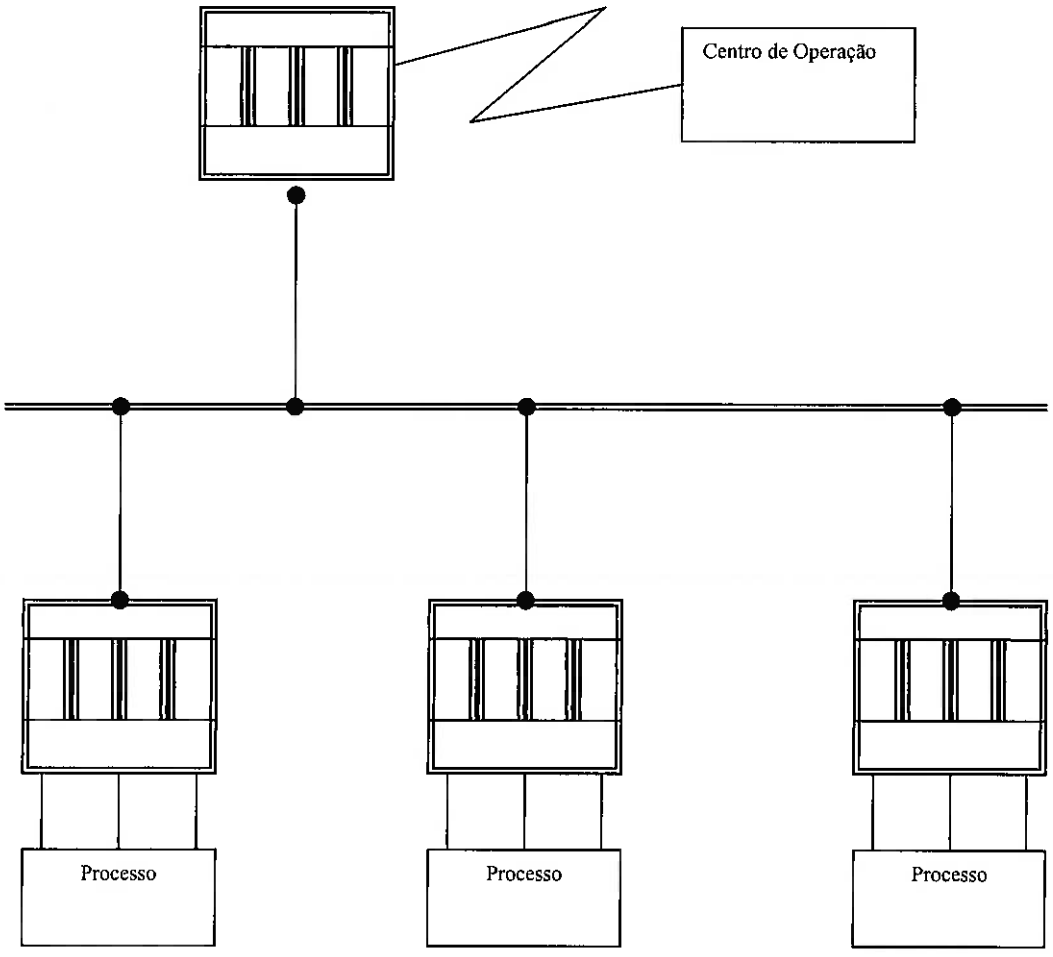

Figura 3-3 Arquitetura Distribuída

Uma UTR, independente de sua arquitetura, pode ser dividida em duas partes principais, estreitamente acopladas:

- Uma unidade operativa diretamente conectada ao processo executa funções de aquisição de dados e de comando;
- Uma unidade de comunicação que envia dados e recebe comandos dos sistemas SCADA.

As Unidades Terminais Remotas atualmente disponíveis no mercado são verdadeiros computadores de subestação. Utilizam microprocessadores de alta velocidade e sistemas operacionais de tempo real que permitem a implementação de uma série de funções avançadas. Dentre estas podemos destacar: comunicação com vários Centros de Operação, adição de selo de tempo nos eventos, conversão para unidades de engenharia, Interface Homem Máquina e automatismo.

Todavia, todas estas facilidades têm um preço: uma Unidade Terminal Remota típica do sistema de potência da CTEEP tem um custo estimado entre US$35.000 e US$50.000, dependendo do porte.

3.1.3 SISTEMA DE TRANSMISSÃO DA CTEEP

A Companhia de Transmissão de Energia Elétrica Paulista foi inicialmente constituída a partir da cisão parcial da Companhia Energética de São Paulo (CESP), tendo dela incorporado todos os ativos de transmissão. Posteriormente, em 09/11/2001 incorporou a Empresa Paulista de Transmissão de Energia (EPTE), que por sua vez foi constituída a partir da cisão parcial da Eletricidade de São Paulo (ELETROPAULO).

O sistema de transmissão da CTEEP consiste de mais de 18.000 km de circuitos e 99 subestações, operando em tensões de 440, 345, 230, 138, 88 e 69 kV, sendo a capacidade de transformação instalada superior a 35.000 MVA.

3.1.4 SISTEMA DE SUPERVISÃO E CONTROLE DA CTEEP

A operação do sistema de potência da **CTEEP** é coordenada a partir de um Centro de Operação do Sistema - COS, localizado na S/E de Bom Jardim, em Jundiaí, e de três Centros Regionais de Operação - CRO, localizados em Bauru - CROB, Cabreúva - CROC e São Paulo - CRO-SP.

Todas as funções de supervisão e controle do SSC são suportadas pelo sistema Ranger, SCADA/EMS fornecido pela ABB Network Management. A arquitetura do sistema é baseada no modelo cliente-servidor, com os servidores de aplicação instalados no COS e os servidores de comunicação instalados nos CRO. Cabe a estes últimos a tarefa de efetuar a varredura das UTR e enviar os dados coletados para os servidores de aplicação bem como receber os comandos a serem enviados às UTR.

A comunicação com as UTR é feita através de linhas privativas com velocidade de 1200 bits por segundo em modo não balanceado, com os servidores de comunicações atuando como mestre. São utilizados os protocolos HDLC/AM, IEC 60870-5-101, e DNP 3.0. As UTR utilizadas são de fabricação MICROLAB, CMW e BCM e não possuem recursos de Seqüência de Eventos.

O Controle Automático de Geração (CAG) no estado de São Paulo é de responsabilidade da CTEEP. Para implementar esta função, as UTR das usinas incluídas no CAG estão conectadas a Interfaces de Controle de Geração (ICG). Estes equipamentos convertem os valores de variação de potência requeridos pelo CAG em pulsos, enviado-os em seguida à reguladores de velocidade das Unidades Geradoras.

Para atender às necessidades de supervisão e controle das empresas de geração, também criadas no processo de cisão da CESP, incluindo a CESP remanescente, existem Centros Regionais de Operação da Geração instalados em Jupiá, Chavantes e Bariri.

Em todos os Centros de Operação, estão colocadas à disposição dos despachantes Estações de Trabalho interligadas em redes locais (LAN), que por sua vez são

interligadas aos servidores de aplicação através de uma WAN. Esta mesma WAN é utilizada para interligar os servidores de comunicações com os servidores de aplicação.

O sistema possui um total de 116 UTR instaladas em 92 subestações. São supervisionados aproximadamente 24.000 pontos de indicação de estado e 8000 pontos de telemedição. Os telecomandos de disjuntores/seccionadoras somam 1740 e os de tap de transformador 109.

A figura 3-4 mostra a arquitetura do SSC.

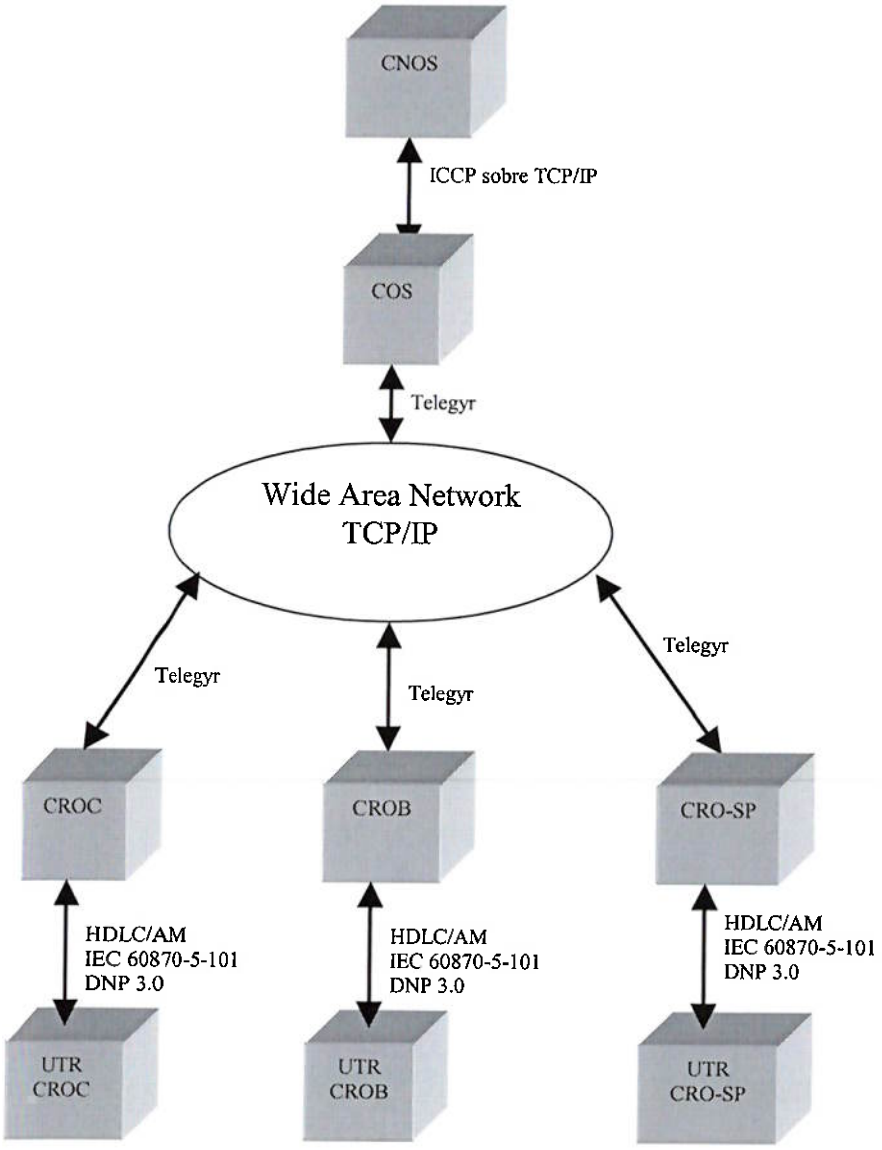

Figura 3-4 Sistema de Supervisão e Controle da CTEEP

3.2 METODOLOGIA DE MODELAGEM

Para a modelagem do Servidor de Comunicação, foi utilizada a seguinte metodologia:

1. Definição dos requisitos;
2. Definição dos casos de uso e construção dos respectivos diagramas;
3. Identificação das classes através da análise gramatical dos requisitos;
4. Identificação dos atributos e operações de cada classe;
5. Organização das classes em hierarquias;
6. Elaboração dos diagramas de classes;
7. Elaboração dos diagramas de seqüência;
8. Definição da arquitetura do sistema.

3.3 REQUISITOS DO SISTEMA

O Servidor de Comunicações deverá ser uma unidade independente capaz de interoperar com Unidades Terminais Remotas (UTR) Microlab e com Sinalizadores Registradores de Alarmes (SRA), concentrando as informações adquiridas em Bases de Dados.

Deverá possuir no mínimo 3 portas seriais padrão RS 232 para comunicação com sistemas supervisórios utilizando protocolo IEC 60870-5-101. Cada porta serial deverá ter sua própria Base de Dados de forma a permitir a seleção de quais informações devem ser enviadas para cada supervisório.

A comunicação com as Unidades Terminais Remotas Microlab deverá ser feita através de porta serial padrão RS 485 operando em multipontos utilizando o protocolo HDLC/AM.

Para comunicação com os Sinalizadores Registradores de Alarmes, deverão estar disponíveis no mínimo duas portas seriais padrão RS 485 também operando em multipontos, utilizando protocolo MODBUS.

O sistema deverá dispor de recursos para adequar-se às diferentes configurações das Unidades Terminais Remotas e Sinalizadores Registradores de Alarmes.

O hardware a ser utilizado deverá ser compatível com o padrão IBM PC, processador de 450 MHz, 128 MB de memória, gabinete do tipo industrial padrão 19 polegadas e alimentação em 48 Volts, corrente contínua.

Os aplicativos deverão ser desenvolvidos em C++ e suportados pelo sistema operacional de tempo real QNX 6.x.

A figura 3-5 ilustra as funcionalidades requeridas:

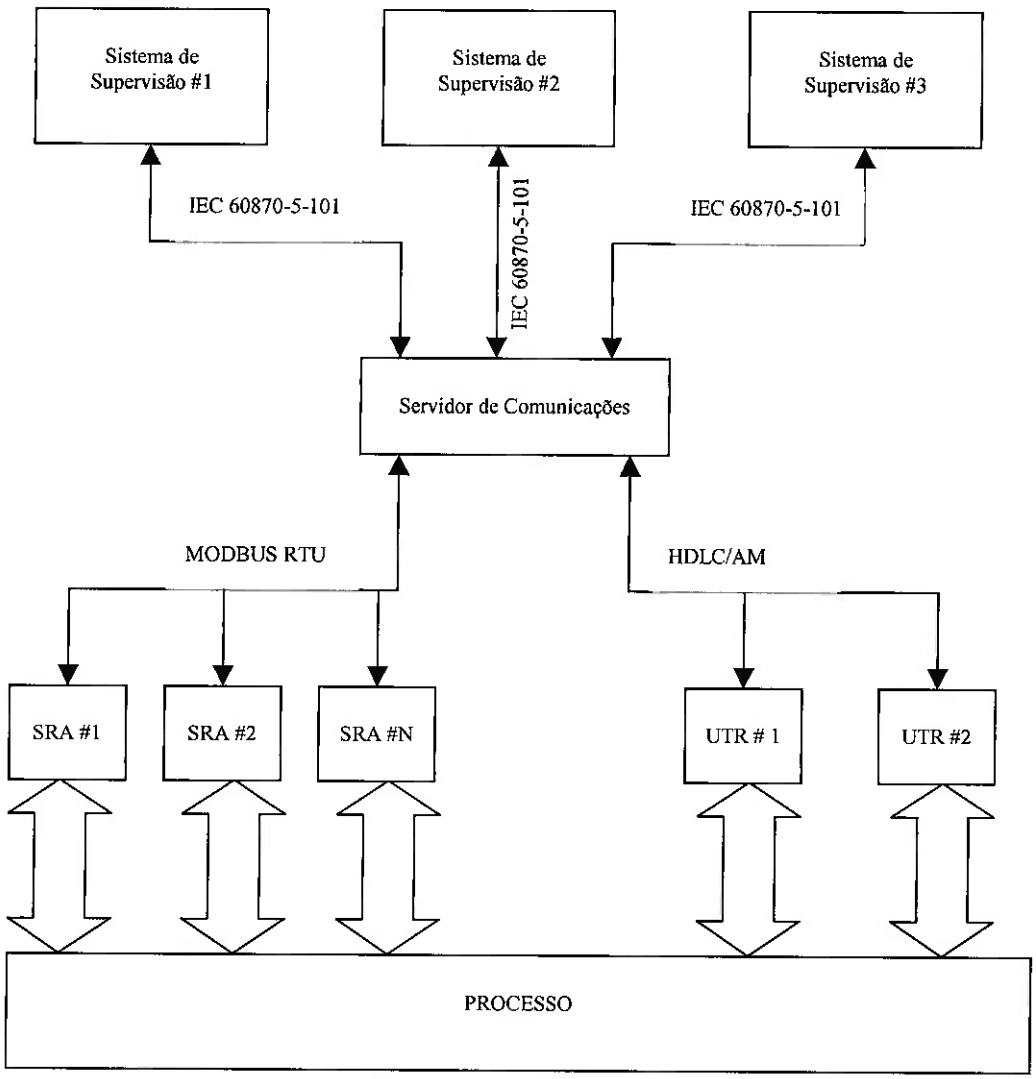

Figura 3-5 Funcionalidades do Sistema

3.3.1 REQUISITOS DO PROTOCOLO IEC 60870-5-101

O perfil a ser implementado no protocolo IEC 60870-5-101 deverá suportar as seguintes Unidades de Serviço de Aplicação de Dados (Application Service Data Units - ASDU) definidas em IEC (1995) e IEC (2001) mostradas na tabela abaixo:

