UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA DE CAMPINAS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

AUTOMATIZAÇÃO DO

GERENCIAMENTO DE CONFIGURAÇÃO DE SOFTWARE

Por: MIRIAM AKEMI MANABE CAPRETZ

Orientador: Prof. Dr. MARIO JINO

Dissertação apresentada à Faculdade de Engenharia de Campinas da Universidade Estadual de Campinas, como parte dos requisitos exigidos para a obtenção do Grau de Mestre.

AGOSTO - 1988.

Este exemplar corresponde à redação final da tese defendida por Miriam Akemi Manabe Capretz e aprovada pela Comissão Julgadora em 12 de agosto de 1988.

Mario Jino

AGRADECIMENTOS

Ao Prof. Dr. Mario Jino, pela orientação eficiente e precisa, pelo apoio e incentivo recebido nas atividades acadêmicas, pelas sugestões e discussões de idéias, pela amizade, compreensão e dedicação empreendida para que este trabalho fosse concluído no seu devido tempo.

Ao Prof. Dr. Caetano Traina Junior, pela amizade, pelas sugestões e discussões de suas idéias que contribuíram para este trabalho.

Ao Mário Bento de Carvalho, pela oportunidade que me foi dada de trabalhar no projeto SIPS e pelo apoio dado para que este trabalho fosse realizado.

Ao Luiz Fernando Capretz, pelo incentivo e constante cobrança para que este trabalho se concluísse.

Aos amigos da Divisão de Engenharia de Software do Instituto de Automação do Centro Tecnológico para Informática: Fábio Nauras Akhras, Márcia Cristina Carvalho Costa, Wagner Cezarino, Gláucia Dantas Franco Azevedo e Angelina Oliveira, que com seu apoio possibilitaram a realização deste trabalho.

Agradeço também a todos que, direta ou indiretamente, colaboraram para a concretização deste trabalho.

RESUMO

O Gerenciamento de Configuração de Software (GCS) é uma disciplina que visa controlar as modificações de cada um dos componentes gerados durante o ciclo de vida de um produto de software. Os princípios do GCS formam a base de qualquer processo de desenvolvimento de software, quando suportado manual ou automaticamente em um ambiente de desenvolvimento de software. A automatização desta disciplina facilita o seu uso por projetistas, pois sem um suporte computacional adequado, é uma tarefa trabalhosa e que requer tempo, sendo portanto negligenciada pelos projetistas.

O objetivo deste trabalho é mostrar como os procedimentos da disciplina Gerenciamento de Configuração de Software podem ser automatizados no SIPS (Sistema Integrado para Produção de Software), um ambiente voltado à produção de software, que está em desenvolvimento no Instituto de Automação do Centro Tecnológico para Informática. Para isto, a disciplina Gerenciamento de Configuração de Software foi formalizada e modelada usando o MRO (Modelo de Representação de Objetos). A sua implementação básica consiste na construção de primitivas para o ambiente SIPS e de uma ferramenta denominada Gerenciador de Configuração de Software. A automatização do GCS proposta é possível uma vez que o ambiente SIPS fornece uma série de características favoráveis providas por uma base de dados centralizada e única que é utilizada por várias ferramentas para desenvolvimento de sistemas de software cobrindo potencialmente todas as fases do ciclo de vida.

ABSTRACT

Software Configuration Management (SCM) is a discipline whose objective is to control changes in each one of components generated during the life cycle of a software product. The principles of SCM constitute a base for any software development process, when supported manual or automatically in a software development environment. The automation of this discipline eases its use by designers; without an adequate computational support, it is a hard task and demands time, being usually neglected by designers.

The objective of this work is to show how the procedures of Software Configuration Management can be automated in SIPS (Integrated System for Software Production), an environment aimed to support the software production, under development at Automation Institute of Informatica Technological Center. For this purpose, the Software Configuration Management discipline must be formalized and modeled using the ORM (Object Representation Model). Its basic implementation consists in the construction of primitives for SIPS environment and a tool named Software Configuration Manager. The SCM automation proposed is possible because SIPS environment provides suitable characteristics supplied by a centralized database used by several tools for development of software systems covering potentially all life cycle phases.

ÍNDICE

Pág.

LISTA DE FIGURAS

Pág.

CAPÍTULO 1

INTRODUÇÃO

A evolução do software não acompanhou na mesma proporção a evolução do hardware. O software, quando desenvolvido da maneira tradicional, na maioria das vezes funciona de forma inadequada, tem um custo maior que o estimado, ultrapassa o cronograma, é mal documentado, prejudicando não só a manutenção como o próprio desenvolvimento do sistema.

Desenvolver grandes projetos de software, não só em tamanho como também em complexidade, tem se tornado uma atividade difícil e cara. Grandes projetos ainda envolvem os problemas de comunicação e coordenação, pois envolvem muitas pessoas e consequentemente surgem os problemas inerentes à interação humana, consistência das informações ao longo do projeto, dificultando o controle da própria evolução do projeto.

A necessidade de comunicação e coordenação deve-se ao fato de ocorrerem mudanças contínuas no software. Todos os sistemas grandes e amplamente utilizados estão sujeitos a correções infindáveis, melhoramentos, e diversificações; isto porque é aparentemente mais barato modificá-los do que refazê-los. Entretanto, modificações contínuas significam comunicação contínua em todas as estruturas do sistema, decisões de projeto, detalhes de implementação para o programador e controle contínuo para evitar (ou baixar) a deterioração do sistema.

Existem grandes investimentos na construção de software. Como mudanças no software são necessárias, seria bom se fosse possível preservar esses investimentos. Porque mudanças no software são tão difíceis? Para a maioria dos sistemas de software, o projeto original é inacessível, pois embora a análise de requisitos e especificações originais tenham sido documentadas, nem sempre são atualizadas para refletir as modificações feitas.

Pesquisas recentes em metodologias de programação, linguagens de programação, técnicas de especificação e verificação levam a uma visão diferente: o objetivo principal é reduzir a complexidade e tamanho de sistemas de software para

dimensões gerenciáveis. Entretanto, não importando o sucesso que será obtido por estas abordagens, as aplicações de computação tendem a se expandir até os limites da tecnologia. Assim, os problemas de coordenação, comunicação e atualização inerentes ao desenvolvimento de grandes projetos têm que ser resolvidos.

Diante dos problemas que ocorrem no desenvolvimento de um projeto de software, verificou-se a necessidade de disciplinas que assegurem a integridade do produto final, isto é, um produto que satisfaz as necessidades funcionais do usuário, que pode ser completa e facilmente acompanhado ao longo de seu ciclo de vida, que cumpre critérios especificados de desempenho, que tem um custo final compatível com o estimado e que é liberado dentro do prazo estabelecido; a integridade do produto é uma medida da satisfação das necessidades e expectativas do usuário.

Para desenvolver um produto que possua as características de integridade, é importante a utilização de disciplinas que forneçam apoio ao trabalho dos gerentes de projetos. Entre essas disciplinas encontram-se:

- Gerenciamento de Configuração de Software (GCS);
- Garantia de Qualidade;
- Validação e Verificação;
- Teste e Avaliação.

A aplicação adequada destas disciplinas pelo gerente de projeto é imprescindível para o sucesso do sistema, principalmente para grandes sistemas que envolvem muitos programadores e que consistem de vários subsistemas.

é necessária a imposição de disciplina ao longo do ciclo de desenvolvimento de um software de forma que o mecanismo iterativo de fato convirja aos produtos requeridos. O resultado da aplicação desta disciplina é tornar o software mais visível ao gerenciamento, permitindo desta forma que a organização responsável controle todo o ciclo do desenvolvimento de um software.

O objetivo deste trabalho é mostrar como os procedimentos da disciplina Gerenciamento de Configuração de Software podem ser automatizados no ambiente SIPS (Sistema Integrado para Produção de Software), um ambiente voltado à produção de software, que está em desenvolvimento no Instituto de Automação do Centro Tecnológico para Informática.

No Capítulo 2 apresentam-se os conceitos pertinentes a esta disciplina. São apresentados os motivos que têm levado a se tentar automatizar o GCS bem como exemplos de automatização de GCS; estes incluem ferramentas individuais para automatizar um procedimento específico e ambientes em cuja concepção está incluída a automatização do GCS.

No Capítulo 3 apresenta-se o ambiente SIPS (Sistema Integrado para Produção de Software) dentro de um enfoque voltado a ressaltar as características necessárias deste ambiente para o desenvolvimento deste trabalho.

No Capítulo 4 apresenta-se a definição da metodologia GCS para o ambiente SIPS, com a definição de termos para esta metodologia, o posicionamento dos conceitos do GCS no SIPS, o modelamento desta metodologia para o ambiente SIPS bem como um exemplo de integração da metodologia GCS com a metodologia Análise Estruturada de Sistemas para o SIPS.

O Capítulo 5 discorre sobre a forma de implementação desta metodologia para o SIPS, apresentando as modificações necessárias para o ajustamento desta metodologia ao ambiente, bem como a especificação de novas primitivas e da ferramenta Gerenciador de Configuração de Software para suportar a automatização do GCS no SIPS.

No Capítulo 6 apresentam-se as conclusões e sugestões para trabalhos futuros.

CAPÍTULO 2

O GERENCIAMENTO DE CONFIGURAÇÃO DE SOFTWARE

2.1 - Introdução

Uma das maneiras de definirmos gerenciamento de configuração é através da inspeção de cada uma das palavras individualmente. **Gerenciamento** é o processo de estabelecer e organizar objetivos, seguido pelo planejamento e emprego de recursos para atingir esses objetivos. **Configuração** refere-se a um conjunto interrelacionado de características funcionais e/ou físicas de hardware/software que constituem um produto.

O Gerenciamento de Configuração de Software (GCS) [2], [4], [5], [20] e [30], é a disciplina que identifica a configuração de um sistema em instantes de tempos discretos, com o propósito de controlar sistematicamente mudanças nessa configuração e manter a integridade e a rastreabilidade do sistema, ao longo de seu ciclo de vida. Possui a responsabilidade de gerenciamento de cada detalhe de um projeto de software, desde a sua concepção, desenvolvimento, término e manutenção do produto através do controle de várias versões de todos os documentos, componentes de software, decisões de projeto, etc., relativos a cada fase do desenvolvimento do sistema em suas várias formas e linguagens de representação. A evolução e os desdobramentos de cada item do sistema ao longo do tempo, e dentro da estrutura lógica, devem ser acompanhados e monitorados durante a vida do sistema. Ele deve registrar, fornecer e controlar as informações sobre as quais todas as outras ferramentas e métodos do ciclo de vida operam.

O GCS é uma disciplina criada com o objetivo de assegurar a integridade do produto final. Ela é bastante útil ao gerente de projeto na realização de uma tarefa complexa, permitindo que esta seja realizada de uma maneira ordenada, maximizando a produtividade e minimizando os erros.

O GCS deve registrar a história de todas as informações usadas pelo projeto para possibilitar a rastreabilidade. Deve também incorporar o controle de mudanças nas partes de um sistema, armazenando todo o histórico associado às mudanças para garantir a segurança e a integridade. Todas as atividades do ciclo de vida devem ser acompanhadas e monitoradas e a localização de todas as versões devem ser conhecidas.

Para tanto, o GCS requer uma rigorosa disciplina para o desenvolvimento e uma grande quantidade de informação deve ser coletada, mantida, controlada e relatada para a contínua evolução do sistema.

É importante observar que a utilização do GCS, em um projeto de desenvolvimento de software, requer tempo e dinheiro. Deve-se, no entanto enfatizar que esses gastos têm retornos, tais como: um produto que se comporta de forma esperada, que está de acordo com as exigências do usuário, que é bem projetado, bem documentado, cujo custo não ultrapassa o valor estimado e que é entregue no prazo correto.

No contexto deste trabalho, será adotada a definição de software dada em [4]:

"Software é informação, que é:

- estruturada com propriedades lógicas e funcionais;
- criada e mantida sob várias formas e representações durante o ciclo de vida;
- apropriada para processamento em computador quando em seu estado completamente desenvolvido".

Desta forma, software não é somente um conjunto de programas de computadores, mas inclui a documentação requerida para definí-los, desenvolvê-los e mantê-los.

O GCS é derivado de uma disciplina aplicada tradicionalmente ao hardware [24], o Gerenciamento de Configuração, que está definida como:

"Uma disciplina que aplica direção técnica e administrativa, e supervisão para:

a) Identificar e documentar características físicas e funcionais de um item de configuração;

b) Controlar mudanças nestas características;

c) Registrar e relatar processamento de alterações e estado de implementação;

d) Verificar aderência à especificação e outros requisitos contratuais associados".

Princípios e procedimentos de Gerenciamento de Configuração, tradicionalmente aplicados ao desenvolvimento de hardware, têm sido desenvolvidos e aplicados para programas de computador nos últimos vinte e cinco anos [20]. Contudo, diferenças fundamentais entre software e hardware não permitem a aplicação direta de Gerenciamento de Configuração de Hardware ao software. Recentemente, estão surgindo esforços no sentido de estabelecer a disciplina de Gerenciamento de Configuração de Software.

2.2 - Funções do Gerenciamento de Configuração de Software

Uma discussão de GCS introduz um conjunto de questões complexas:

- Como uma organização identifica e gerencia as várias versões de um programa e sua documentação de maneira que uma alteração seja eficientemente aplicada ao sistema?

- Como uma organização controla as alterações antes e depois do software ser entregue ao cliente?

- Quem tem a responsabilidade para aprovar e priorizar as alterações?

- Como é possível garantir que a alteração foi feita corretamente?

- Que mecanismo é usado para comunicar aos interessados sobre as alterações que vão sendo efetuadas?

Estas questões levam-nos às quatro funções do GCS definidas em [4]: identificação, controle, auditagem e administração de estado.

2.2.1 - Identificação de Configuração de Software

Esta função visa definir, identificar e rotular os componentes do software e os relacionamentos entre os mesmos, componentes estes que requeiram atenção individual por parte do GCS.

O gerenciamento efetivo do desenvolvimento de um sistema requer uma definição cuidadosa dos componentes da **configuração básica** (definida abaixo); mudanças nestes componentes também devem ser definidas uma vez que estas mudanças, junto com as configurações básicas, especificam a evolução do sistema.

Uma **configuração básica** do sistema nada mais é do que um ponto de referência ou um marco no desenvolvimento de um sistema [7]. As configurações básicas servem como marcos que dividem as várias fases do ciclo de vida do software. Atualizar estas configurações básicas são como dar um passo na evolução do sistema dentro do ciclo de vida. O papel da identificação de configuração de software no processo do GCS é fornecer rótulos aos componentes que estabelecem os marcos e a sua evolução.

Uma atualização de uma configuração básica representa uma configuração básica mais um conjunto de mudanças que foram incorporadas a ela. Cada uma destas configurações básicas estabelecidas durante o ciclo de vida de um sistema de software controla os passos subsequentes do desenvolvimento do software. Inicialmente, é estabelecida uma configuração básica de software que representa o software atual no seu estado mais recente. Quando mudanças são feitas à configuração básica estabelecida mais recentemente, então, do ponto de vista do gerente de configuração de software, esta configuração básica e suas alterações representam o software atual no seu estado mais recente (embora do ponto de vista do projetista o software atual pode estar em um estágio mais avançado).

A entidade mais básica na identificação de configuração de software é o **Item de Configuração de Software (ICS)**. Visto pela perspectiva de GCS, uma configuração básica de software aparece como um conjunto de ICSs. Os ICSs dentro de uma configuração básica estão interrelacionados através de uma árvore hierárquica como aparece na Figura 2.1, que mostra um exemplo de evolução de ICSs em um sistema. Como o sistema de software evolui ao longo de seu ciclo de vida, o número de níveis nesta hierarquia geralmente aumenta; a primeira configuração básica pode consistir de somente um ICS. O nível mais baixo de ICSs na hierarquia pode ainda estar sob desenvolvimento e portanto não estar ainda

sob controle do GCS. Estas entidades podem ser objetos de projeto (op) ou componentes de programas.

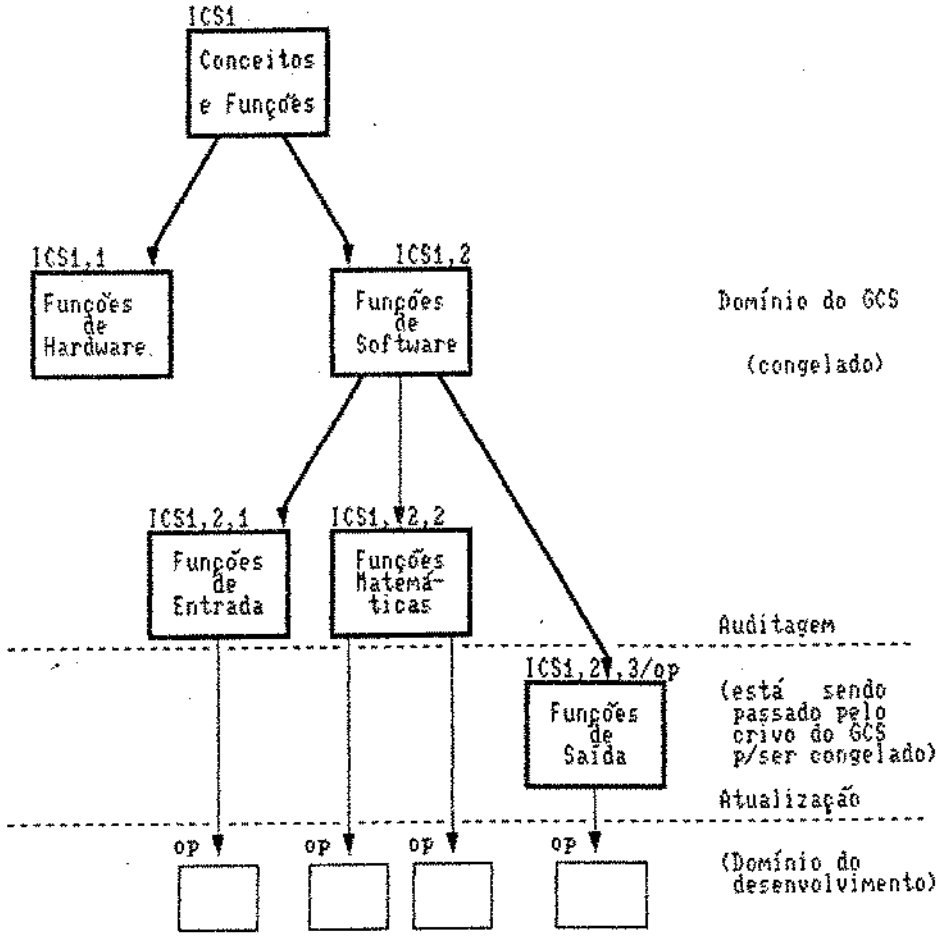


Figura 2.1 - A Evolução de ICSs junto com a Evolução do Sistema

Cada configuração básica e cada membro da família de atualizações associadas existirão em uma ou mais formas, tais como documentos de projeto, código fonte em disco, ou código objeto executável.

Executando a função de identificação, o gerente de configuração de software está, de fato, registrando os estados dos ICSs. Cada configuração básica e suas atualizações associadas coletivamente representam a evolução do software durante cada um dos estágios de seu ciclo de vida. Estes estágios estão relacionados uns com os outros. Assim, a coleção de estágios do ciclo de vida parece uma sequência de vários estados de árvores de ICSs relacionados e sobrepostos. Imaginando esta coleção de sequências dos vários estados tratados em uma ordem

cronológica pode-se ver a história da evolução do software. Consequentemente, a identificação de configurações básicas e suas atualizações fornece um documento explícito que liga todos os estágios do ciclo de vida do software. Com o apoio desta documentação, o projetista de software pode assegurar a integridade do seu produto, e o usuário de software pode assegurar a integridade do produto que está adquirindo.

2.2.2 - Controle de Configuração de Software

Esta função visa controlar modificações no sistema através de controle de preparação de modificações e controle de execução de modificações. O documento básico para esta função é a **Proposta de Modificação de Software (PMS)** que deve conter informações sobre modificações propostas tais como: descrição das modificações, justificativas, resultados de análise sobre impacto das modificações, etc.