Identificador do Tipo da ASDU	Rótulo da ASDU	Descrição
1	M_SP_NA_1	Indicação de estado simples
5	M_ST_NA_1	Indicação de posição de tap
9	M_ME_NA_1	Valor medido, normalizado
30	M_SP_TB_1	Indicação de estado simples com selo de tempo
45	C_SC_NA_1	Comando simples
47	C_RC_NA_1	Comando de tap
48	C_SE_NA_1	Comando de ajuste de ponto de operação
70	M_EI_NA_1	Fim de inicialização
100	C_IC_NA_1	Comando de interrogação
103	C_CS_NA_1	Comando de sincronização de relógio
105	C_RP_NA_1	Comando de inicialização
106	C_CD_NA_1	Comando de aquisição de atraso

3.3.2 REQUISITOS DO PROTOCOLO MODBUS

A implementação do protocolo MODBUS deverá suportar os códigos de função mostrados na tabela abaixo:

Função	Descrição
03	Leitura de registro
06	Escrita de valor em registro
23	Leitura e escrita em registros
24	Leitura de fila

3.3.3 REQUISITOS DO PROTOCOLO HDLC/AM

O protocolo HDLC/AM deverá suportar os códigos de operação mostrados na tabela abaixo:

Código Operação (hex)	Descrição
00	Reinicialização da UTR
01	Mudanças de estado da UTR
0F	Pedido de estado da UTR
13	Carga de banda morta
41	Pedido de envio de mudanças de TSI
42	Integridade de TSI
43	Sinaliza último bloco de resposta do código de operação 42
51	Pedido de mudanças de TSS
52	Integridade de TSS
53	Sinaliza último bloco de resposta do código de operação 52
92	TCM – Seleção para ligar
93	TCM – Seleção para desligar
94	TCM – Ligar
95	TCM – Desligar
96	Execução de comando de aumentar
97	Execução de comando de diminuir
C2	Integridade de TMD
C3	Sinaliza último bloco de resposta do código de operação C2
C9	Pedido de envio de mudanças de TMT
CA	Integridade de TMT
CB	Sinaliza último bloco de resposta do código de operação CA
D1	Pedido de envio de mudanças de TMA
D2	Integridade de TMA
D3	Sinaliza último bloco de resposta do código de operação D2

3.4 DIAGRAMAS DE CASOS DE USO

Os Casos de Uso definem os cenários de utilização do sistema a ser construído e constituem a primeira etapa da análise orientada a objetos. Sua construção começa com a identificação dos atores (pessoas ou dispositivos) e dos papéis que irão representar quando o sistema estiver em operação. Os casos de uso descrevem de que formas os atores interagem com o sistema. Jacobson (1992) apud Pressman (2001) sugere algumas questões que os casos de uso devem responder:

- Quais são as principais tarefas ou funções executadas pelo ator?
- Qual informação do sistema o ator irá adquirir, produzir ou alterar?
- O ator deverá informar ao sistema a respeito de mudanças no ambiente externo?
- Que informações o ator deseja do sistema?
- O ator deseja ser informado a respeito de mudanças inesperadas?

Conforme Booch (1999), diagramas de casos de uso são utilizados em UML para modelar os aspectos dinâmicos de um sistema. Seus principais usos estão na modelagem do contexto de um sistema e na modelagem dos requisitos de um sistema.

As figuras 3-6 a 3-10 apresentam os Diagramas de Caso de uso para o Servidor de Comunicações.

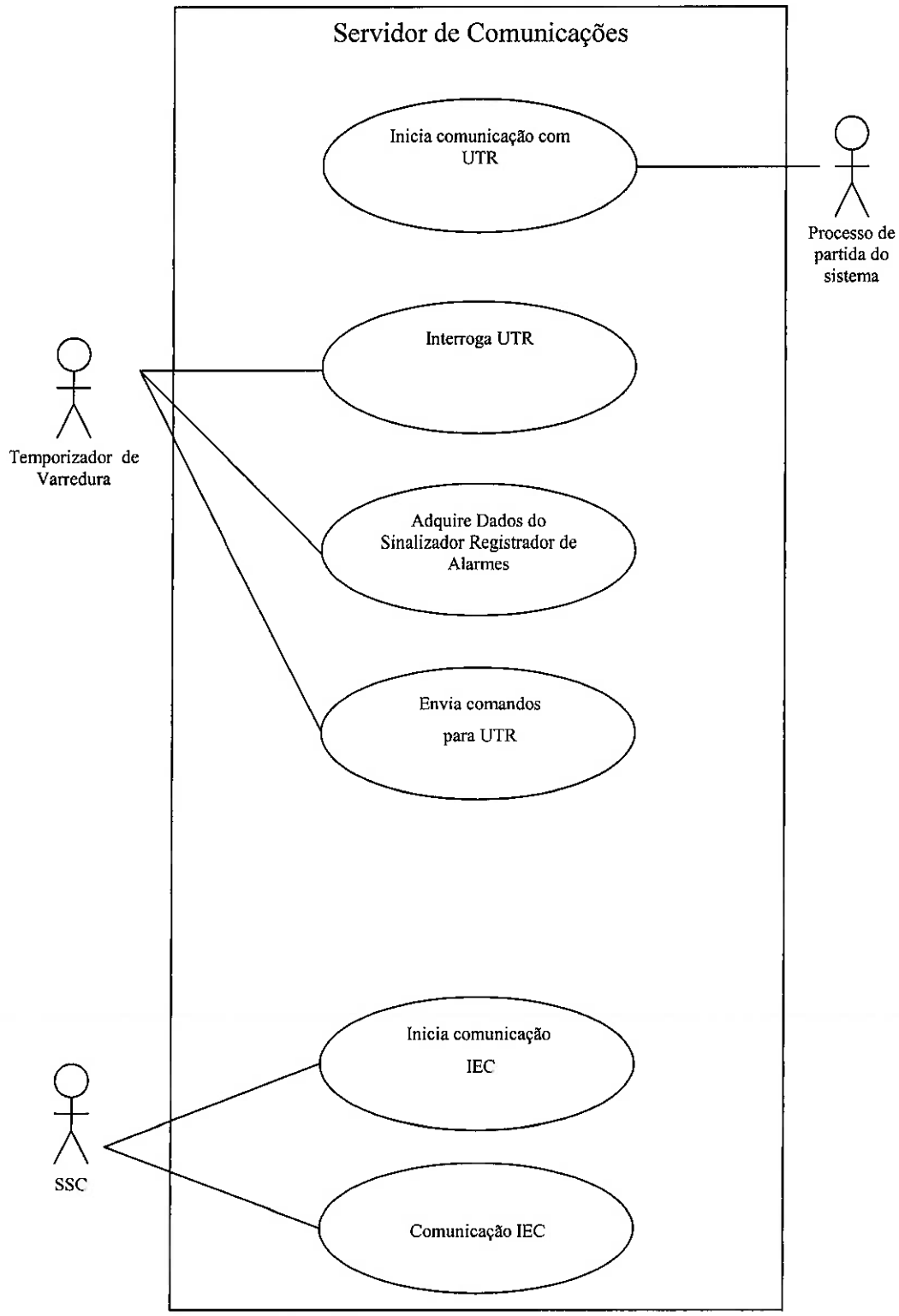

Figura 3-6 Diagrama de Contexto

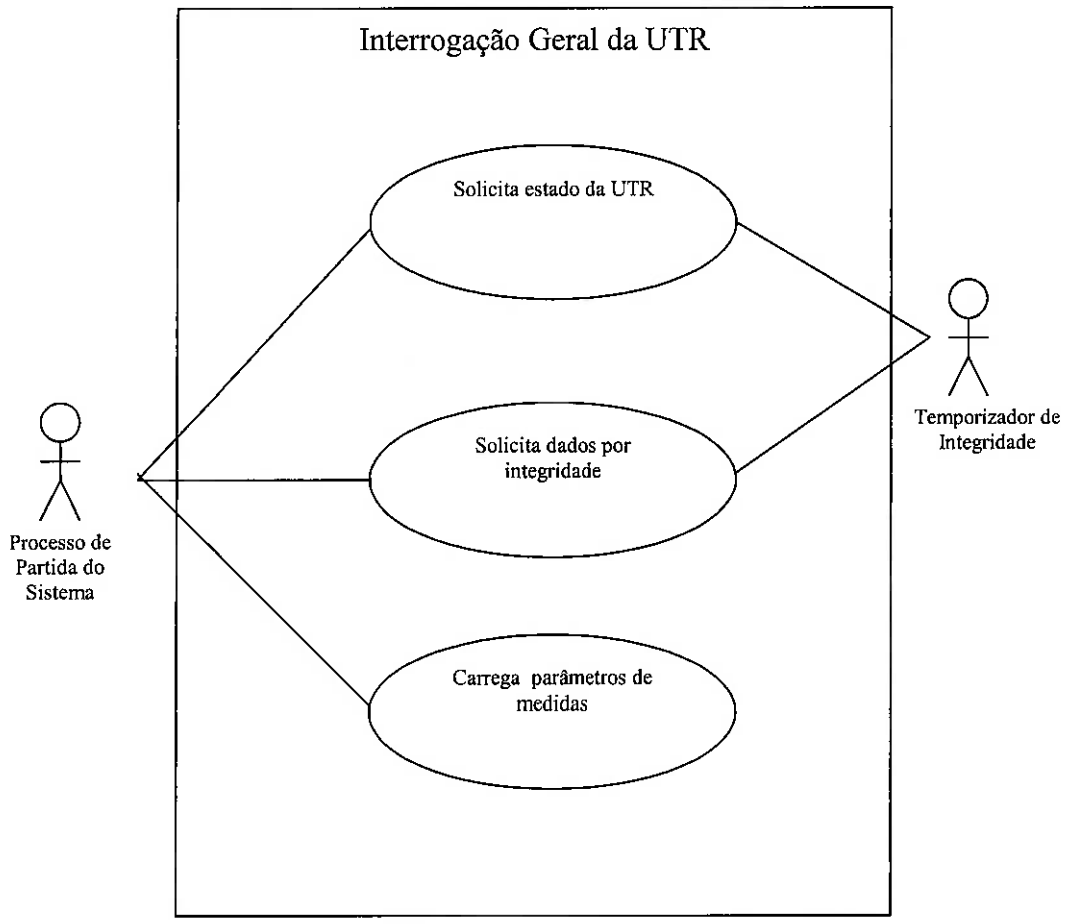

Figura 3-7 Interrogação Geral da UTR

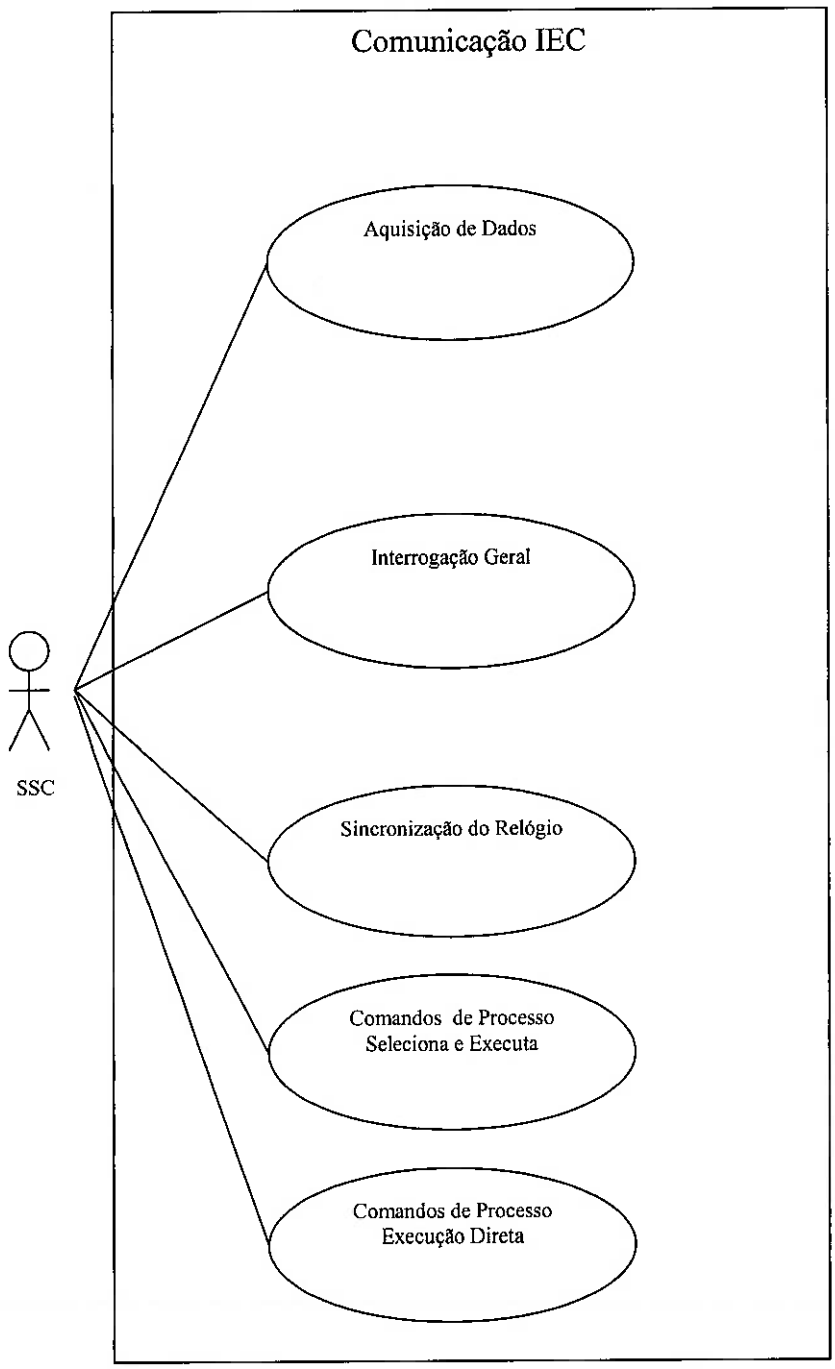

Figura 3-8 Comunicação IEC

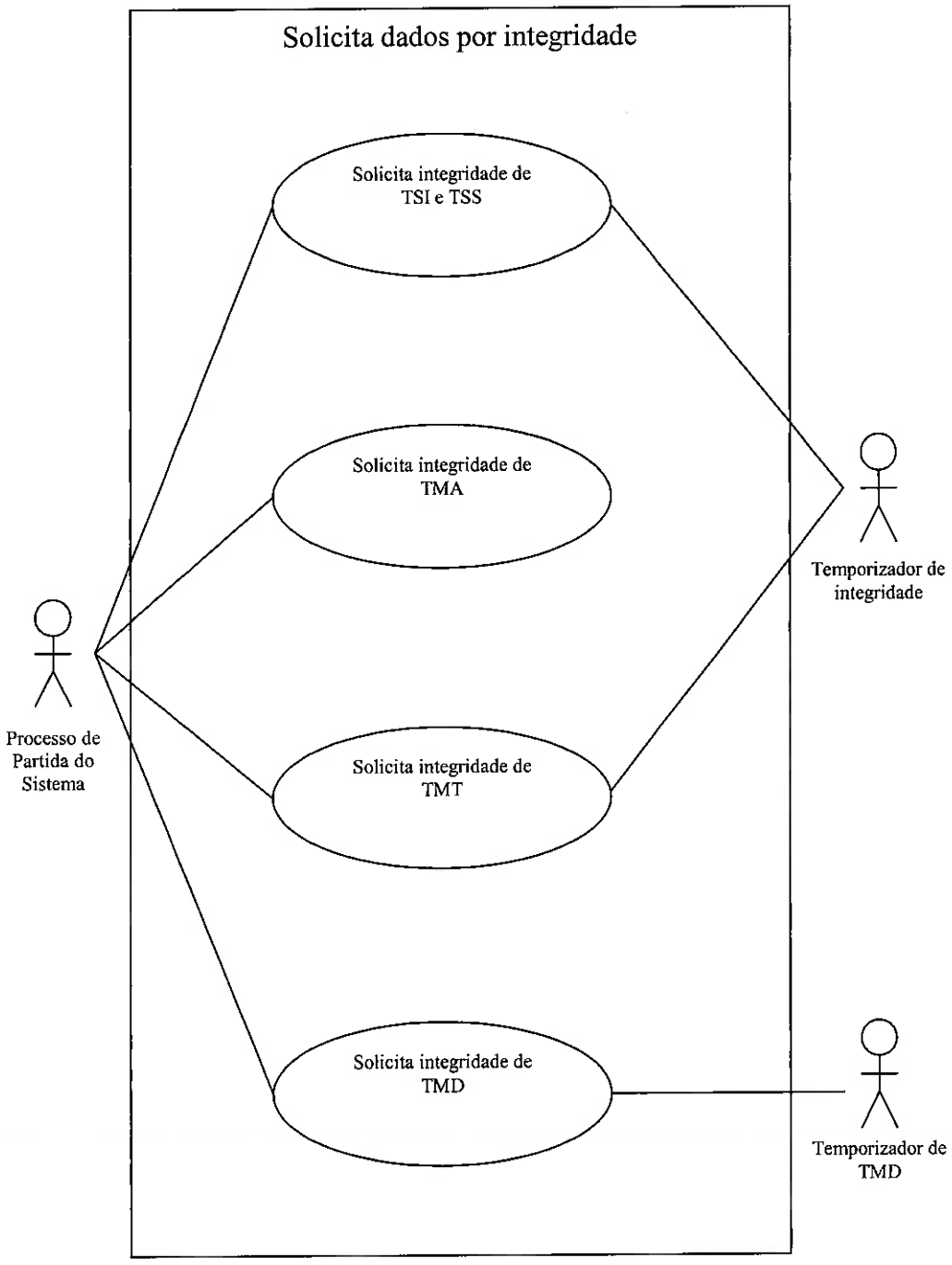

Figura 3-9 Solicita dados por integridade

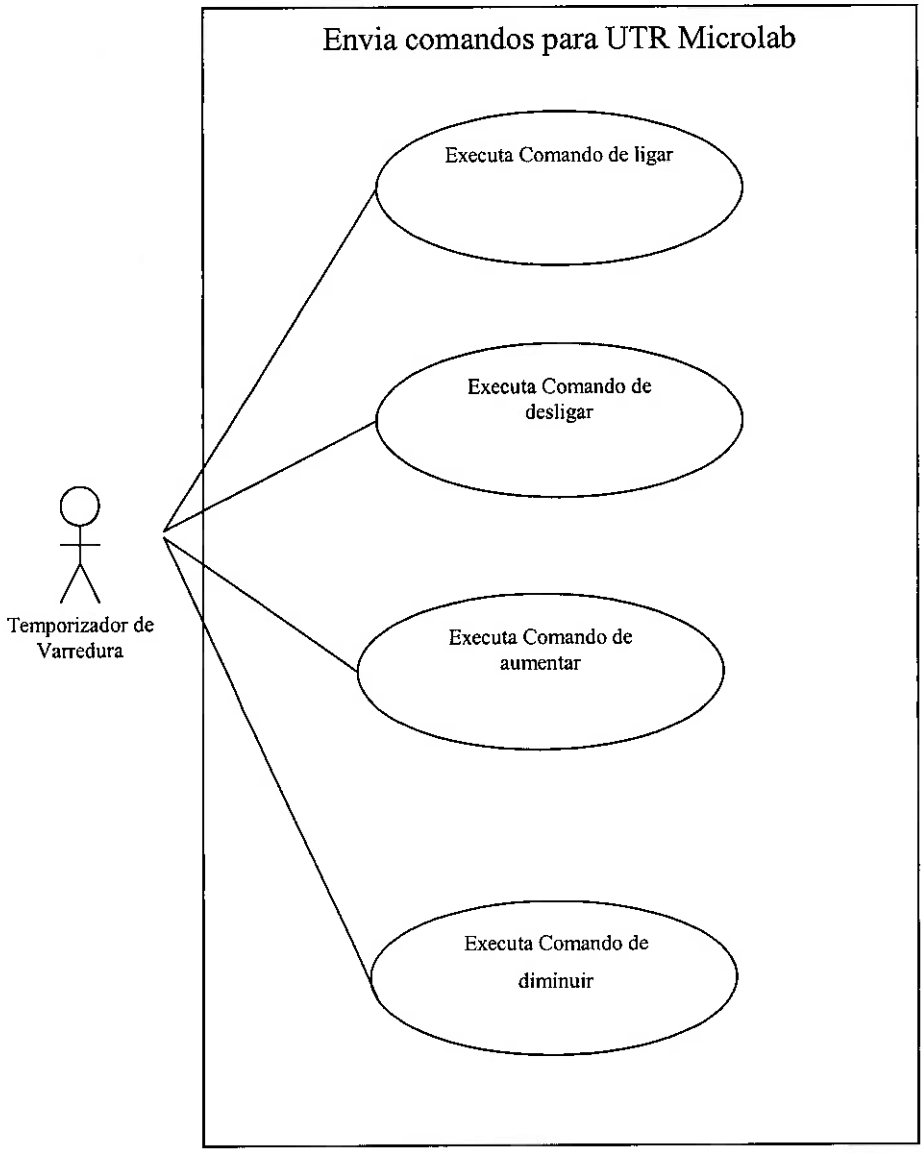

Figura 3-10 Envia comandos para UTR Microlab

3.5 DESCRIÇÃO DOS CASOS DE USO

Caso de Uso N° 1

Nome: Inicia Comunicação com UTR.

Descrição: Envia mensagem de solicitação de conexão com UTR Microlab.

Atores: Processo de Partida do Sistema.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente.

Curso normal dos eventos:

- O sistema envia uma mensagem SNRM.
- A UTR responde com uma mensagem UA.
- O sistema ativa os Temporizadores e executa o Caso de Uso Interrogação Geral da UTR.

Pós-condição: UTR conectada ao Servidor de Comunicações.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente tenta restabelecer a conexão.
- Se o Servidor de Comunicações recebe uma mensagem com falha, envia novamente uma mensagem SNRM.

- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação.
- Se o Servidor solicitar o envio de informações com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).
- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, enviará novamente uma mensagem SNRM.

Caso de Uso N° 2

Nome: Solicita Estado da UTR.

Descrição: Envia mensagens de solicitação de estado da UTR Microlab.

Atores: Processo de Partida do Sistema e Temporizador de integridade.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O sistema envia uma mensagem com pedido de estado da UTR.
- A UTR responde com mensagem de estado.
- O Sistema atualiza o Banco de Dados.

Pós-condição: Banco de Dados atualizado com informações de estado da UTR.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.

- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.

- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.

- Se o Servidor solicitar o envio de informações com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).

- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, o Caso de Uso Inicia Comunicação com UTR será executado novamente.

Caso de Uso N° 3

Nome: Solicita dados por integridade.

Descrição: Envia mensagens de solicitação de integridade para a UTR Microlab.

Atores: Processo de Partida do Sistema e Temporizador de Integridade.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O Sistema envia uma mensagem com pedido de integridade de TSI.
- A UTR informa o estado de todos os pontos de TSI.
- O Sistema atualiza todos os pontos de TSI do Banco de Dados.

- O Sistema envia uma mensagem com pedido de integridade de TSS.
- A UTR informa o estado de todos os pontos de TSS.
- O Sistema atualiza todos os pontos de TSS do Banco de Dados.
- O Sistema envia uma mensagem com pedido de integridade de TMA.
- A UTR informa o estado de todos os pontos de TMA.
- O Sistema atualiza todos os pontos de TMA do Banco de Dados.
- O Sistema envia uma mensagem com pedido de integridade de TMT.
- A UTR informa o estado de todos os pontos de TMT.
- O Sistema atualiza todos os pontos de TMT do Banco de Dados.

Pós-condição: Banco de Dados atualizado com todas as informações da UTR.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.
- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.
- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.
- Se o Servidor solicitar o envio de informações com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).
- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, o Caso de Uso Inicia Comunicação com UTR será executado novamente.

- Se um ou mais dos endereços iniciais dos conjuntos de pontos usados nas mensagens de solicitação de integridade não existirem, a mensagem enviada pela UTR será de mensagem com falha de formatação.

Caso de Uso N° 4

Nome: Solicita integridade de TSS.

Descrição: Envia mensagens de solicitação de integridade de TSS para a UTR Microlab.

Atores: Processo de Partida do Sistema e Temporizador de Integridade

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O sistema envia uma mensagem com pedido de integridade de TSS.
- A UTR informa o estado de todos os pontos de TSS.
- O Sistema atualiza a Base de Dados e envia mensagens RR até receber uma mensagem de fim de integridade de TSS.

Pós-condição: Atualização das informações de integridade de TSS da UTR.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.

- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.

- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.

- Se o Servidor solicitar o envio de informações com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).

- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, executa novamente o Caso de Uso Inicia Comunicação com UTR.

- Se os endereços de pontos usados na mensagem de solicitação de TSS forem de pontos de TSI, a mensagem enviada pela UTR será de integridade de TSI.

- Se os endereços de pontos usados na mensagem de solicitação de TSS forem de pontos de TSS e de pontos de TSI, a mensagem enviada pela UTR será uma composição de integridade de TSS seguida por integridade de TSI.

- Se um ou mais dos endereços iniciais dos conjuntos de pontos usados na mensagem de solicitação de integridade de TSS não existirem, a mensagem enviada pela UTR será de mensagem com falha de formatação.

Caso de Uso N° 5

Nome: Solicita integridade de TMA.

Descrição: Envia mensagens de solicitação de integridade de TMA para a UTR Microlab.

Atores: Processo de Partida do Sistema e Temporizador de Integridade.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O sistema envia uma mensagem com pedido de integridade de TMA.
- A UTR informa o estado de todos os pontos de TMA.
- O Sistema atualiza a Base de Dados e envia mensagens RR até receber uma mensagem de fim de integridade de TMA.

Pós-condição: Atualização das informações de integridade de TMA da UTR.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.
- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.
- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.
- Se o Servidor solicitar o envio de informações com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).
- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, executa novamente o Caso de Uso Inicia Comunicação com UTR.

- Se um ou mais dos endereços iniciais dos conjuntos de pontos usados na mensagem de solicitação de integridade de TMA não existirem, a mensagem enviada pela UTR será de mensagem com falha de formatação.

Caso de Uso N° 6

Nome: Solicita integridade de TMD.

Descrição: Envia mensagens de solicitação de integridade de TMD para a UTR Microlab.

Atores: Processo de Partida do Sistema e Temporizador de TMD.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O sistema envia uma mensagem com pedido de integridade de TMD.
- A UTR informa o estado de todos os pontos de TMD.
- O Sistema atualiza a Base de Dados e envia mensagens RR até receber uma mensagem de fim de integridade de TMD.

Pós-condição: Atualização das informações de integridade de TMD da UTR.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.

- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.

- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.

- Se o Servidor solicitar o envio de informações com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).

- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, executa novamente o Caso de Uso Inicia Comunicação com UTR.

- Se um ou mais dos endereços iniciais dos conjuntos de pontos usados na mensagem de solicitação de integridade de TMD não existirem, a mensagem enviada pela UTR será de mensagem com falha de formatação.

Caso de Uso N° 7

Nome: Solicita integridade de TMT.

Descrição: Envia mensagens de solicitação de integridade de TMT para a UTR Microlab.

Atores: Processo de Partida do Sistema e Temporizador de Integridade.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O sistema envia uma mensagem com pedido de integridade de TMT.

- A UTR informa o estado de todos os pontos de TMT.

- O Sistema atualiza a Base de Dados e envia mensagens RR até receber uma mensagem de fim de integridade

Pós-condição: Atualização das informações de integridade de TMT da UTR.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.
- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.
- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.
- Se o Servidor solicitar o envio de informações com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).
- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, executa novamente o Caso de Uso Inicia Comunicação com UTR.
- Se um ou mais dos endereços iniciais dos conjuntos de pontos usados na mensagem de solicitação de integridade de TMT não existirem, a mensagem enviada pela UTR será de mensagem com falha de formatação.

Caso de Uso N° 8

Nome: Carrega Parâmetros de Medidas.

Descrição: Envia mensagens de carga de parâmetros de medidas para a UTR Microlab.

Atores: Processo de Partida do Sistema.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O processo de partida envia uma mensagem de carga de parâmetros de medidas para a UTR.
- A UTR responde com uma mensagem de aceite da carga dos parâmetros.

Pós-condição: Atualização dos parâmetros de banda morta das medidas analógicas da UTR.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.
- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.

- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.

- Se o Servidor solicitar o envio de informações com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).

- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, executa novamente o Caso de Uso Inicia Comunicação com UTR.

- Se um ou mais dos endereços de pontos de medidas usados na mensagem de carga de parâmetros não existirem, a mensagem enviada pela UTR será de mensagem com falha de formatação.

Caso de Uso N° 9

Nome: Interroga UTR.

Descrição: Envia mensagens de varredura para a UTR Microlab.

Ator: Temporizador de varredura.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O Sistema envia uma mensagem RR de varredura.

- A UTR responde com uma mensagem I de informação ou com mensagem RR se não tiver informações para enviar.

- Se a UTR responder com mensagem I de informação, o sistema atualiza a Base de Dados com as informações recebidas.

Pós-condição: Atualização das informações de estado e validade para os pontos digitais, de valor e validade para as medições analógicas e digitais e de estado da UTR.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.
- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.
- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.
- Se o Servidor enviar uma mensagem RR com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).
- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, executa novamente o Caso de Uso Inicia Comunicação com UTR.

Caso de Uso N° 10

Nome: Executa Comando Ligar.

Descrição: Envia mensagens de Seleção e Execução de Comandos Ligar para a UTR Microlab.

Ator: Temporizador de varredura.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O Servidor de Comunicações verifica que há um comando de ligar pendente e envia uma mensagem de Seleção do ponto para Comando de Ligar para a UTR.
- A UTR responde com uma mensagem de aceite da Seleção.
- O Servidor de Comunicações envia uma mensagem, de Execução de Comando Ligar no ponto selecionado, para a UTR.
- A UTR responde com uma mensagem de aceite da Execução do Comando no ponto selecionado.
- O Servidor de Comunicações atualiza a Base de Dados, removendo o comando da relação de comandos pendentes.

Pós-condição: Execução do Comando Ligar e atualização das informações de estado e validade para o Comando Ligar o ponto selecionado.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.
- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.

- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.

- Se o Servidor de Comunicações enviar uma mensagem de Seleção ou de Execução de comando com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).

- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, executa novamente o Caso de Uso Inicia Comunicação com UTR.