A evolução de um sistema de software é, na linguagem de GCS, o desenvolvimento de configurações básicas e a incorporação de uma série de modificações nas configurações básicas. Um papel do controle de configuração de software é fornecer o mecanismo administrativo de precipitar, preparar, avaliar e aprovar ou desaprovar todas as modificações propostas aos componentes de configurações básicas ao longo do ciclo de vida de um projeto de software.

O software, para o propósito de GCS, é uma coleção de ICSs que estão interrelacionados de uma maneira bem definida. As primeiras configurações básicas e suas atualizações associadas são documentos de especificação; nos últimos estágios e suas atualizações associadas, cada ICS pode se manifestar em alguma ou várias representações de software. O controle de configuração de software preocupa-se em gerenciar modificações de ICSs em todas as suas representações.

Este processo envolve três ingredientes básicos:

a) Documentação (tais como formulários administrativos e material de suporte técnico e administrativo) para formalmente precipitar e definir uma modificação proposta para um sistema de software;

b) Um corpo organizacional para formalmente avaliar e aprovar ou desaprovar uma modificação proposta para um sistema de software (Comitê de Controle de Configuração - CCC);

c) Procedimentos para controlar modificações de um sistema de software. A Proposta de Modificação de Software (PMS) é um documento de controle, contendo informações tais como descrição da modificação proposta, identificação da organização que requisitou a modificação, razões para a modificação, identificação das configurações básicas e ICSs afetadas e especificação do impacto sobre custo e cronograma. PMSs são revisadas e coordenadas por um CCC, que é tipicamente um corpo que representa todas as unidades da organização que tem algum tipo de interesse nas modificações propostas.

A Figura 2.2 mostra o processo de controle de configuração de software. Tal como mostrado nesta figura, a incorporação de modificações não é uma função do GCS, mas sim o monitoramento do processo da implementação da modificação que resulta em uma incorporação de modificação. Esta figura enfatiza ainda que a análise possivelmente requerida para preparar uma PMS está fora do escopo do GCS. Esta análise, referente a custo, cronograma, impacto e definições da modificação, etc., é avaliada pelo CCC do qual fazem parte pessoas que estão envolvidas ou afetadas pela modificação proposta. Deve-se observar que as PMSs não aprovadas pelo CCC não são simplesmente descartadas, mas arquivadas para possíveis referências futuras.

Os documentos de modificações sugeridas, os quais podem ser de três tipos, são apresentados pelas diversas pessoas envolvidas com o projeto.

A Proposta de Modificação pode ser originada por programadores, operadores, testadores, gerentes ou clientes que desejam modificar itens que estão sendo correntemente controlados no sistema.

O Relatório de Discrepâncias pode ser originado por programadores, testadores, gerentes, clientes, etc., ao se observarem erros na configuração corrente.

A Incorporação de Novas Funções pode ser originada por engenheiros de sistemas ou gerentes de projeto requisitando novas funções.

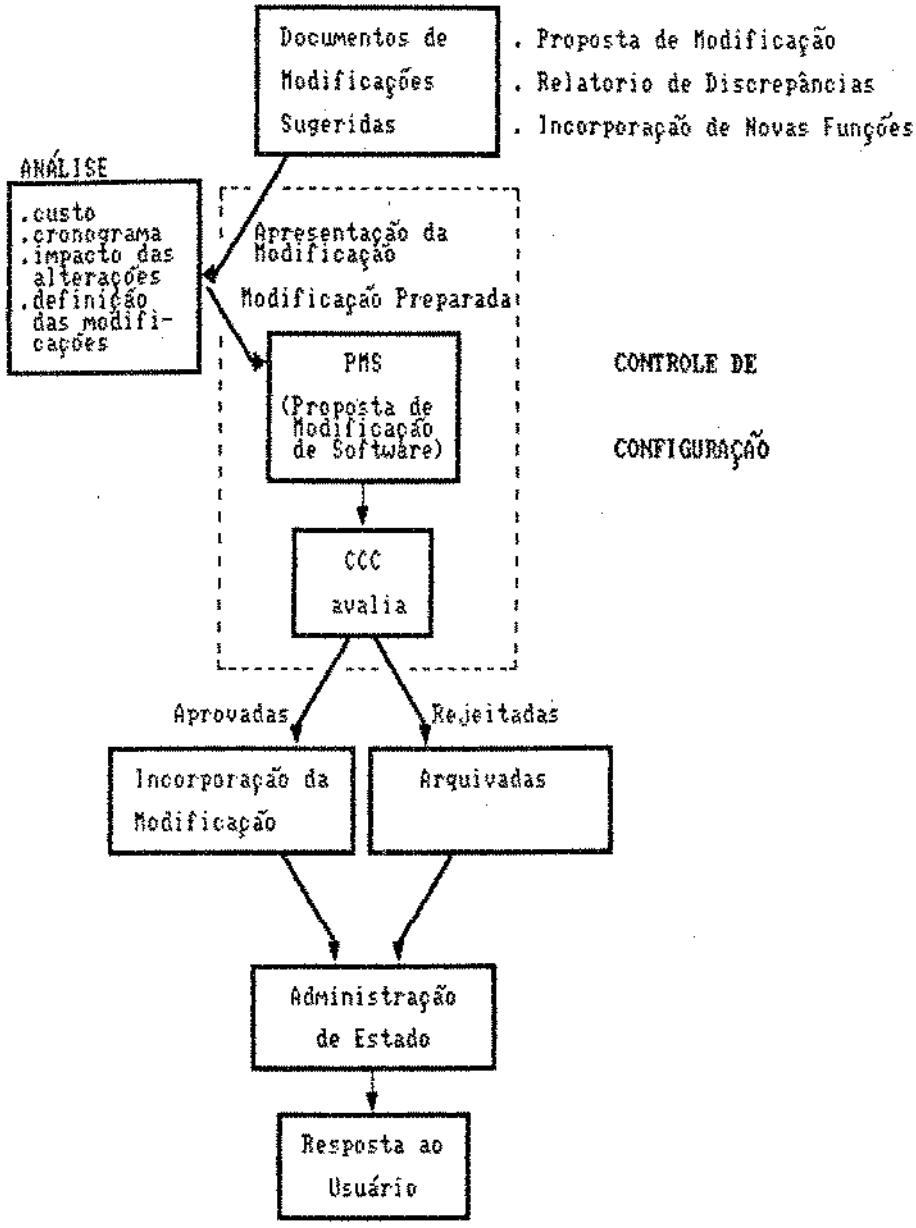


Figura 2.2 - O Controle de Configuração de Software

2.2.3 - Auditagem de Configuração de Software

A auditagem de configuração de software garante o produto final para todas as pessoas envolvidas com este produto de

software: o usuário, o cliente e o vendedor. Primeiro, ao usuário é assegurado que o software satisfaz suas necessidades operacionais e de desempenho. Segundo, ao cliente é assegurado que o software está pronto no prazo previsto e sem erros. E finalmente, ao vendedor é assegurado que seu produto tem evoluído de uma maneira rastreável (e portanto manutenível) e é aceito pelo usuário.

A auditagem de um projeto de software envolve dois processos fundamentais - **verificação e validação** - no contexto do software. Ela também fornece o mecanismo para estabelecer formalmente uma configuração básica [7].

A **verificação** garante que as atualizações pretendidas para cada ICS em uma configuração básica estão presentes na configuração básica sucessiva. Portanto, cada configuração básica candidata a ser criada é verificada em relação à configuração básica precedente. Cada verificação estabelece a rastreabilidade entre os ICSs de uma configuração básica e os de configurações básicas sucessoras ou predecessoras. A verificação consiste na correlação de ICSs de um produto com os ICSs correspondentes da configuração básica precedente. ICSs correlacionados são então comparados para garantir que o intento da configuração básica precedente foi implementado no produto auditado.

A **validação** garante que a configuração de ICSs resolve o problema de maneira correta, ou seja, que as necessidades do usuário são satisfeitas; a cada estágio do ciclo de vida, o software é confrontado com os objetivos especificados na especificação de requisitos. A validação determina a rastreabilidade entre os ICSs de uma configuração básica e os ICSs da especificação de requisitos. Uma vez que nem sempre todos os requisitos são completamente conhecidos no início do projeto, estes podem ser modificados durante o projeto. Neste caso, as configurações básicas existentes devem ser revalidadas em relação aos novos requisitos.

O processo de auditagem de software torna visível ao gerente o estado corrente do software dentro de seu ciclo de vida. A auditagem também revela se os requisitos do projeto estão sendo satisfeitos e se o intento da configuração básica precedente foi preenchido, como mostra a Figura 2.3. Com esta visibilidade, o gerente do projeto pode avaliar a integridade do produto de software que está sendo desenvolvido, resolver problemas que surgirem durante as auditagens e corrigir defeitos no processo do desenvolvimento. A visibilidade fornecida pelo software auditado fornece uma base para estabelecer o produto auditado no ciclo de vida como uma nova configuração básica.

O resultado de um software auditado pode ser o estabelecimento de uma configuração básica, o redirecionamento de uma tarefa do projeto, ou um ajustamento dos recursos aplicados no projeto.

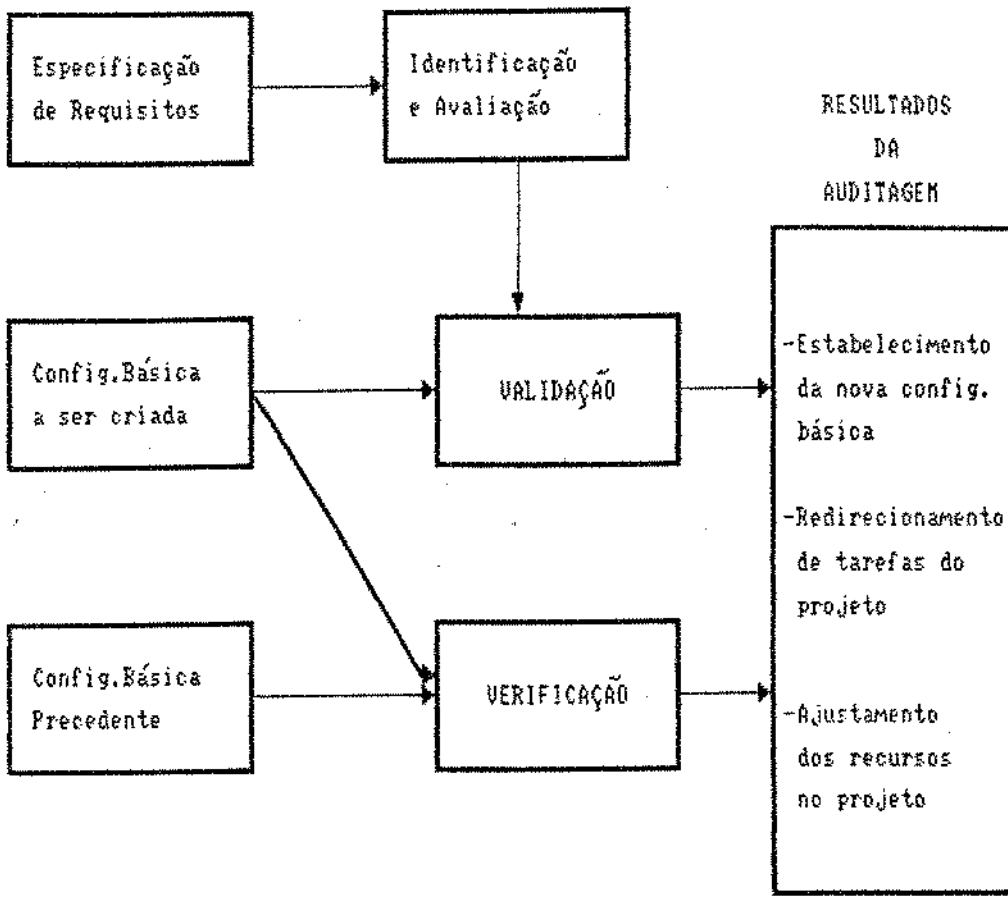


Figura 2.3 - O Processo de Auditagem de Software

A seguir, esclarece-se a maneira como os processos de verificação e validação podem ser combinados para fornecer um procedimento genérico para a auditagem de produtos do ciclo de vida de um software. O procedimento está representado na Figura 2.4. O modelo desta figura está projetado para ser aplicado a qualquer produto do ciclo de vida do software.

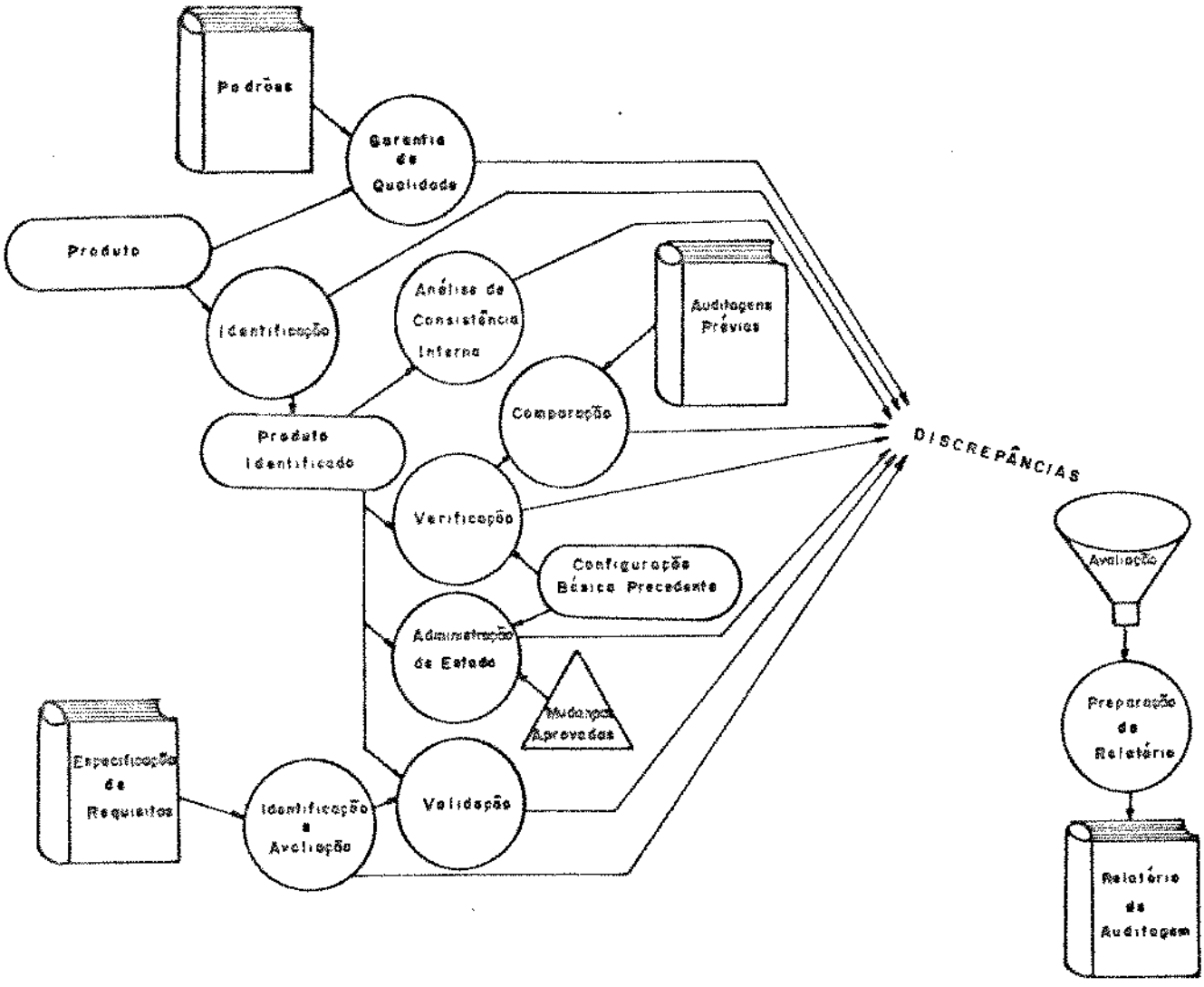


Figura 2.4 - Um Modelo de Procedimento para a Auditagem do Software

O primeiro passo do processo mostrado na Figura 2.4 examina a especificação de requisitos, assumindo que o conteúdo deste documento não tenha sido previamente identificado e avaliado quanto à completeza, clareza, consistência e testabilidade. Geralmente, a especificação de requisitos requer identificação e avaliação somente para auditar o primeiro produto de software desenvolvido, ou quando os requisitos são alterados.

No próximo passo, o produto passa pelo processo de garantia de qualidade. Este processo garante que o produto está em conformidade aos padrões que haviam sido especificados. Estes padrões podem ser governamentais, industriais, comerciais ou profissionais; eles foram préestabelecidos pelo gerente de projeto ou gerente geral. Um exemplo da função de garantia de qualidade é determinar se o produto concorda no formato, conteúdo e metodologia com os padrões prescritos.

Após o processo de garantia de qualidade, o produto é identificado, ou seja, identificam-se os ICSs que constituem o produto.

O produto identificado é então examinado quanto à completeza, clareza, consistência interna e testabilidade. Este exame é executado através de comparações cuidadosas com o conteúdo de outras seções do documento. As inconsistências, ambiguidades, omissões, adições ou mudanças inapropriadas observadas no escopo são registradas.

A seguir, o produto identificado é verificado contra a configuração básica precedente. Se não houver configuração básica precedente, este passo não é executado. Em alguns casos é necessário verificar o produto identificado em relação as auditagens anteriores bem como com a configuração básica precedente. Os ICSs correlacionados são então comparados para garantir que o intento da configuração básica precedente foi implementado no produto auditado.

As discrepâncias observadas durante a verificação são comparadas com os resultados de auditagens prévias e dos relatórios de administração de estado de configuração. A resolução das discrepâncias prévias é também determinada neste processo. Quando o produto sendo auditado é uma atualização de um produto de software auditado previamente, a administração de estado é executada. Este processo consiste em comparar os ICSs do produto identificado com os ICSs identificados na configuração básica precedente, para determinar quais ICSs, se houver, foram alterados, retirados ou adicionados. Estas alterações, retiradas e adições são comparadas com a lista de mudanças aprovadas para a configuração básica precedente e as discrepâncias são registradas.

A seguir, o produto identificado é validado em relação à especificação de requisitos identificada. Os ICSs do produto que está sendo auditado são correlacionados com os ICSs da especificação de requisitos. Este é um passo para obtenção da rastreabilidade. Os ICSs correlacionados são comparados em

detalhes para certificar que os requisitos foram totalmente satisfeitos no produto.

Após todos esses exames terem sido feitos, as discrepâncias descobertas são avaliadas e um relatório final é preparado. A avaliação é baseada em fatos observados. O auditor então usa sua experiência para desenvolver conclusões baseadas no tipo, número e complexidade das discrepâncias observadas. A conclusão reflete o julgamento que deve ser colocado no relatório da auditagem.

O relatório de auditagem é produzido e enviado ao Comitê de Controle de Configuração (CCC) e aos gerentes de projeto. O CCC atua sobre cada relatório fornecido pela equipe de auditagem, aprovando o produto de software ou direcionando o trabalho de remover as deficiências relatadas pela auditagem.

Estes passos, definem um exemplo do processo da auditagem para um produto do ciclo de vida do software. Para o desenvolvimento de um sistema estes passos devem ser seguidos para cada um dos produtos das fases do ciclo de vida que vão sendo gerados.

2.2.4 - Administração de Estado da Configuração

Esta função fornece um mecanismo para manter um registro de como o sistema está evoluindo, bem como o seu estado corrente durante o desenvolvimento.

A administração de estado acompanha e relata todos os itens de software formalmente identificados e controlados, e mantém esses registros para suportar o processo de auditagem de software. Esta função registra todas as atividades das outras três funções do GCS e fornece a história do software dentro de seu ciclo de vida.

A administração de estado fornece os meios pelo qual se organizam todos os elementos isolados que constituem a história do software, determinando quais ocorrências aconteceram ao longo do desenvolvimento de um software, bem como quando elas ocorreram. O processo da administração de estado compreende três processos distintos: coleta, armazenamento e apresentação dos dados. Este processo pode ser visualizado na Figura 2.5.