- Se houverem falhas internas à UTR que impossibilitem a execução do Comando Ligar, a UTR envia mensagem de falhas para o Servidor de Comunicações.

Caso de Uso N° 11

Nome: Executa Comando Desligar.

Descrição: Envia mensagens de Seleção e Execução de Comandos Desligar para a UTR.

Ator: Temporizador de Varredura.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O Servidor de Comunicações verifica que há um comando pendente na Base de Dados e envia uma mensagem de Seleção do ponto para Comando de Desligar para a UTR.

- A UTR responde com uma mensagem de aceite da Seleção.
- O Servidor de Comunicações envia uma mensagem, de Execução de Comando Desligar no ponto selecionado, para a UTR.
- A UTR responde com uma mensagem de aceite da Execução do Comando no ponto selecionado.
- O Servidor de Comunicações atualiza a Base de Dados, removendo o comando da relação de comandos pendentes.

Pós-condição: Execução do Comando Desligar e atualização das informações de estado e validade para o Comando Desligar o ponto selecionado.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.
- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.
- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.
- Se o Servidor solicitar o envio de informações com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).

- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, o Caso de Uso Inicia Comunicação com UTR será executado novamente.

- Se houverem falhas internas à UTR que impossibilitem a execução do Comando Desligar, a UTR envia mensagem de falhas para o Servidor de Comunicações.

Caso de Uso N° 12

Nome: Executa Comando Aumentar.

Descrição: Envia mensagens de Execução de Comandos Aumentar para a UTR.

Ator: Temporizador de varredura.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O Servidor de Comunicações verifica que há um comando pendente e envia uma mensagem de Execução de Comando Aumentar no ponto desejado, para a UTR.

- A UTR responde com uma mensagem de aceite da Execução do Comando no ponto desejado.

- O Servidor de Comunicações atualiza a Base de Dados, removendo o comando da relação de comandos pendentes.

Pós-condição: Execução do Comando Aumentar e atualização das informações de estado e validade para o Comando Aumentar o ponto desejado.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.

- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.

- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.

- Se o Servidor solicitar o envio de informações com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).

- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, o Caso de Uso Inicia Comunicação com UTR será executado novamente.

- Se houverem falhas internas à UTR que impossibilitem a execução do Comando Aumentar, a UTR envia mensagem de falhas para o Servidor de Comunicações.

Caso de Uso N° 13

Nome: Executa Comando Diminuir.

Descrição: Envia mensagens de Execução de Comandos Diminuir para a UTR.

Ator: Temporizador de varredura.

Pré-condição: Toda infra-estrutura computacional e de comunicação com a UTR está funcionando corretamente e UTR já conectada (comunicação já iniciada).

Curso normal dos eventos:

- O Servidor de Comunicações verifica que há um comando pendente e envia uma mensagem, de Execução de Comando Diminuir no ponto desejado, para a UTR.
- A UTR responde com uma mensagem de aceite da Execução do Comando no ponto desejado.
- O Servidor de Comunicações atualiza a Base de Dados, removendo o comando da relação de comandos pendentes.

Pós-condição: Execução do Comando Diminuir e atualização das informações de estado e validade para o Comando Diminuir o ponto desejado.

Lista de casos alternativos:

- Se a UTR não responde, o Servidor de Comunicações coloca a unidade em falha, temporiza e periodicamente executa o Caso de Uso Inicia Comunicação com UTR.
- Se o Servidor de Comunicações recebe uma mensagem com falha, o mesmo solicita a retransmissão. Se o número de falhas consecutivas exceder o limite, a conexão é desfeita.
- Se a UTR recebe uma mensagem com falha, a mesma não responde, permanecendo silenciosa e não executando qualquer ação, resultando em time-out no Servidor de Comunicações.

- Se o Servidor solicitar o envio de informações com a conexão desfeita, a UTR responde com mensagem de modo desconectado (DM).

- Se a qualquer instante, o Servidor de Comunicações receber uma mensagem DM da UTR Microlab, o Caso de Uso Inicia Comunicação com UTR será executado novamente.

- Se houverem falhas internas à UTR que impossibilitem a execução do Comando Diminuir, a UTR envia mensagem de falhas para o Servidor de Comunicações.

Caso de Uso N° 14

Nome: Adquire dados do Sinalizador Registrador de Alarmes.

Descrição: Envia mensagens no protocolo MODBUS para aquisição de dados de seqüência de eventos armazenados no Sinalizador Registrador de Alarmes (SRA).

Ator: Temporizador de varredura.

Pré-condição: Toda infra-estrutura computacional e de comunicação com o Sinalizador Registrador de Alarmes está funcionando corretamente.

Curso normal dos eventos:

- O Servidor de Comunicações envia uma mensagem de Leitura de Registro com o endereço do registro indicador de disponibilidade de informações de seqüência de eventos.

- O SRA responde com o valor do registro solicitado.

- O Servidor de Comunicações envia uma mensagem de Leitura de Fila com o endereço da fila que contêm a seqüência de eventos.

- O SRA envia as informações de seqüência de eventos.

- O Servidor de Comunicações atualiza a Base de Dados e envia uma mensagem de Escrita em Registro com o endereço do registro de quitação de mensagem.
- O SRA responde com o eco da última mensagem.
- O SRA aguarda o tempo equivalente a quatro caracteres e então libera a fila de eventos e apaga o registro indicador de disponibilidade de informações de seqüência de eventos.

Pós-condição: Base de Dados atualizada com seqüência de eventos e fila de eventos do SRA liberada.

Lista de casos alternativos:

- O SRA não responde se receber uma mensagem com erro de comunicação (paridade ou CRC).
- Se o Servidor de Comunicação não receber resposta em um tempo pré-definido, o Caso de Uso é encerrado.
- Se o SRA recebe uma mensagem sem erro de comunicação, mas não consegue trata-la, irá responder com uma mensagem de exceção.
- Se o Servidor de Comunicações recebe uma mensagem com erro de comunicação (paridade ou CRC), a mensagem de interrogação será repetida.

Caso de Uso N° 15

Nome: Inicia comunicação IEC.

Descrição: Estabelece enlace de comunicação entre o SSC e o Servidor de Comunicações.

Ator: SSC.

Pré-condição: Toda infra-estrutura computacional e de comunicação com o SSC está funcionando corretamente.

Curso normal dos eventos:

- O SSC envia uma mensagem de requisição de estado do enlace.
- O Servidor de Comunicações responde com uma mensagem de estado do enlace.
- O SSC envia uma mensagem de reiniciação do enlace remoto.
- O Servidor de Comunicações responde com uma mensagem de reconhecimento.
- O SSC envia uma mensagem de requisição de dados de classe 2.
- O Servidor de Comunicações envia uma mensagem com a informação M_EI_NA_1 (fim de inicialização).

Pós-condição: Enlace entre SSC e Servidor de Comunicações inicializado.

Lista de casos alternativos:

- O Servidor de Comunicações não responderá a mensagens recebidas com falha.

- Se o Servidor de Comunicações receber uma requisição de dados de classe 2 antes de terminar o processo de iniciação, responderá com uma mensagem de "enlace ocupado" ou de "dado requisitado não disponível", dependendo do caso.

Caso de Uso N° 16

Nome: Aquisição de Dados (IEC).

Descrição: Envia os valores das variáveis do processo que sofreram alteração desde a última interrogação.

Ator: SSC.

Pré-condição: Toda infra-estrutura computacional e de comunicação com o SSC está funcionando corretamente.

Curso normal dos eventos:

- O SSC envia uma mensagem de requisição de dados de classe 2 ou de classe 1, dependendo da resposta da requisição anterior.
- O Servidor de Comunicações envia uma mensagem de resposta contendo os valores das variáveis do processo de classe 2 ou de classe 1 dependendo da requisição e informando se existem dados de classe 1 aguardando envio.
- O Servidor de Comunicações atualiza a Base de Dados removendo da fila os eventos reportados e marcando como "NÃO ALTERADOS" os dados enviados.

Pós-condição: Valores das variáveis do processo enviadas ao Sistema de Supervisão e Controle.

Lista de casos alternativos:

- A primeira mensagem deverá requisitar dados de classe 2.
- Se não houver dados das classes 1 e 2 a serem enviados, o Servidor de Comunicações responderá com uma mensagem de "dado requisitado não disponível".
- Se houver apenas dados de classe 1 disponíveis, um mensagem de requisição de dados de classe 2 será respondida com uma mensagem "dado requisitado não disponível" informando que existem dados de classe 1 a serem enviados

Caso de Uso N° 17

Nome: Interrogação geral.

Descrição: Responde a uma mensagem de interrogação geral enviando os valores de todas as variáveis do processo.

Ator: SSC.

Pré-condição: Toda infra-estrutura computacional e de comunicação com o SSC está funcionando corretamente.

Curso normal dos eventos:

- O SSC envia uma mensagem com a informação C_IC_NA_1 indicando "ativação" como causa da transmissão.
- O Servidor de Comunicações responde com uma mensagem de reconhecimento.
- O SSC envia uma mensagem de requisição de dados de classe 2.

- O Servidor de Comunicações responde com uma mensagem com a informação C_IC_NA_1 indicando "confirmação de ativação" como causa da transmissão.
- O SSC executa o Caso de Uso "Aquisição de Dados" até receber uma mensagem com a informação C_IC_NA_1 indicando "término de ativação" como causa da transmissão.
- O Servidor de Comunicações atualiza a Base de Dados removendo da fila os eventos reportados e marcando como "NÃO ALTERADOS" todos os dados enviados.

Pós-condição: Valores de todas as variáveis do processo enviadas ao Sistema de Supervisão e Controle.

Lista de casos alternativos:

- O Servidor de Comunicações não responderá a mensagens recebidas com falha.

Caso de Uso N° 18

Nome: Sincronização do Relógio.

Descrição: Sincroniza relógio do Servidor de Comunicações com relógio do SSC.

Ator: SSC.

Pré-condição: Toda infra-estrutura computacional e de comunicação com o SSC está funcionando corretamente.

Curso normal dos eventos:

- O SSC envia uma mensagem com a informação C_CD_NA_1, indicando "ativação" como causa da transmissão.
- O Servidor de Comunicações responde com uma mensagem de reconhecimento.
- O SSC envia uma mensagem de requisição de dados de classe 2.
- O Servidor de Comunicações responde com a mensagem C_CD_NA_1, indicando "confirmação de ativação" como causa da transmissão.
- O SSC calcula o atraso de transmissão e envia uma mensagem com a informação C_CD_NA_1, indicando "espontâneo" como causa da transmissão.
- O Servidor de Comunicações envia uma mensagem de reconhecimento.
- O SSC envia uma mensagem com a informação C_CS_NA_1 com o horário de transmissão corrigido pelo atraso de transmissão, indicando "ativação" como causa da transmissão.
- O Servidor de Comunicações sincroniza seu relógio e responde com uma mensagem de reconhecimento.
- O SSC envia uma mensagem de requisição de dados de classe 2.
- O Servidor de Comunicações responde com a mensagem C_CS_NA_1, indicando "confirmação de ativação" como causa da transmissão.

Pós-condição: Relógio do Servidor de Comunicações sincronizado com o SSC.

Lista de casos alternativos:

- O Servidor de Comunicações não responderá a mensagens recebidas com falha.

Caso de Uso N° 19

Nome: Comandos de processo seleciona e executa.

Descrição: Trata mensagens de comandos de processo do tipo seleciona antes de executar.

Ator: SSC.

Pré-condição: Toda infra-estrutura computacional e de comunicação com o SSC está funcionando corretamente.

Curso normal dos eventos:

- O SSC envia uma mensagem com a informação C_SC_NA_1 para seleção do ponto, indicando "ativação" como causa da transmissão.
- O Servidor de Comunicações indica o ponto como "selecionado" na Base de Dados, inicia um temporizador e responde com uma mensagem de reconhecimento.
- O SSC envia uma mensagem de requisição de dados de classe 2.
- O Servidor de Comunicações responde com uma mensagem com a informação C_SC_NA_1 indicando "confirmação de ativação" como causa da transmissão.
- O SSC envia uma mensagem com a informação C_SC_NA_1 para execução do comando no ponto selecionado, indicando "ativação" como causa da transmissão.

- O Servidor de Comunicações indica o ponto como "execução ativada" na Base de Dados, cancela a temporização de seleção, e responde com uma mensagem de reconhecimento.
- O SSC envia uma mensagem de requisição de dados de classe 2.
- O Servidor de Comunicações responde com uma mensagem com a informação C_SC_NA_1 indicando "confirmação de ativação" como causa da transmissão.

Pós-condição: Base de Dados do Servidor de Comunicações atualizada com o ponto a ser comandado.

Lista de casos alternativos:

- O Servidor de Comunicações não responderá a mensagens recebidas com falha.
- Se o comando de execução exceder o limite de tempo pré-determinado, o Servidor de Comunicações cancela a seleção do ponto.
- Se o ponto a ser comandado não existir, o Servidor de Comunicações responde com uma mensagem de reconhecimento negativo.
- Se um outro ponto for selecionado antes da execução do comando anterior, o Servidor de Comunicações cancela a seleção anterior.

Caso de Uso N° 20

Nome: Comandos de processo execução direta.

Descrição: Trata mensagens de comandos de processo do tipo execução direta.

Ator: SSC.

Pré-condição: Toda infra-estrutura computacional e de comunicação com o SSC está funcionando corretamente.

Curso normal dos eventos:

- O SSC envia uma mensagem com a informação C_RC_NA_1 para execução de comando, indicando "ativação" como causa da transmissão.
- O Servidor de Comunicações indica o ponto como "execução ativada" na Base de Dados e responde com uma mensagem de reconhecimento.
- O SSC envia uma mensagem de requisição de dados de classe 2.
- O Servidor de Comunicações responde com uma mensagem com a informação C_RC_NA_1 indicando "confirmação de ativação" como causa da transmissão.

Pós-condição: Base de Dados do Servidor de Comunicações atualizada com o ponto a ser comandado.

Lista de casos alternativos:

- O Servidor de Comunicações não responderá a mensagens recebidas com falha.
- Se o ponto a ser comandado não existir, o Servidor de Comunicações responde com uma mensagem de reconhecimento negativo.

3.6 IDENTIFICAÇÃO DAS CLASSES

A identificação das classes e objetos do Servidor de Comunicações foi feita em conformidade com os critérios recomendados por Pressman (2001), com o exame da narrativa do sistema a ser construído em busca de nomes. Uma vez identificados os nomes deve-se então construir uma tabela identificando os objetos em potencial, baseando-se nos seguintes critérios:

- **Entidades externas** (p. ex., outros sistemas, dispositivos, pessoas) que produzem ou consumem informação a ser usada por um sistema baseado em computador.
- **Coisas** (p. ex., relatórios, interfaces, cartas, sinais) que são parte do domínio de informação para o problema.
- **Ocorrências ou eventos** (p. ex. a transferência de uma propriedade ou o término de uma série de movimentos de um robot) que ocorrem dentro do contexto da operação do sistema.
- **Funções** (p. ex. gerente, engenheiro, vendedor) executadas por pessoas que interagem com o sistema.
- **Unidades organizacionais** (p. ex. divisão, grupo, equipe) que são relevantes para uma aplicação.
- **Lugares** (p.ex. chão de fábrica, armazéns) que estabelecem o contexto do problema e do conjunto de funções do sistema.
- **Estruturas** (p. ex. sensores, veículos, computadores) que definem uma classe de objetos ou no caso extremo, são relacionadas com classes de objetos.

Extraindo os nomes da narrativa do problema, temos a seguinte relação de objetos em potencial:

Objeto/Classe Potencial	Classificação Geral
Configurador	Estrutura
Unidades Terminais Remotas	Entidade externa
Sinalizador Registrador de Alarmes	Entidade externa
Bases de Dados	Estrutura
Porta serial	Parte do sistema
Sistemas Supervisórios	Entidade externa
Protocolo	Regras de comunicação

Coad e Yourdon (1991) apud Pressman (2001) sugerem seis características que devem ser usadas conforme o analista considera cada objeto em potencial para inclusão no modelo de análise:

1. Retenção de Informação: O objeto em potencial será útil durante a análise apenas se informação a seu respeito precisar ser relembrada para que o sistema funcione.

2. Fornecimento de serviços: O objeto em potencial precisa ter um conjunto identificável de operações que possam mudar o valor de seus atributos de alguma forma.

3. Atributos múltiplos: Objetos com um único atributo provavelmente poderão ser melhor representados como atributo de um outro objeto.

4. Atributos comuns: Um conjunto de atributos pode ser atribuído ao objeto em potencial e estes atributos são aplicáveis a todas as ocorrências do objeto.

5. Operações comuns: Um conjunto de operações pode ser atribuído ao objeto em potencial e estes atributos são aplicáveis a todas as ocorrências do objeto.

6. Requisitos essenciais: Entidades externas que aparecem no espaço do problema e produzem ou consomem informação essencial para operação de qualquer solução para o sistema, serão quase sempre definidas como objetos no modelo de análise.

Aplicando-se estes critérios aos objetos em potencial listados pela análise da narrativa do problema, obtemos o seguinte resultado:

Objeto/Classe Potencial	Característica Aplicável
Configurador	Aceito: todas são aplicáveis
Unidades Terminais Remotas	Aceito: todas são aplicáveis
Sinalizador Registrador de Alarmes	Aceito: todas são aplicáveis
Bases de Dados	Aceito: todas são aplicáveis
Porta serial	Rejeitado: 1 e 2 não se aplicam
Sistemas Supervisórios	Aceito: todas são aplicáveis
Protocolo	Rejeitado: 1 e 2 não se aplicam

Deve-se observar que a lista acima não pode ser considerada como definitiva. Objetos adicionais podem ter que ser adicionados para completar o modelo; alguns objetos poderão ter que ser divididos em dois ou mais objetos que colaboram entre si e alguns dos objetos rejeitados podem ser tornar atributos dos objetos aceitos.

3.7 DIAGRAMAS DE CLASSES

Conforme Booch (1999), Diagramas de Classes são utilizados para representar os aspectos estáticos de um sistema. Estes diagramas permitem visualizar as classes e os relacionamentos entre elas. Em UML uma classe é representada como um retângulo vertical, usualmente divido em 3 partes. Cada uma destas partes contém as seguintes informações, de cima para baixo: nome, atributos e operações. Opcionalmente, o retângulo poderá conter uma quarta parte, utilizada para definir as responsabilidades da classe.

Um atributo é a propriedade de uma classe que descreve as informações que a classe pode armazenar e representam propriedades compartilhadas por todos os objetos da classe.

Uma operação é a implementação de um serviço que pode ser prestado por qualquer objeto da classe para outros objetos ou para si mesmo.

Os diagramas a seguir são a representação em UML das classes que irão constituir o Servidor de Comunicações. Para evitar sobrecarregar os diagramas, alguns atributos e operações comuns a qualquer classe como, por exemplo, construtores, destrutores foram omitidos.