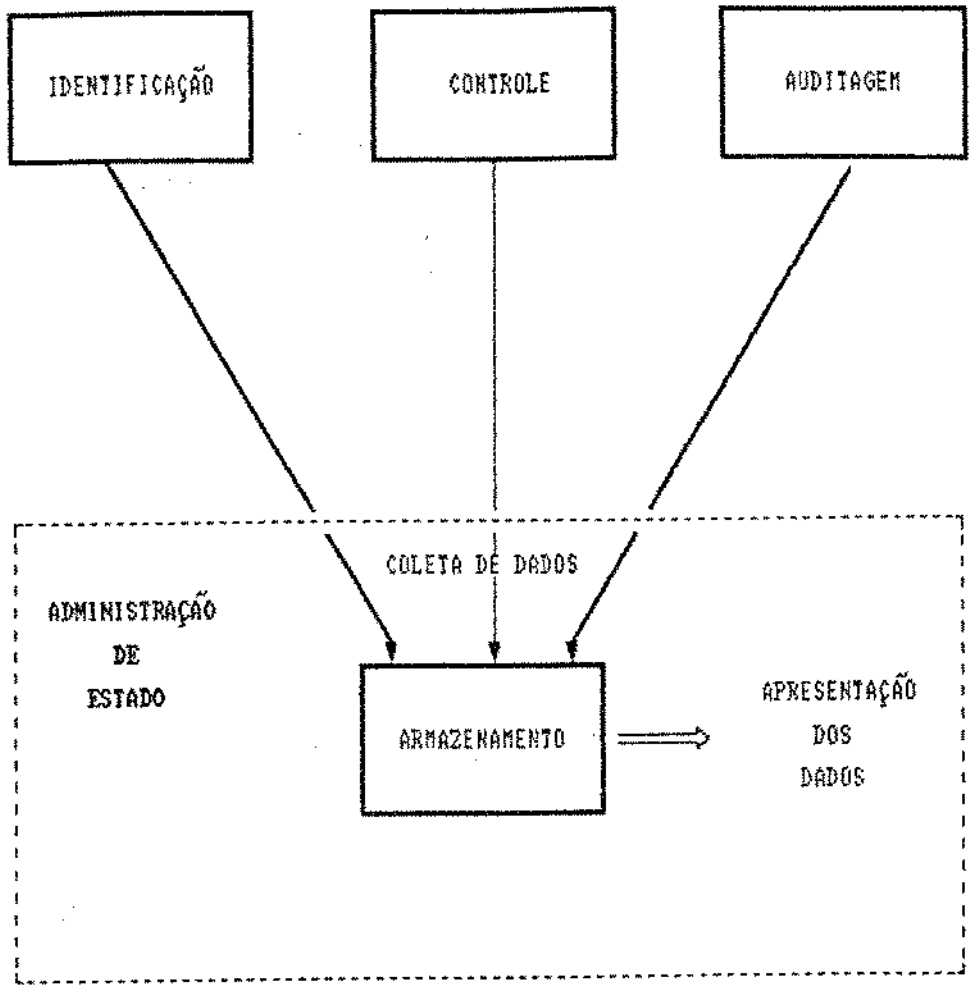


Figura 2.5 - Administração de Estado da Configuração

A coleta de dados é a função primordial deste processo pois deve registrar as ocorrências ao longo do desenvolvimento do software.

Esta função basicamente relata dados, tais como:

- a data da geração de cada representação de uma configuração básica e de suas atualizações;
- a data da geração de cada ICS;
- descrição textual de cada ICS;
- estado da PMS (aprovada, esperando uma ação, rejeitada);

- descrição textual de cada PMS;

- estado da documentação técnica e administrativa de cada configuração básica (p. ex. plano de teste a ser executado em uma configuração básica);

- deficiências descobertas em uma configuração básica durante a auditagem.

A função de armazenamento de dados é alimentada com os dados coletados pela administração de estado, dados estes adquiridos durante o desenvolvimento de um projeto.

A apresentação dos dados representa o processo de realimentação da administração de estado para as demais funções do GCS e pode ser feito através de diversos relatórios.

Exemplos destes relatórios são:

- atas de reuniões do CCC (para documentar as diversas atividades deste corpo);

- relatório de estado da configuração básica, que deve ocorrer periodicamente (de preferência, com a mesma frequência das reuniões do CCC, para permitir sua análise);

- estados atualizados das PMSs;

- quais ICSs em um novo projeto são dependentes da aplicação em um projeto "X" semelhante, e quão semelhantes são? Quais são totalmente independentes?

2.3 - O GCS ao Longo de um Ciclo de Vida

Os conceitos apresentados podem ser melhor entendidos ao longo do ciclo de vida de um projeto de software. A seguir é apresentado um ciclo de vida bem como a atuação do GCS sobre ele.

Existem vários modelos de ciclo de vida em uso e descritos na literatura, sendo que as fases específicas e os nomes variam de um modelo para outro.

Um ciclo de vida como o adotado por Rook [23], permite-nos

melhor expressar a atuação de GCS sobre o mesmo. Este ciclo de vida está apresentado na Figura 2.6 e contém as seguintes fases:

- início do projeto;
- especificação de requisitos;
- projeto estrutural;
- projeto detalhado;
- codificação e teste de unidades;
- integração e teste;
- teste de aceitação;
- operação e manutenção;
- produto descartado.

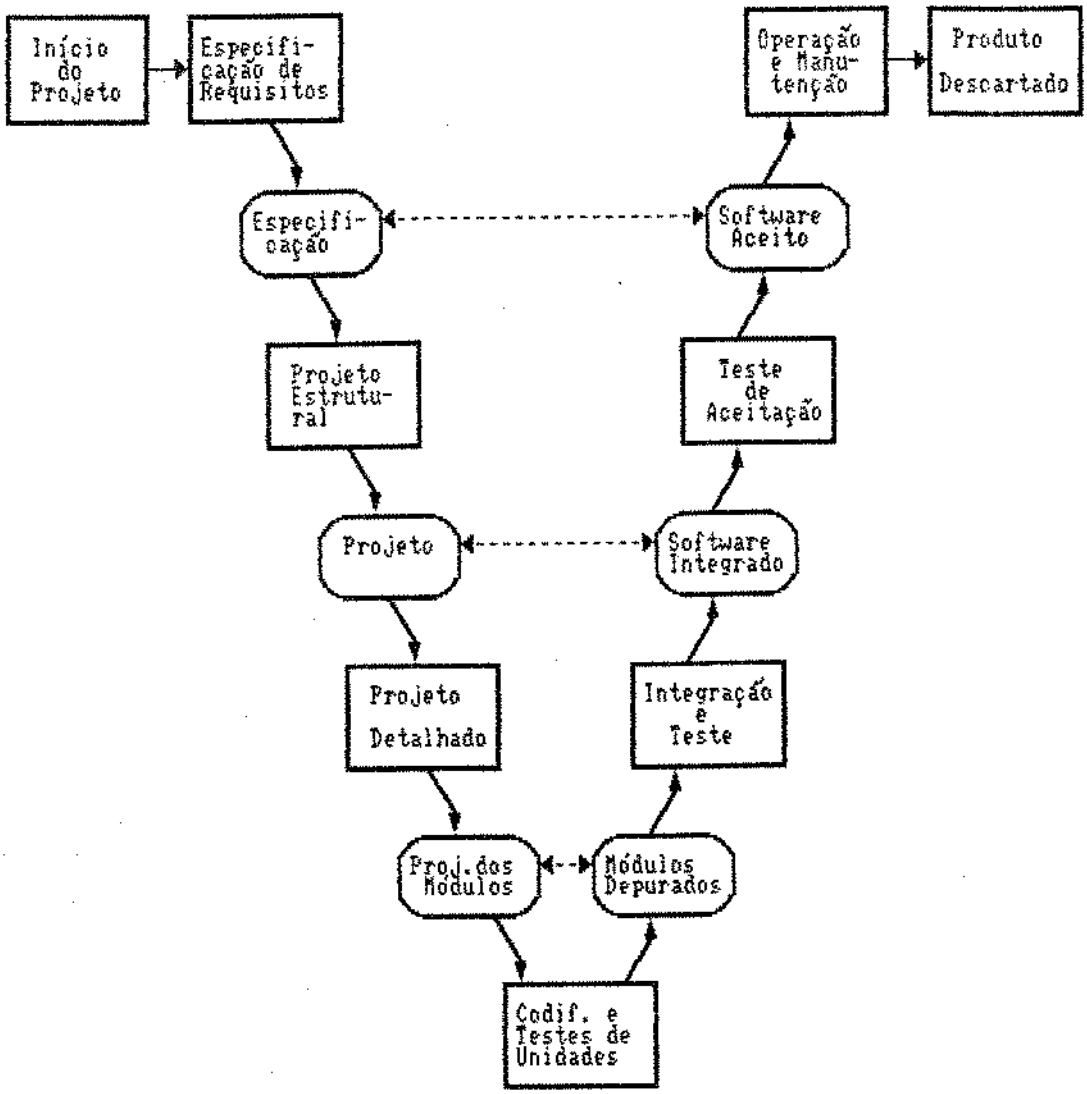


Figura 2.6 - Um Ciclo de Vida de Software

Cada fase de desenvolvimento é definida em termos de suas saídas, ou seja, do produto gerado. Os produtos de cada fase representam os pontos ao longo do desenvolvimento onde existe uma clara mudança na ênfase, onde uma definição do produto emergente é estabelecida e é usada como a base para a próxima definição derivada. Desta forma, estes produtos são os marcos naturais do progresso do desenvolvimento e oferecem uma visibilidade objetiva no progresso alcançado, constituindo-se nas configurações básicas relacionadas com a fase que a geraram.

O término de cada fase é determinado pela conclusão satisfatória do processo de auditagem. Estes produtos então formam a configuração básica para que a próxima fase possa ser desenvolvida. Os produtos da próxima fase são então medidos e verificados em relação às configurações básicas anteriores antes deles se constituirem uma nova configuração básica. Desta forma, um projeto é construído através da geração de configurações básicas sucessivas.

Na Figura 2.6, este processo é apresentado na forma de um diagrama em "V". Neste diagrama as caixas retangulares representam as fases do ciclo de vida e as formas ovaladas representam as configurações básicas. A forma do diagrama mostra a simetria entre a decomposição sucessiva do projeto com a construção do produto por estágios sucessivos de integração e testes. Cada fase do projeto é validada em relação à configuração básica anterior. Cada fase de integração é validada em relação à correspondente configuração básica de projeto ou especificação do outro lado do diagrama.