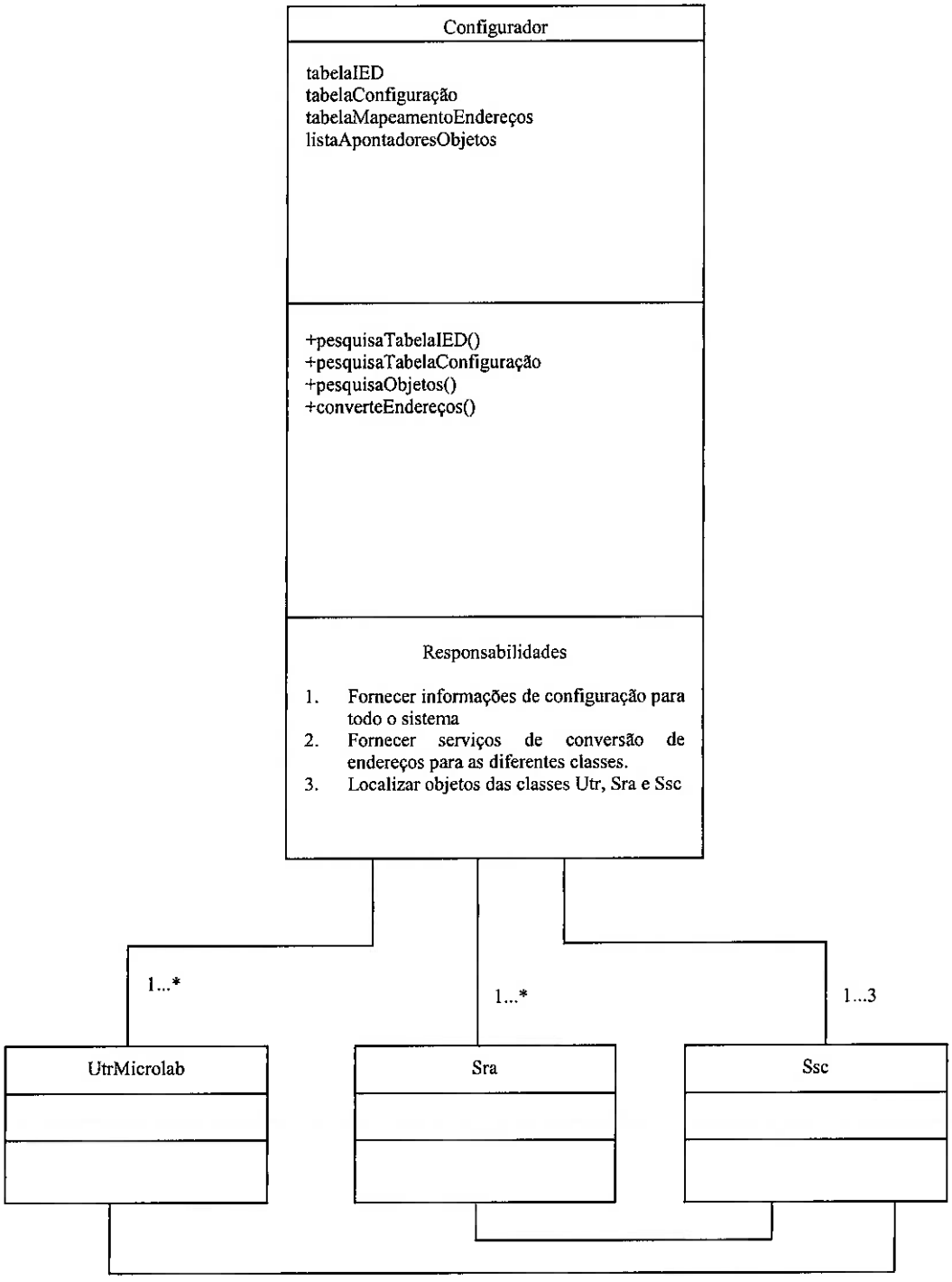

Figura 3-11 Classe Configurador

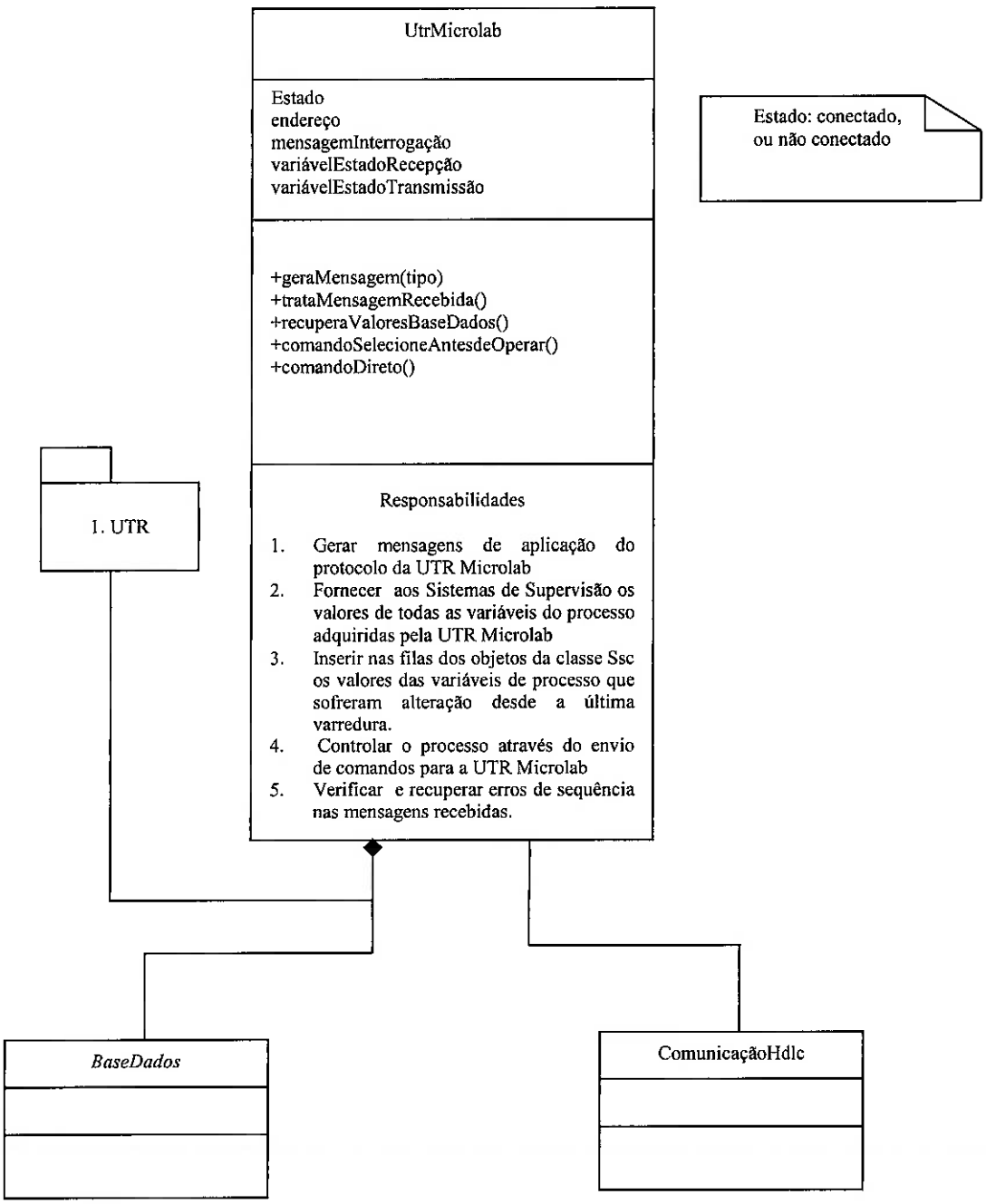

Figura 3-12 Classe UTRMicrolab

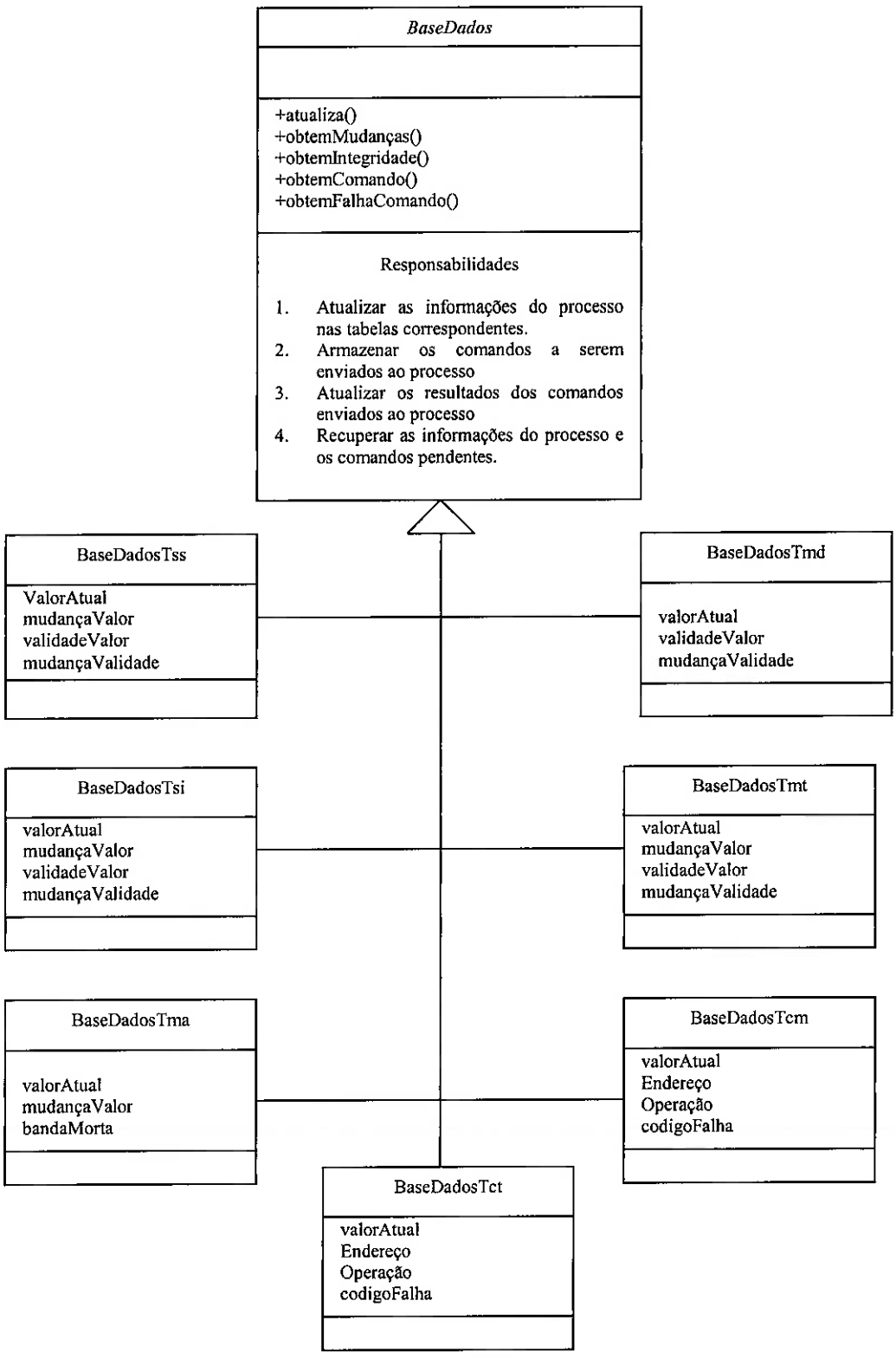

Figura 3-13 Classe BaseDados

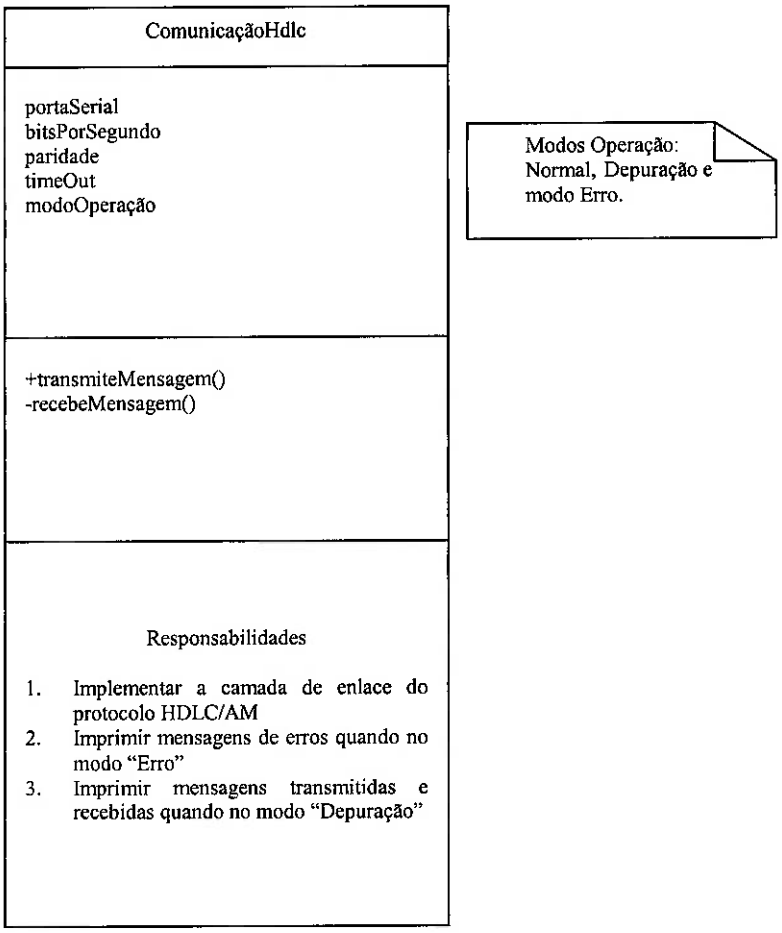

Figura 3-14 Classe ComunicaçãoHdlc

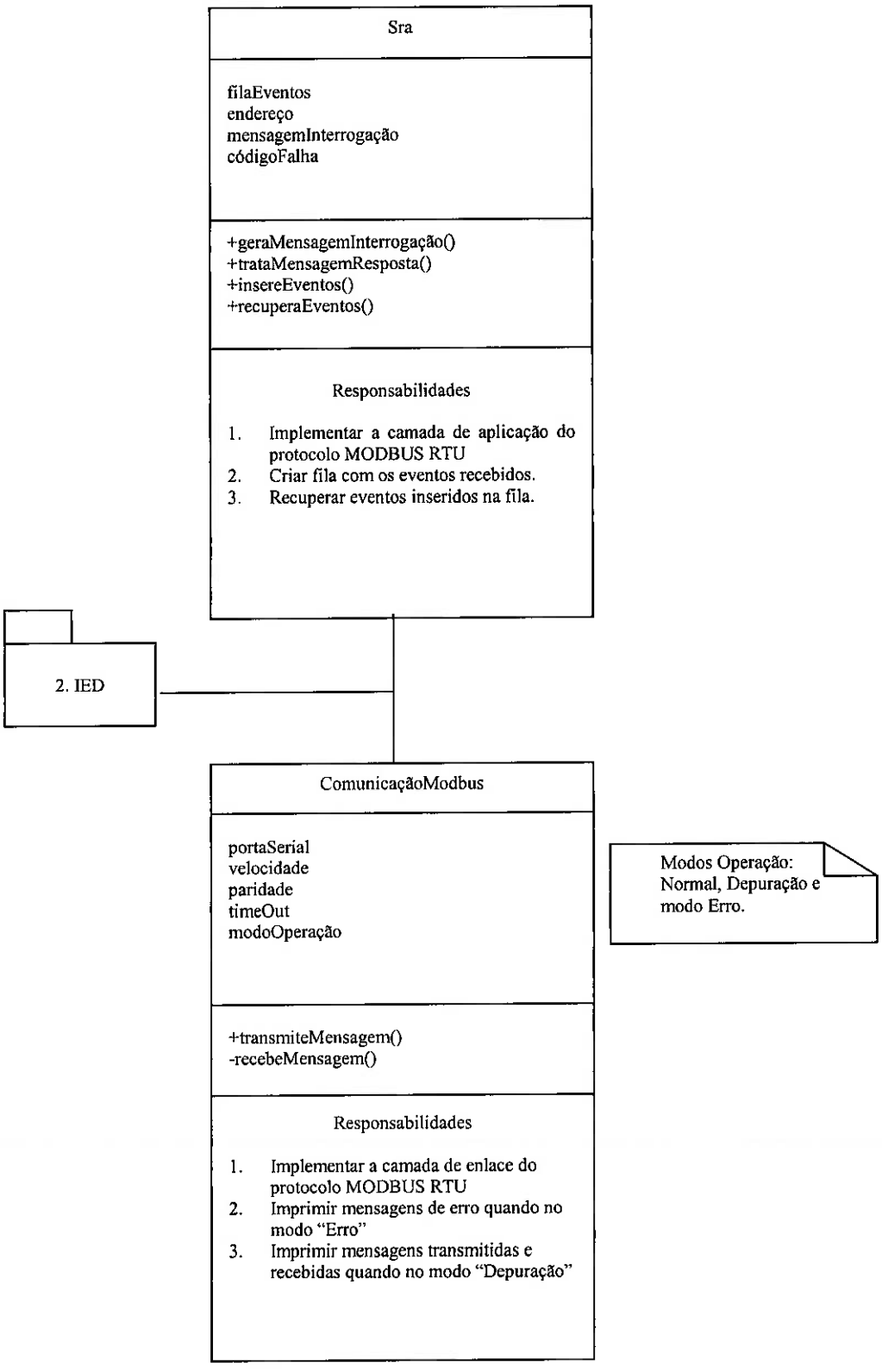

Figura 3-15 Classes Sra e ComunicaçãoModbus

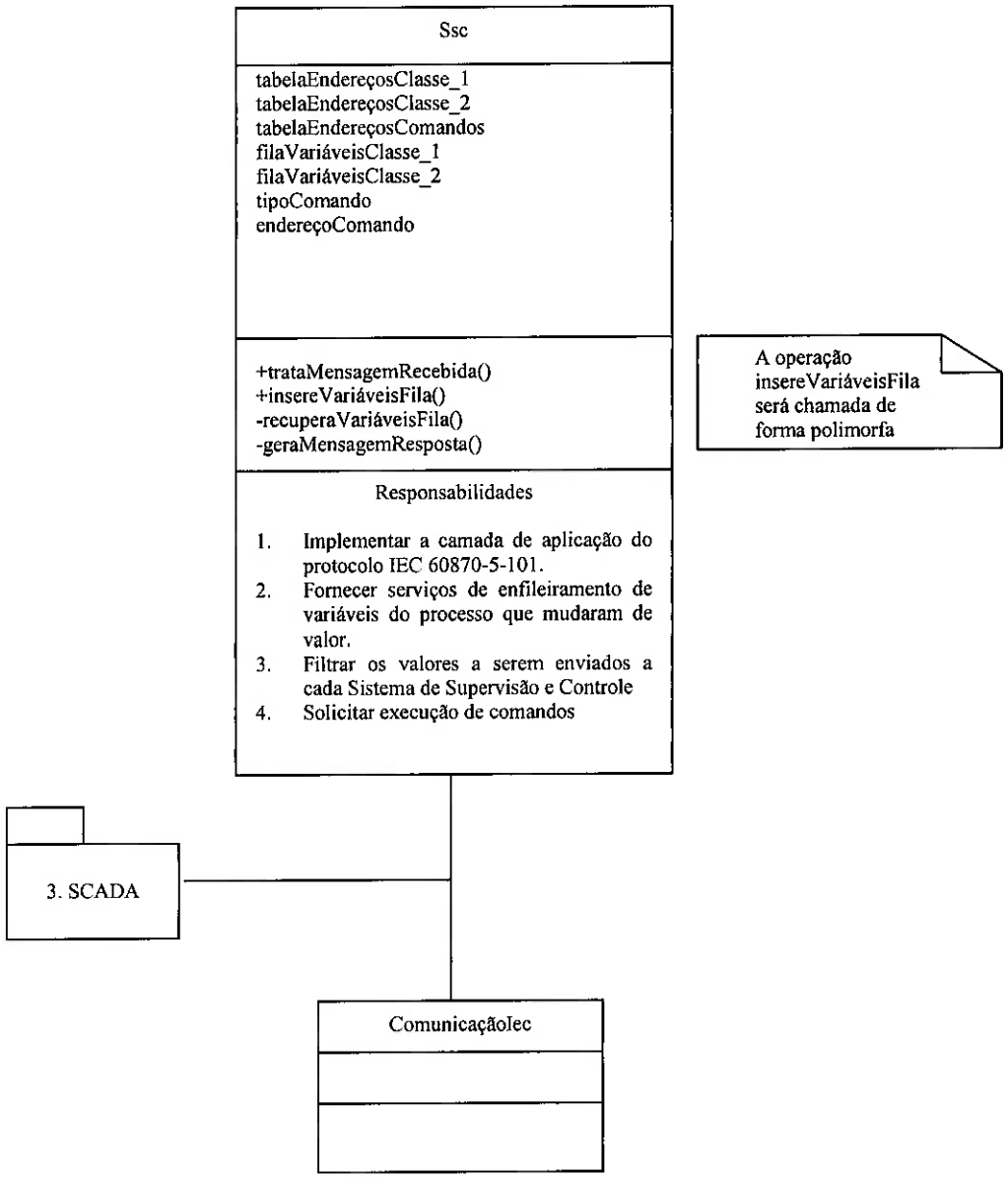

Figura 3-16 Classe Ssc

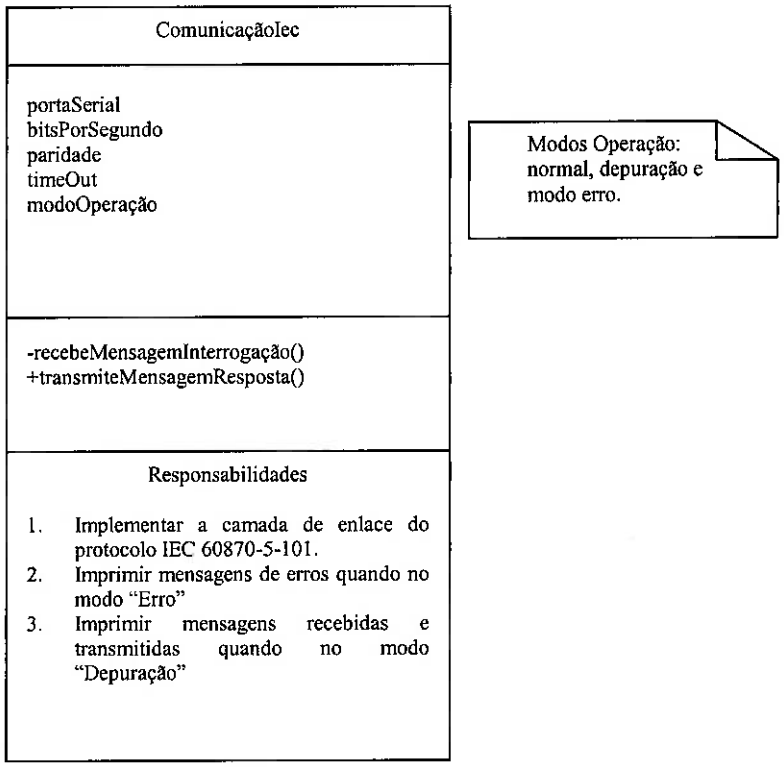

Figura 3-17 Classe ComunicaçãoIec

3.8 DIAGRAMAS DE SEQÜÊNCIA

Conforme Booch (1999), Diagramas de Seqüência procuram capturar o comportamento dinâmico de um sistema mostrando as trocas de mensagens entres os objetos para diversos cenários. É uma ferramenta muito importante na fase de análise para ajudar a identificar atributos e operações não definidos durante a construção dos Diagramas de Classes.

Os diagramas a seguir mostram os Diagramas de Seqüência para os seguintes cenários:

- Inicia comunicação com UTR Microlab, figura 3-18
- Solicita estado da UTR Microlab, figura 3-19
- Solicita dados por integridade da UTR Microlab, figura 3-20
- Adquire dados do Sinalizador Registrador de Alarmes, figura 3-21
- Responde interrogação de Sistemas de Supervisão e Controle, figura 3-22