A seguir são listadas as configurações básicas geradas por cada uma das fases do ciclo de vida de desenvolvimento do software:

- início do projeto: uma arquitetura de sistema validada, baseada em um estudo de projeto com alocações básicas de hardware e software, e uma concepção de operação incluindo alocações básicas de homem-máquina. Um plano de projeto em alto nível, com marcos, recursos, responsabilidades, cronogramas e atividades macros. Padrões e procedimentos definidos.

- fase de especificação de requisitos: uma especificação completa e validada das funções, interfaces e desempenho requerido para o produto de software. Um plano de projeto detalhado.

- fase de projeto estrutural: uma especificação completa e verificada de toda a arquitetura de software e hardware, estrutura de controle e estrutura de dados para o produto de software, e outros componentes necessários tais como esboços de

manual de usuário e planos de testes.

- fase de projeto detalhado: uma especificação completa e verificada da estrutura de controle, estrutura de dados, relações de interface, algoritmos chaves e um esboço para cada componente do programa.

- fase de codificação e testes de unidades: um conjunto completo e verificado de componentes de programas.

- fase de integração e teste: um produto de software totalmente operacional.

- fase de teste de aceitação do software: um produto de software aceito pelo usuário.

- fase de manutenção: uma atualização totalmente operacional do produto de software. Este objetivo deve ser repetido para cada atualização seguindo a sequência completa de desenvolvimento.

- produto descartado: uma transição clara das funções executadas pelo produto para o seu substituto sucessor (se houver).

A realização perfeita de um produto de software requer um controle rígido sobre todas as representações de software ao longo de seu ciclo de vida. É inevitável que as especificações estarão sujeitas a contínuas pressões para mudanças ao longo do ciclo de vida do produto, para corrigir erros, introduzir melhorias e atender a novos requisitos do mercado. A tarefa do GCS é a de prevenir o caos de mudanças incontroladas.

A saída de cada fase de desenvolvimento deve ser verificada e validada em relação às configurações básicas relevantes anteriores. O GCS garante que todas as correções necessárias são introduzidas antes de se estabelecer uma configuração básica e que somente definições atualizadas de configurações básicas são usadas pelas fases subsequentes.

Uma vez que uma configuração básica tenha sido formalmente estabelecida, seu conteúdo somente poderá ser mudado através da operação do processo de controle de configuração de software. Isto tem as seguintes vantagens:

- nenhuma alteração é feita sem o consentimento de todas as partes interessadas;

- a complexidade para efetuar uma mudança tende a estabilizar o produto;

- existe sempre disponível uma versão definitiva do produto, ou de qualquer dos produtos intermediários controlados (configurações básicas).

2.4 - A Automatização do GCS

O GCS envolve tarefas técnicas e administrativas, e é usado como uma forma de gerenciar tarefas complexas. Os procedimentos manuais que têm sido usados para executar o GCS têm se mostrados enfadonhos e frequentemente pouco confiáveis. Automatizar o GCS parece ser uma proposta que garante que estes procedimentos se tornem mais eficientes, efetivos e confiáveis.

A metodologia de GCS em sua concepção é muito mais rica e profunda do que a apresentada nas seções anteriores deste capítulo. Como o objetivo principal deste trabalho é o de automatizar esta metodologia, principalmente sob o enfoque de uma base de dados que suporte os conceitos de controle de versões e de configurações, foram abordados mais profundamente somente os pontos que nos parecem relevantes para este objetivo.

A disciplina GCS está dando os primeiros passos rumo à automatização. Os requisitos necessários à automatização desta tarefa ainda não estão bem compreendidos tanto por quem desenvolve ferramentas desta espécie, como para os usuários destas ferrsamentas. Evidências da falta de entendimento podem ser observados em bibliografias referentes a esta disciplina. Definições de termos apropriados são extremamente generalizadas ou nem existem, e discussões de automatizações efetivas estão ainda em fase de projeto.

Talvez um dos grandes problemas encontrados quanto à automatização da disciplina GCS ao longo de um ciclo de vida, seja o fator conceituação do que representa o software. Uma grande parcela dos esforços dispendidos na automatização do GCS caracteriza o software como programas de computador. Desta forma, geralmente o controle de versões e de configurações restringe-se somente à fase de implementação e de manutenção do código gerado. Um exemplo bem característico deste enfoque pode ser visto em [25] onde se caracteriza o GCS para programas de computador e tenta-se transportar a filosofia de GCS de programas para controlar também a documentação necessária para gerar tais programas.

Neste trabalho, por termos adotado uma definição de software mais abrangente, a disciplina GCS cobre todas as fases do ciclo de vida indistintamente, ou seja, além de se preocupar com o controle do código correspondente à programas, preocupa-se também em controlar todos os documentos necessários ou produzidos para a geração desses programas.

O gerenciamento de configuração é frequentemente confundido com o sub-problema de configurar o software, que embora também seja importante é de valor limitado diante do propósito real desta disciplina. O gerenciamento de configuração deveria começar quando as primeiras palavras dos primeiros documentos que documentam o projeto são escritas e continuar ao longo de todo o seu ciclo de vida.

Esta falta de conhecimento tem resultado em uma escassez de ferramentas adequadas para o efetivo gerenciamento da configuração de software.

A bibliografia examinada mostra que a disciplina GCS tem sido bastante estudada pelo Departamento de Defesa e pelas Forças Armadas dos EUA, uma vez que os seus sistemas automatizados exigem elevadíssimo grau de confiabilidade, aliado ao alto custo do material empregado e ao seu significado no contexto do esquema de defesa norte-americano. Diante de tal fato, nota-se que são estas as organizações que detêm a maior experiência na área, e o aprimoramento de seus documentos reflete um efetivo emprego de GCS [6].

No Brasil, o GCS ainda não ocupa uma posição relevante, podendo até concluir-se que não é uma disciplina conhecida. Um trabalho sobre o assunto desenvolvido no Brasil é uma tese de mestrado realizada na PUC/RJ em 1981 [13]. Este trabalho discorre sobre o tema visto sob a ótica da aplicação manual da disciplina.

2.5 - Ferramentas e Ambientes que Suportam a Automatização do GCS

Uma publicação de 1984 que auxilia na classificação de ferramentas disponíveis na área de GCS é o "The STARTS Guide" (Software Tools for Application to large Real Time Systems) [26].

Segundo [26], as ferramentas que melhor expressam a funcionalidade de GCS não estão ainda maduras (por exemplo, Saviour, PCS, Perspective), enquanto as ferramentas mais maduras (MAKE, SCCS, RCS) têm uma funcionalidade limitada.

Tanto o Saviour, como o PCS e o Perspective apresentam em comum o aspecto de armazenamento das informações em uma base de dados.

O Saviour é uma ferramenta que fornece um conjunto integrado de facilidades para controle de versões, controle de acesso, préprocessamento e construção de sistemas.

O Perspective é um ambiente integrado de suporte a programação que suporta o desenvolvimento de programas escritos em Pascal. As facilidades de GCS para o ambiente referem-se à necessidade de fornecer um compartilhamento disciplinado de componentes de programas dentro de uma equipe de desenvolvimento.

O PCS é um subconjunto de ferramentas de software, parte de um conjunto de ferramentas que formam o BATES AUTOMATED PROGRAMMING SUPPORT ENVIRONMENT (BAPSE). As ferramentas que compreendem o PCS estão relacionadas com: Controle de Projeto, Controle e Produção de Documentação e Gerenciamento de Configuração.

O MAKE [11] aceita descrições de configurações de sistemas e pode automaticamente construir o sistema a partir destas descrições. Ferramentas tais como o Source Code Control System (SCCS) [22] e o Revision Control System (RCS)[28], [29], permitem manter todas as alterações efetuadas em um arquivo texto. O problema chave, não abordado por nenhuma dessas ferramentas mencionadas, é o de manter a integridade da configuração do sistema.

2.5.1 - O GCS Suportado por Base de Dados em Ambientes de Engenharia de Software

A consideração da funcionalidade que é desejada em ferramentas de gerenciamento de configuração leva-nos a concluir que tal funcionalidade não pode ser conseguida efetivamente sem um ambiente apropriado, suportado por uma base de dados. Uma base de dados também oferece a melhor maneira de agrupar as ferramentas de gerenciamento de configuração com as demais ferramentas que cobrem as fases do ciclo de vida, de forma a fornecer um ambiente unificado para o controle do projeto como um todo.

O GCS pode ser exercido dentro de um ambiente de desenvolvimento de software automatizado. Ao se modelar e construir ferramentas para automatizar metodologias para o desenvolvimento de software pode-se, então, modelar os dados de tal forma que estas metodologias fiquem subordinadas àquelas ferramentas e práticas associadas com o gerenciamento de

configuração do produto de software.

O modelamento do GCS para que uma base de dados possa suportá-lo em ambientes de engenharia de software são raros de serem vistos na bibliografia disponível. O modelamento proposto por [18] usa o Modelo Entidade-Relacionamento (ME-R) [8]. Apesar de apresentar um enfoque genérico, de forma a cobrir todas as fases do ciclo de vida, de certa forma o modelamento fixa um tipo de ciclo de vida específico; os conjuntos de entidades refletem os vários níveis de abstrações que eles representam. Um conjunto de entidades "sistema" caracteriza uma família de produtos de software, sendo que um "produto" é um membro do sistema com capacidades funcionais bem definidas. Um "subproduto" é parte de um sistema e possui um papel funcional prédefinido durante o ciclo de vida do sistema. Em um nível mais baixo, "versões" de subprodutos são consideradas; uma "configuração" representa o conjunto coordenado de versões de um produto. Desta forma, os relacionamentos entre os conjuntos de entidades do modelamento referem-se aos elos entre os diferentes níveis de abstração, relacionando a ordem cronológica do processo de desenvolvimento de um sistema.

O enfoque dado no modelamento de Lockemann [18] difere bastante daquele adotado neste trabalho, feito segundo o MRO (Modelo de Representação de Objetos), descrito no Capítulo 3, e no qual procurou-se utilizar ao máximo as características disponíveis deste modelo.

2.5.2 - Ambientes Automatizados que Suportam o GCS

Nota-se que atualmente vários ambientes de desenvolvimento de software têm se preocupado em automatizar o GCS. A seguir são relacionados alguns desses ambientes, bem como é caracterizado o nível de automatização atingido para cada um destes ambientes. Logicamente, esta coletânea não é a mais completa possível, mas consegue mostrar o crescimento da importância desta disciplina no âmbito de ambientes automatizados de engenharia de software.