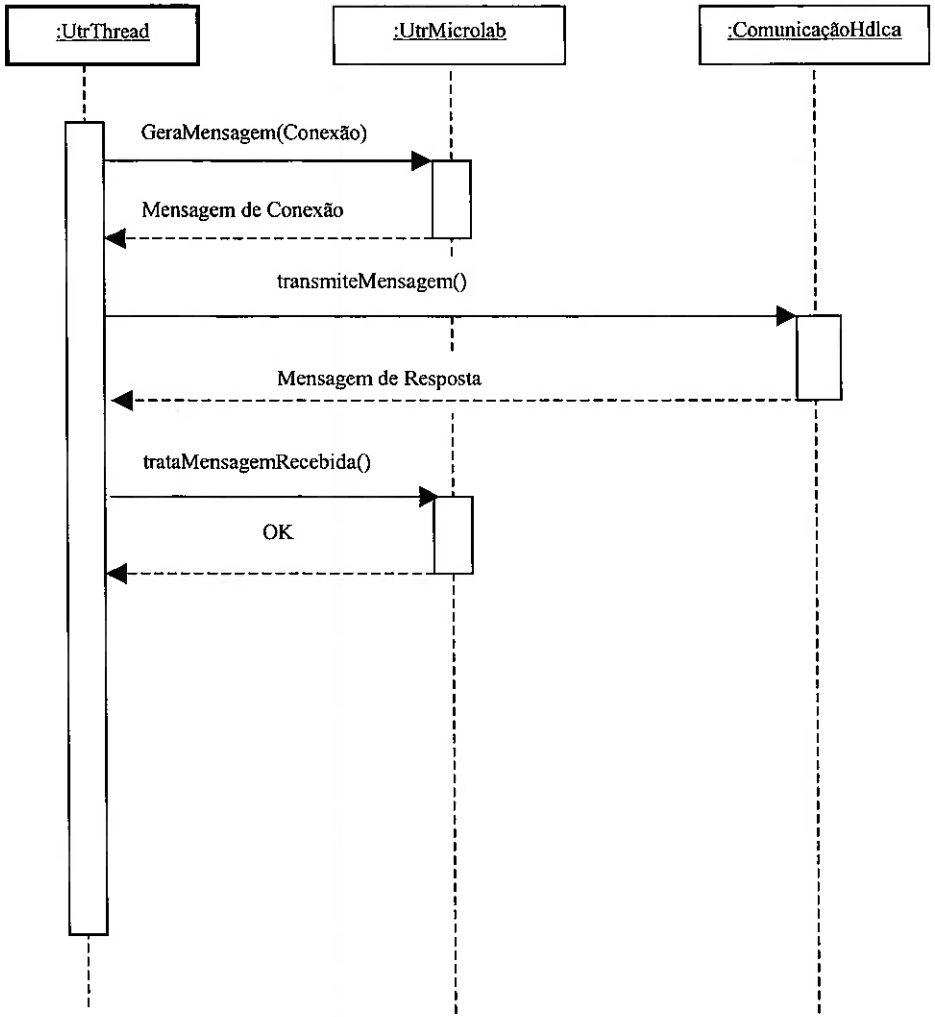

Figura 3-18 Inicia Comunicação com UTR Microlab

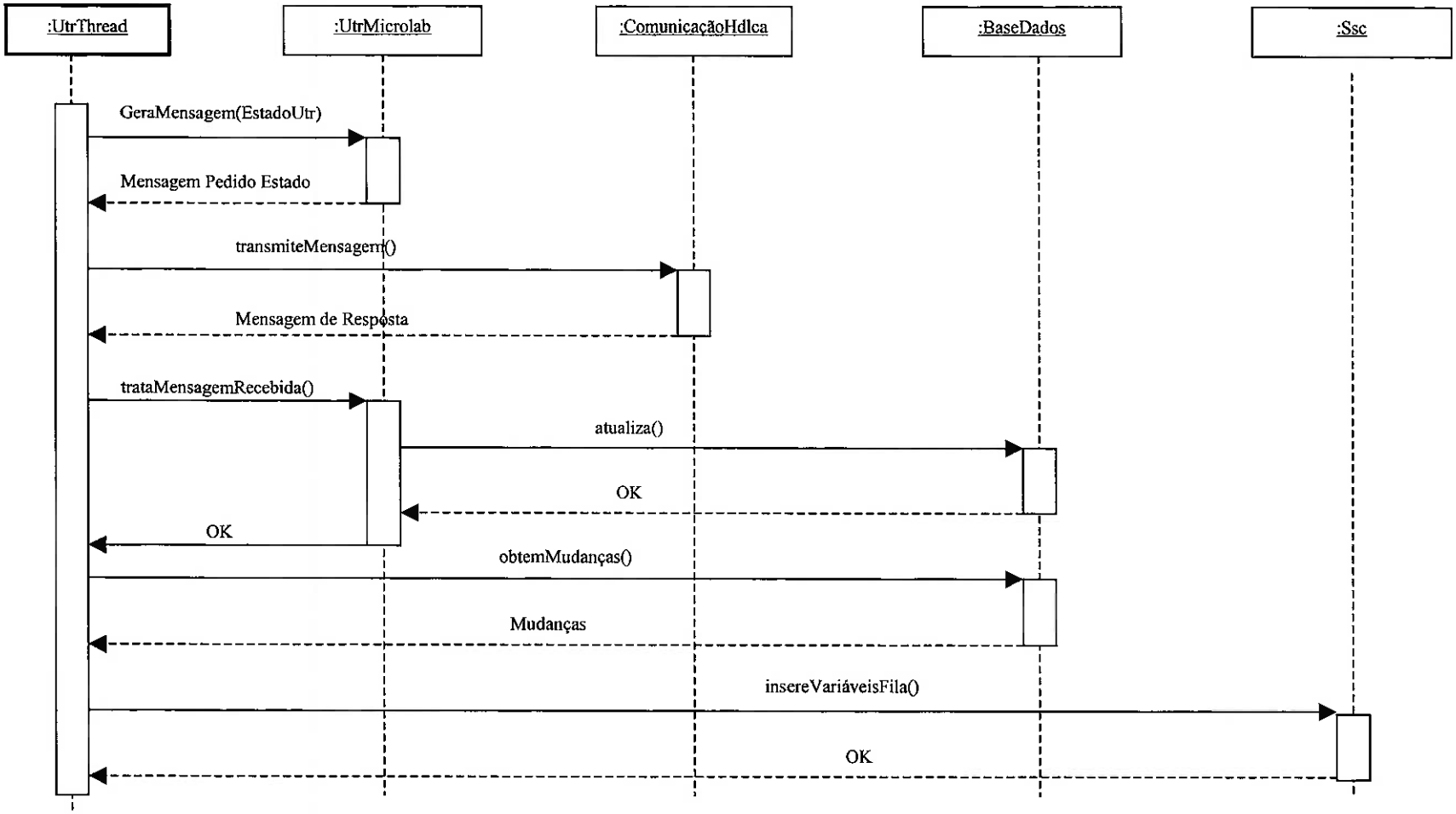

Figura 3-19 Solicita Estado da UTR Microlab

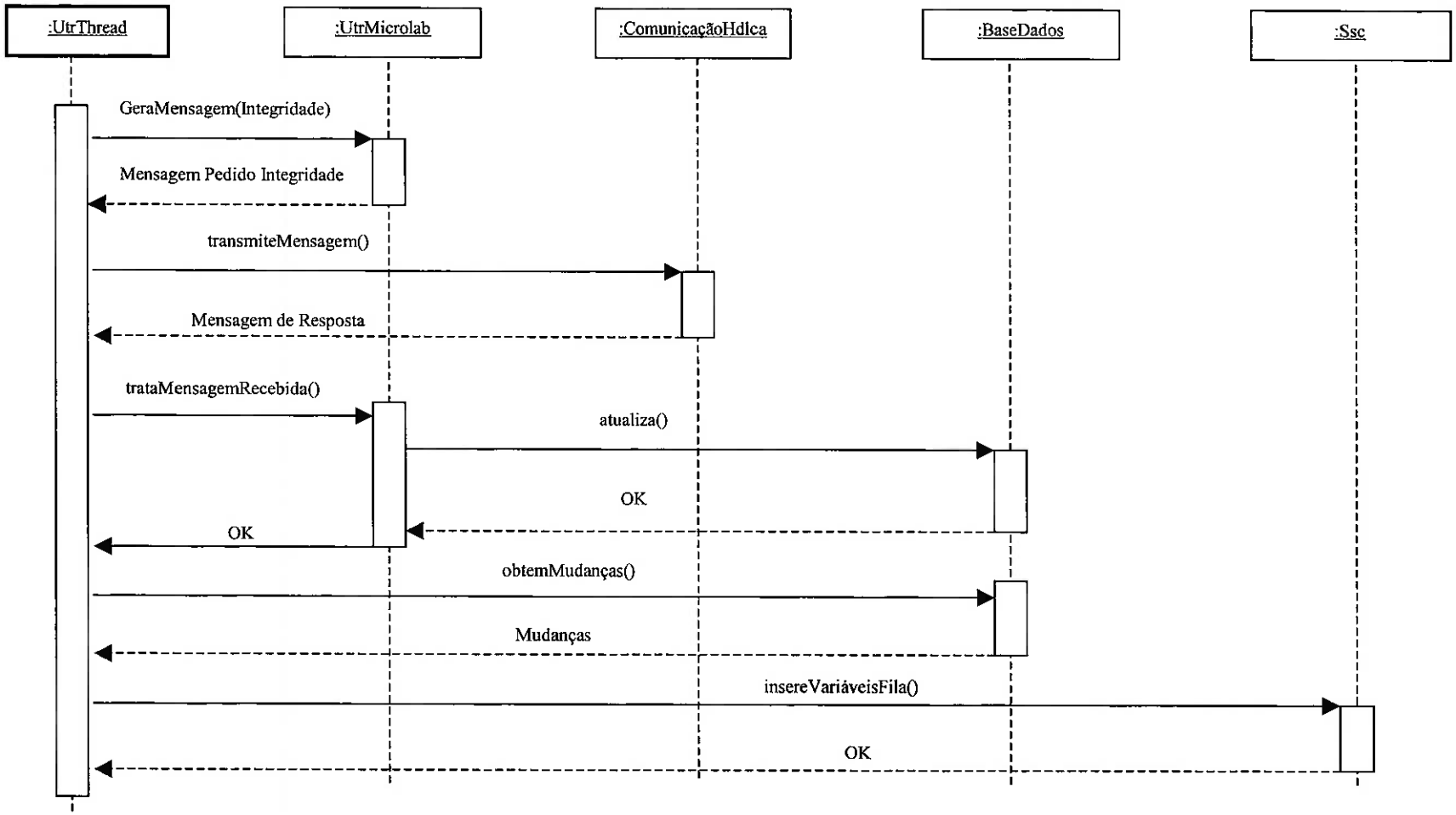

Figura 3-20 Solicita dados por integridade para a UTR Microlab

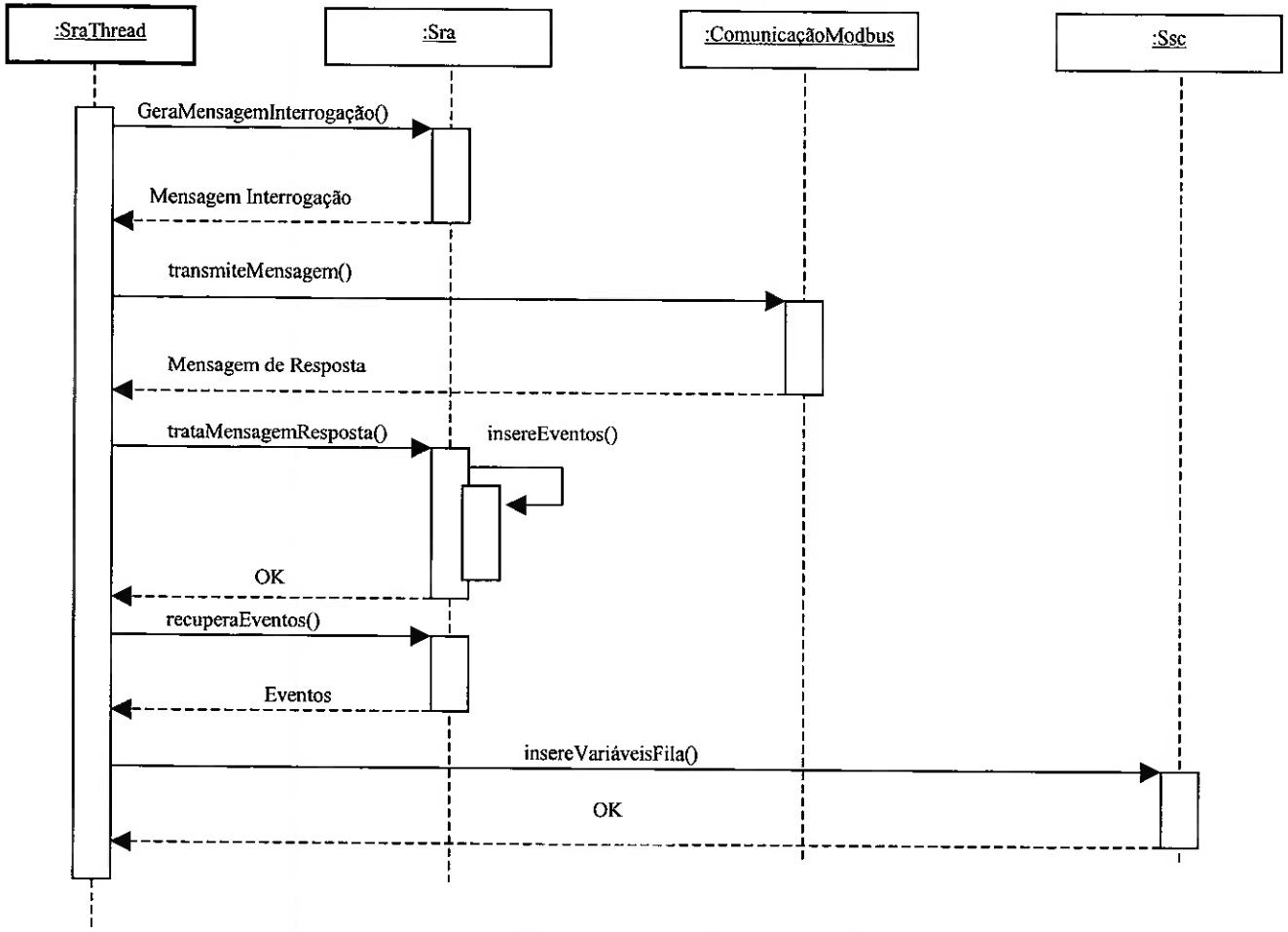

Figura 3-21 Adquire dados do Sinalizador Registrador de Alarmes

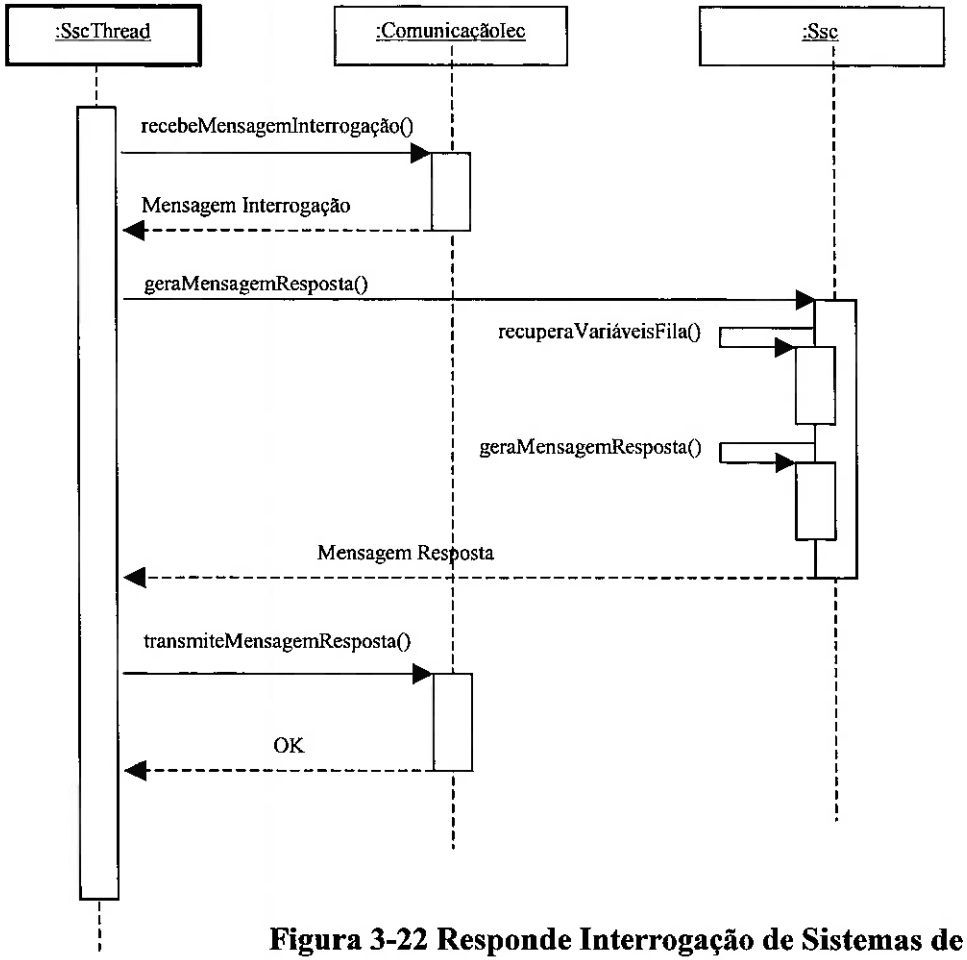

Figura 3-22 Responde Interrogação de Sistemas de Supervisão e Controle

3.9 ARQUITETURA

A definição da arquitetura de uma casa exige a elaboração de vários desenhos, cada um deles captando uma visão do problema. A planta mostra a divisão e as dimensões dos diversos cômodos, a elevação mostra a fachada, e visões específicas são necessárias para mostrar os sistemas hidráulico e elétrico. Não é possível ter-se todas as informações necessárias em um único desenho.

Sistemas de software apresentam o mesmo problema. A arquitetura de um sistema de software utiliza abstrações de alto nível para mostrar o resultado da montagem de diversos componentes agrupados de forma a satisfazer os requisitos funcionais e não funcionais. Kruchten (1995) propôs o que ele chamou de "4+1 Visões" para representar a arquitetura de um sistema de software.

Estas múltiplas visões são compostas de:

- Uma visão lógica, englobando o modelo de objetos do sistema;
- Uma visão do processo para capturar os aspectos de concorrência e sincronização;
- Uma visão física, para descrever o mapeamento do software nos componentes de hardware;
- Uma visão de desenvolvimento que descreve a organização estática do software em seu ambiente de desenvolvimento.