Todos os ambientes e sistemas vistos apresentam uma característica comum que é a de armazenamento das informações centralizadas em uma única base de dados.

Vale ressaltar que estes ambientes e sistemas apresentam características e enfoques diferentes para a resolução do mesmo problema. Neste trabalho, estes ambientes e sistemas são vistos somente sob a ótica de GCS e assim classificados. Esta

classificação subdivide os ambientes e sistemas em duas categorias: aqueles que suportam o controle de versões e de configurações de programas e aqueles que apresentam uma abordagem de GCS mais genérica.

Ambientes que Suportam Controle de Versões e de Configurações de Programas

Podemos relacionar como sistemas que se preocupam basicamente com o controle de versões e de configurações dos programas gerados, fornecimento de históricos completos das versões geradas dos códigos fontes e reconstrução de um sistema de forma automática a partir das novas versões geradas:

- DSEE Configuration Management System (CM), que é um componente DSEE (DOMAIN Software Engineering Environment) [17] para exercer a atividade de GCS;

- Adele [3], [9], [10];

- SVCE (System Version Control Environment) [14], que é um componente do ambiente Gandalf;

- Mosaix [27], que é uma ferramenta executada sob o Unix e

- PDE - Project Development Environment [19], que é uma abordagem de GCS que capacita o conjunto de ferramentas de software fornecidas pelo Unix a suportar uma variedade de procedimentos do GCS.

Ambientes que Apresentam um Enfoque mais Genérico de GCS

Como ambientes e sistemas que apresentam um enfoque mais genérico por se caracterizarem como meta-sistemas e permitem cobrir todas as fases do ciclo de vida podemos relacionar:

- Eclipse [1], que possui um esquema que permite a rastreabilidade de objetos ao longo do ciclo de vida, de forma a permitir a reconstrução de qualquer objeto a partir de seus precursores. Este recurso fornece facilidades para a tarefa de análise de impacto das alterações envolvidas em um determinado objeto, bem como apresenta o conceito de evolução de objetos, bastante importante para o processo da auditagem e

- ALMA (Atelier Logiciel sur Machine Abstraite) [16], que está iniciando o projeto preliminar de um sistema de controle de versões. Dentre os recursos atuais disponíveis encontram-se: o controle de acesso às informações da base de dados e uma ferramenta que atua como um filtro durante a atualização dos

dados das áreas de trabalho para a base de dados do ALMA. Desta forma, algumas características da função de auditagem já se encontram implementadas.

Um Sistema que Automatiza Todos os Procedimentos do GCS

Como pode ser visto dentre os ambientes de desenvolvimento de software, poucos avanços têm sido alcançados na área de GCS quando se refere à sua efetiva automatização.

O CMS - Configuration Management System [35], desenvolvido pela General Electric foi construído com o propósito específico de automatizar os procedimentos da disciplina GCS. Este sistema foi o único encontrado na bibliografia examinada que apresenta a disciplina GCS automatizada. Este sistema quando visto somente sob a ótica de GCS é completo e possui uma base de dados que centraliza as informações geradas somente para o propósito de efetuar o GCS. No entanto, apresenta uma deficiência, pois outras informações do projeto também deveriam estar na base de dados para que este sistema se constituísse em um ambiente de desenvolvimento de software que abrangesse as fases do ciclo de vida.

Diante dos ambientes e sistema vistos acima, nota-se que somente o sistema CMS apresenta uma automatização mais concreta das atividades do GCS. Mas, o CMS ainda apresenta uma deficiência no que tange à coleta de informações para o GCS, pois uma forma segura de se obter as informações seria em um ambiente de desenvolvimento de software, no qual outras ferramentas se incumbiriam de desenvolver o software alimentando a base de dados com as informações necessárias de forma automática, para a atuação do GCS.

A proposta deste trabalho é a de automatizar o GCS de forma completa e efetiva no ambiente SIPS. Este propósito é possível uma vez que o SIPS fornece uma série de características favoráveis providas por uma base de dados centralizada e única, que é utilizada por várias ferramentas para desenvolvimento de sistemas de software cobrindo potencialmente todas as fases do ciclo de vida.

CAPÍTULO 3

SIPS - SISTEMA INTEGRADO PARA A PRODUÇÃO DE SOFTWARE

3.1 - O que é o SIPS

Neste capítulo é feita uma breve apresentação do ambiente de apoio ao desenvolvimento de software denominado Sistema Integrado para Produção de Software - SIPS [31], [34], em desenvolvimento no Instituto de Automação do Centro Tecnológico para Informática, mostrando principalmente as características que são necessárias para o entendimento dos capítulos seguintes que tratam da automatização do GCS dentro do SIPS.

O SIPS é um sistema expansível para apoio à produção de software. Seu objetivo é dar suporte ao desenvolvimento integrado de software, abrangendo todas as fases do ciclo de vida, através de ferramentas automatizadas que contribuam para uma maior agilidade no processo de desenvolvimento de sistemas, aumento de produtividade e melhor qualidade do software produzido.

A filosofia de desenvolvimento do SIPS consiste na construção de ferramentas independentes entre si, porém com funções concatenadas, de maneira a atender as necessidades de desenvolvimento de um projeto. A integração entre as ferramentas é conseguida através da padronização das informações armazenadas e manipuladas pelas ferramentas, as quais se apóiam em um sistema de gerenciamento de bases de dados especialmente construído. Além disso, todas as ferramentas empregam uma mesma interface de comunicação com o usuário, mantendo assim a coerência entre os comandos das várias ferramentas, o que facilita a interação com o usuário.

A existência do núcleo comum de gerenciamento de informações para todas as ferramentas e a interface única de comunicação com o usuário fazem do SIPS um sistema integrado para a produção de software; ao mesmo tempo, a independência entre as ferramentas permite a incorporação gradativa de novas ferramentas, tornando-o expansível e capaz de atender novas necessidades de produção de

software.

Para que uma metodologia seja automatizada, ela deve ser representada na ferramenta que a automatiza e, para isso, ela deve ser formalizada. O processo de formalização de uma metodologia impõe uma definição rígida do escopo de abrangência dos casos tratados pela metodologia, bem como da maneira como ela é utilizada. No SIPS, metodologias podem ainda ser integradas entre si para propiciar o suporte integrado às várias fases do processo de produção de software.

O SIPS pode ser considerado como sendo composto de dois subsistemas: **SIPS-META**, que permite a definição de Linguagens de Descrição de Metodologias pelo Especialista de Metodologia; e **SIPS-SISTEMA**, que permite o uso do SIPS pelos projetistas empregando as metodologias já integradas para o desenvolvimento de sistemas de software, aqui denominados sistemas alvos.

O SIPS armazena todas as informações do sistema alvo em uma única base de dados, usando o Modelo de Representação de Objetos descrito na Seção 3.2, independentemente das metodologias empregadas para descrever estes sistemas. Esta base de dados é denominada Base Sistema.

As descrições das metodologias são mantidas em uma outra base de dados, a Base Meta, cujo conteúdo define como a Base Sistema deve ser usada. A Base Meta é inicializada e mantida pelos Especialistas em Metodologias. Estes Especialistas analisam as metodologias a serem integradas ao SIPS, descrevendo-as com uma Meta Linguagem específica do SIPS. A descrição é interpretada pelo Interpretador da Meta Linguagem, que armazena essas informações na Base Meta. O procedimento para modelamento de uma metodologia e sua incorporação ao SIPS é descrito na Seção 3.3.

A Figura 3.1 mostra a arquitetura do SIPS-SISTEMA onde aparecem os vários componentes do ambiente. As ferramentas desenvolvidas para o SIPS são integradas ao ambiente. Qualquer conjunto de ferramentas desenvolvidas para o SIPS pode ter a aparência de um ambiente integrado, obtida através do supervisor. O supervisor é responsável pelo monitoramento e pela ativação das ferramentas com as quais o usuário interage. As ferramentas acessam apenas no modo leitura a Base Meta, a qual contém a definição de como a Base Sistema deve ser usada; a Base Sistema é acessada para armazenar descrições de sistemas alvo. Os acessos às bases de dados do SIPS são feitos através de primitivas que constituem o Gerenciador da Base Meta e Gerenciador da Base Sistema. Esses acessos às bases de dados são monitorados pelo Gerenciador de Projeto e de Configuração de Software. As ferramentas possuem uma interface padrão de comunicação com o usuário, que é obtida através das primitivas que constituem o

Gerenciador da Interface Homem/Máquina.

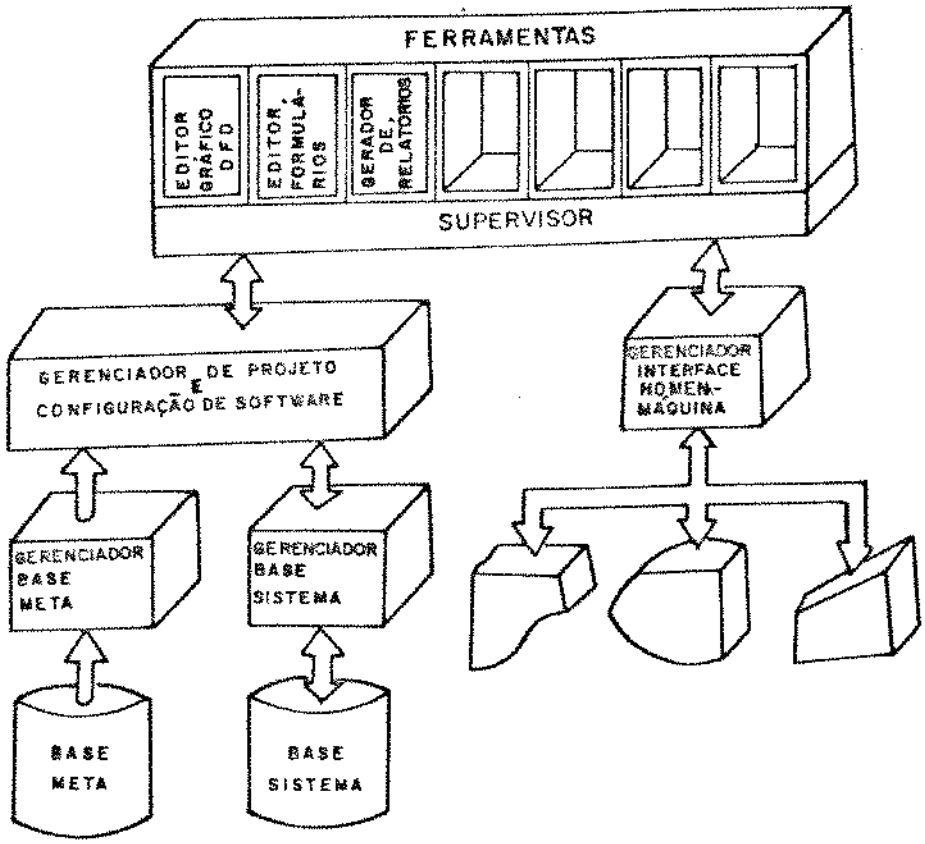


Figura 3.1 - Arquitetura do SIPS-SISTEMA

3.2 - MRO (Modelo de Representação de Objetos)

O modelo de dados em que o SGBD do SIPS se apóia é o MRO, que procura modelar as situações do mundo real, representando as características desejáveis dos objetos que existem no mundo real.