Estas quatro visões são ilustradas por alguns casos de uso ou cenários selecionados, tornando-se a quinta visão. A figura 3-23 descreve o modelo de arquitetura "4+1" visões:

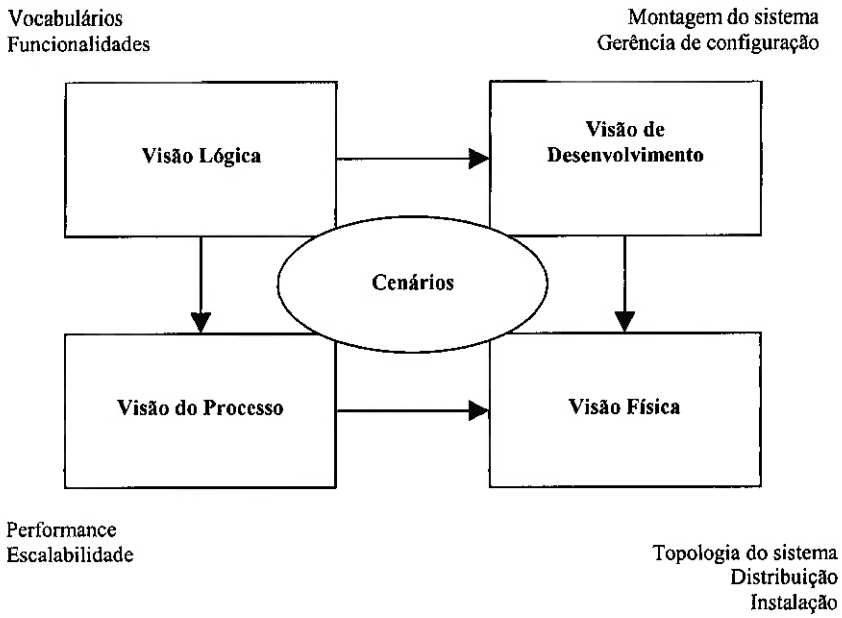

Figura 3-23 Modelo de arquitetura "4+1" visões

Neste trabalho estamos incluindo apenas a visão lógica e os cenários, tendo em vista que a proposta do trabalho é de apresentar apenas o modelo de análise do sistema. As figuras 3-24 e 3-25, apresentam a arquitetura do sistema.

3.9.1 VISÃO LÓGICA

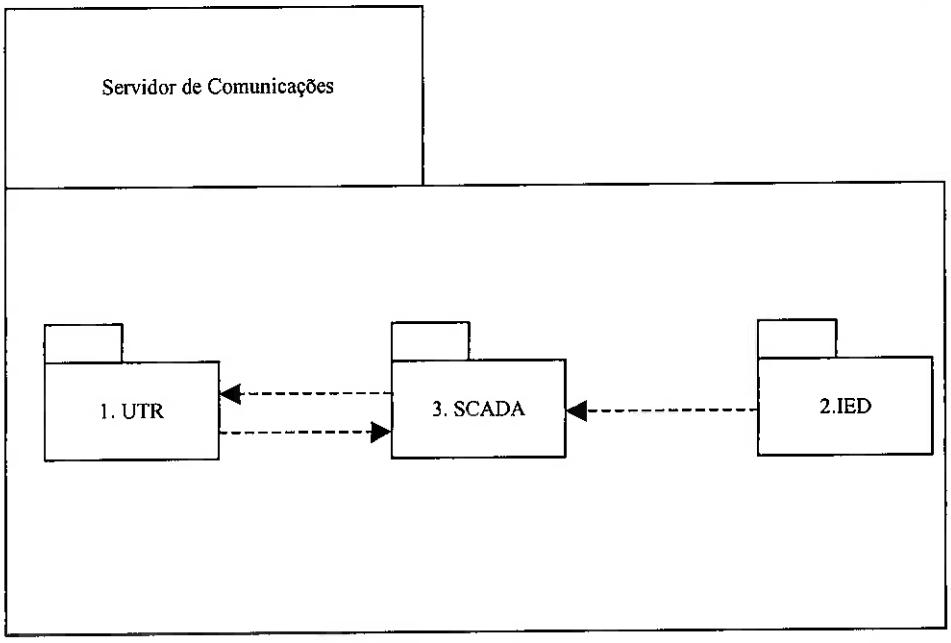

Figura 3-24 Arquitetura do Sistema - Visão Lógica

3.9.2 VISÃO DE CENÁRIO

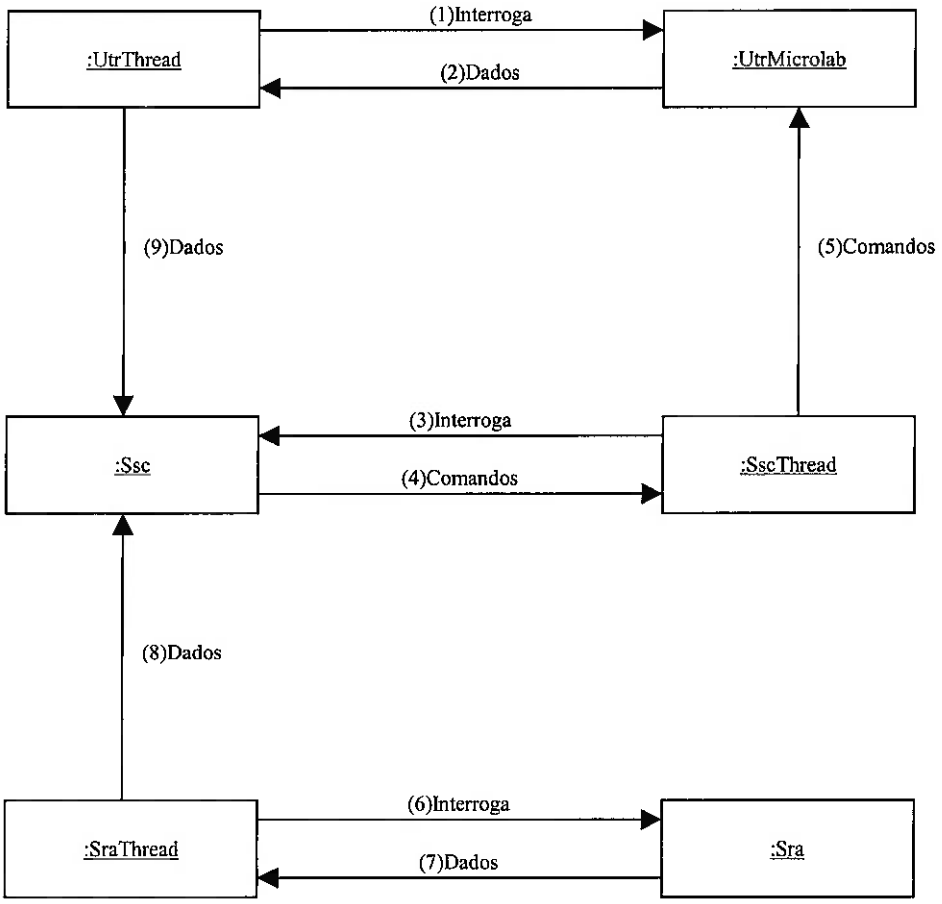

Figura 3-25 Arquitetura do Sistema - Visão de Cenário

4. CONSIDERAÇÕES FINAIS

4.1 TRABALHOS FUTUROS

Projetos de automação de subestações exigem como requisito inicial a integração das informações do processo. O sistema cuja modelagem foi apresentada neste trabalho representa um primeiro esforço de integração de informações operacionais ao nível da instrumentação atualmente existente nas subestações da CTEEP.

Apesar de integrar apenas dois equipamentos de processo, a adoção de tecnologia orientada a objetos torna o sistema modular, facilitando a expansão para inclusão de outros equipamentos. De particular interesse seria a integração de equipamentos que fazem a aquisição de informações não operacionais, tais como equipamentos de visão, proteção e oscilografia.

A disponibilidade de um Servidor de Comunicações integrando instrumentação convencional para aquisição de informações operacionais e não operacionais, poderá também servir como plataforma para uma futura migração para a arquitetura UCA.

A integração de informações, operacionais e não operacionais, conjugada com a disponibilidade de uma rede de comunicação interligando todos os Servidores de Comunicação tornaria possível uma nova geração de sistemas de supervisão e controle com características de escalabilidade, interoperabilidade e expansibilidade muito superiores às oferecidas pelos sistemas atuais.

A figura 4-1 apresenta o esboço de como se poderia implementar uma arquitetura de supervisão e controle com base em servidores de comunicação:

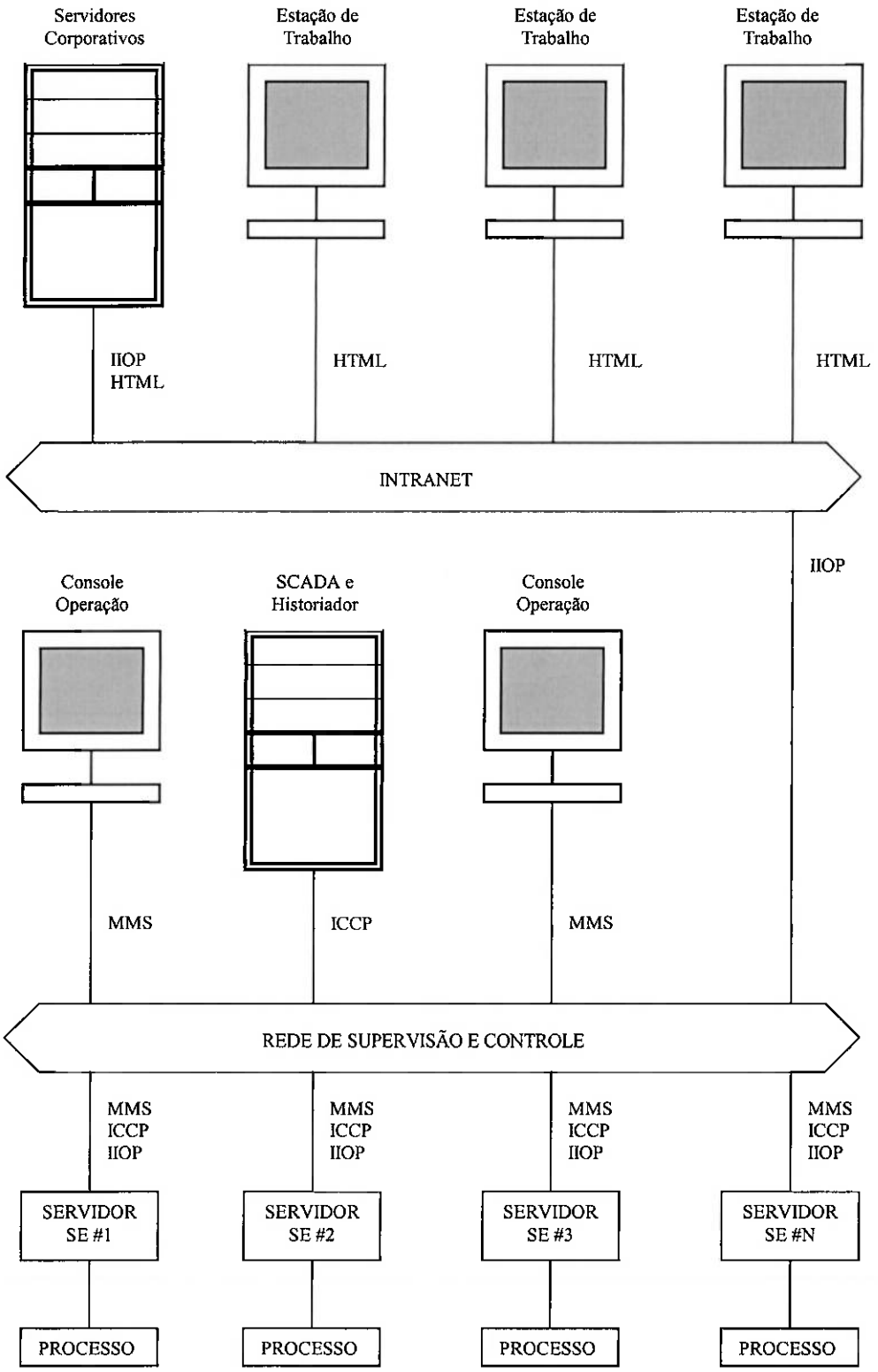

Figura 4-1 Arquitetura baseada em Servidores de Comunicação

Uma arquitetura como a mostrada na figura 4.1 permitiria que as bases de dados ficassem distribuídas nos servidores das diversas instalações. Os servidores seriam aderentes aos padrões, UCA e CORBA. Cada servidor anunciaria as

informações de que dispõe, em conformidade com o protocolo MMS (parte do padrão UCA). Os consoles de operação seriam implementados com a utilização de visualizadores (browsers) MMS. Quando tivessem necessidade de uma determinada informação, interrogariam o servidor correspondente.

A adição de novos pontos de supervisão em uma subestação afetaria apenas o servidor local, mesmo assim, apenas quando se tratar de instrumentação não aderente ao padrão UCA. Novos consoles de operação poderiam ser adicionados sem qualquer necessidade de reconfiguração, tornando muito mais simples a expansão do sistema de supervisão e controle.

A comunicação com Centros de Operação tradicionais seria feita com utilização do protocolo ICCP (também parte do padrão UCA), especificamente desenvolvido para esta finalidade e que já está sendo utilizado pelo ONS para adquirir informações dos Centros de Operação das empresas do sistema interligado.

A integração com os sistemas corporativos utilizaria o padrão CORBA para interoperabilidade entre objetos distribuídos.

O desenvolvimento de um sistema com a arquitetura proposta não é trivial e apresenta grandes desafios. Algumas das tecnologias citadas ainda não atingiram a maturidade necessária. Os padrões UCA, particularmente, carecem ainda de aprovação final. Também a infra-estrutura de telecomunicação da CTEEP necessita de adequações para fornecer os recursos necessários. Todavia, acreditamos que os ganhos potenciais em termos de escalabilidade, interoperabilidade e expansibilidade justificam plenamente o esforço de tentar faze-lo.

4.2 CONCLUSÕES

Neste trabalho foram abordados diversos elementos que devem ser considerados na modernização do processo de supervisão de subestações.

A solução apresentada atende aos requisitos funcionais e não funcionais propostos como objetivos do trabalho, através da implantação de funções de seqüênciamento de eventos, comunicação com mais de um Centro de Operação e preservação dos investimentos feitos nas Unidades Terminais Remotas.

A decisão de utilizar-se tecnologia orientada a objetos para a modelagem do sistema mostrou-se acertada, permitindo a obtenção de uma solução natural para o problema proposto.

Acreditamos que especialmente relevante é a utilização de um Servidor de Comunicações para integrar Unidades Terminais Remotas e Dispositivos Eletrônicos Inteligentes. Este conceito permitirá a adoção de uma estratégia para a evolução dos sistemas de supervisão e controle, tornando-os muito mais abrangentes que os atuais.

5. LISTA DE REFERÊNCIAS

BOOCH, G; RUMBAUGH, J; JACOBSON, I. **The unified modeling language user guide.** USA: Addison-Wesley, 1999. 482p.

INTERNATIONAL ELETROTECHNICAL COMISSION. **IEC 60870-5-101 Telecontrol equipment and systems-Part 5: Transmission protocols – Section 101: Companion standard for basic telecontrol tasks.** Geneve, 1995.

INTERNATIONAL ELETROTECHNICAL COMISSION. **IEC 60870-5-101 Telecontrol equipment and systems-Part 5: Transmission protocols – Section 101: Companion standard for basic telecontrol tasks - Amendment 1.** Geneve, 1995.

KRUCHTEN, P. Architectural blueprints – The "4+1" view model of software architecture. **IEEE Software,** v.12, n.6, p.42-50, 1995.

McNAMARA, J, E. **Technical aspects of data communications.** Bedford, Massachusetts: Digital Press, 1988. 383p.

MICROLAB S.A. **Projeto funcional das unidades terminais remotas.** Rio de Janeiro, 1984.

MODICON, INC., INDUSTRIAL AUTOMATION SYSTEMS. **Modicon Modbus protocol reference guide.** North Andover, Massachusetts, 1996.

PRESSMAN, R, S. **Software engineering: a practitioner's approach.** New York: McGraw-Hill Companies, Inc, 2001. 860p.

ROWLETT, T. **The object-oriented development process.** Upper Saddle River, NJ: Prentice Hall, 2001.