Nesta seção são apresentados somente os conceitos necessários para o desenvolvimento deste trabalho. Mais informações sobre o modelo podem ser obtidas em [32] e [33].

A fundamentação matemática do MRO é a teoria dos grafos, por ser mais adequada para modelos de dados voltados à representação semântica intensa.

Nesse modelo, os nós de um dígrafo representam os **objetos** que existem no mundo real e os arcos representam as associações que existem entre eles, as quais são denominadas **relacionamentos.** Os objetos são identificados através de nomes e são representados através de códigos (identificadores internos).

A categorização dos nós de um dígrafo classifica os objetos através de um **tipo** que é atribuído a cada objeto. O mesmo é feito quanto aos relacionamentos entre os objetos, aos quais se atribuem igualmente **tipos de relacionamentos.** Assim, os objetos e os relacionamentos no MRO são classificados segundo seus respectivos tipos.

Cada relacionamento no MRO indica que os objetos estão associados um ao outro, ou seja, no dígrafo correspondente devem existir dois arcos, um em cada sentido. Cada arco representa um **relacionamento oposto** ao representado pelo seu par; no MRO todo relacionamento tem o seu oposto correspondente.

Representam-se também as **propriedades** dos objetos que é uma denominação genérica dada aos tipos abstratos **comentário, atributo** e **sinônimo** (nome alternativo de objetos), bem como os atributos de relacionamentos.

Aos objetos, além das propriedades, podem ainda ser associados alguns atributos comuns a todos os objetos da base de dados denominados **atributos intrínsecos.** Como atributos intrínsecos de um objeto podemos ter, por exemplo, o nome do objeto e a data de sua criação.

Os objetos podem ser acessados diretamente através de seu identificador; porém, relacionamentos e propriedades somente são acessados através das associações que mantêm com os objetos. Além disso, os atributos de relacionamento somente são acessados através dos relacionamentos aos quais estão associados, os quais por sua vez devem ser acessados através dos objetos que, esses sim, podem ser diretamente acessados (no entanto objetos podem também ser acessados através dos relacionamentos que mantêm com outros objetos).

A Figura 3.2 mostra uma ilustração da representação através do MRO de dois tipos de objetos (autor e livro) que se relacionam através do relacionamento escreve, cujo relacionamento oposto é escrito-por. Autor possui os tipos de atributo endereço-autor e número-livros-publicados e livro possui os tipos de atributo data-lançamento e nome-editora.

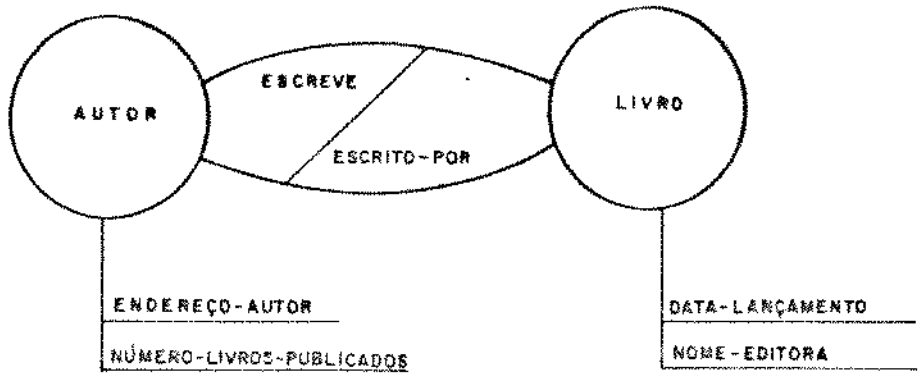


Figura 3.2 - Representação de Objetos usando o MRO

3.3 - O Modelamento de uma Metodologia Usando o MRO e sua Incorporação ao SIPS

A incorporação de uma metodologia ao SIPS é feita por um Especialista da Metodologia, que tenha dela um profundo conhecimento e, além disso, que tenha bom conhecimento de como essa tarefa é realizada com o SIPS.

Para que uma metodologia possa ser suportada pelo SIPS, esta deve ser modelada no MRO e essa modelagem submetida a um interpretador de linguagem de descrição de metodologias do SIPS, denominado Interpretador Meta. A descrição obtida deste interpretador é armazenada na Base Meta.

O processo de modelagem de uma metodologia pode ser resumidamente descrito pelo seguinte procedimento:

- Identificar as palavras chaves da metodologia. Para isso, poder-se-ia analisar um texto que descreva a metodologia e dele extrair todas as palavras que se referem a conceitos da metodologia;

- Separar essas palavras chaves em tipos de objetos e tipos de associações. As palavras que representam elementos da metodologia devem ser identificadas e separadas em tipos de objetos. As palavras que correspondem a associações são as que identificam propriedades de objetos e relacionamentos entre objetos;

- Analisar os tipos de associações. Devem ser classificadas em: tipos de relacionamentos - aquelas que associam dois ou mais objetos; tipos de atributos - aquelas que se referem a valores que se associam a tipos de objetos ou a tipos de relacionamentos; tipos de sinônimos - aquelas que se referem a tipos de identificadores alternativos para um mesmo objeto e tipos de comentários - aquelas que se referem a tipos de textos que podem ser associados a objetos;

- Identificar as associações entre objetos e relacionamentos. A partir desse passo, deve ser identificado como os vários tipos podem se associar, armazenando essas informações em tabelas. Neste passo, identificam-se os tipos de relacionamentos que podem ser associados a cada tipo de objeto;

- Identificar os tipos de relacionamentos opostos a cada relacionamento. Como todo tipo de relacionamento tem um tipo de relacionamento oposto, este tem que ser identificado. A seguir, completam-se as tabelas de quais relacionamentos são associados a cada objeto, usando a confrontação com os relacionamentos opostos;

- Identificar as associações entre atributos e objetos e entre atributos e relacionamentos;

- Identificar as associações entre objetos e comentários e entre objetos e sinônimos;

- Preparar a descrição da modelagem obtida na Linguagem de Descrição de Metodologias do SIPS;

- Submeter esta descrição ao Interpretador Meta para que ela possa ser armazenada na Base Meta; a descrição armazenada será usada a partir de então para a descrição dos sistemas do usuário.

Para exemplificar como é feito o modelamento de uma metodologia usando o MRO e sua incorporação ao SIPS, utilizaremos a metodologia Análise Estruturada de Sistemas [12]. Esta metodologia apresenta os seguintes componentes: Função, Fluxo de Dados, Arquivo Lógico, Entidade Externa, Estrutura de Dados e Elemento de Dados; sendo que os quatro primeiros componentes relacionados aparecem representados graficamente em um diagrama denominado Diagrama de Fluxo de Dados (DFD). Esta metodologia contempla a fase de especificação funcional de um sistema consistindo do DFD representando graficamente a interação entre os componentes e do dicionário de dados que complementa as informações dos componentes gráficos detalhando-os através de descrições e das estruturas e elementos dos dados lógicos que os compõem.

A Figura 3.4 apresenta um DFD resumido de um sistema que aceitará pedidos de livros, fará a verificação no arquivo de disponíveis, verificará, em outro arquivo, se o crédito do

cliente é bom e fará com que o livro (ou livros) pedido seja remetido acompanhado de fatura. No DFD, são utilizados os quatro componentes gráficos da metodologia, ilustrados na Figura 3.3.

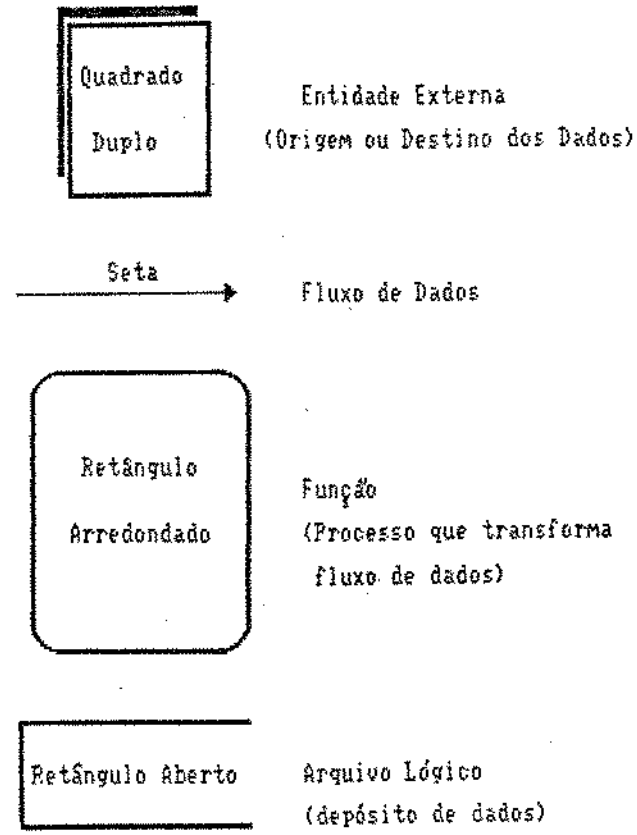


Figura 3.3 - Componentes Gráficos do DFD

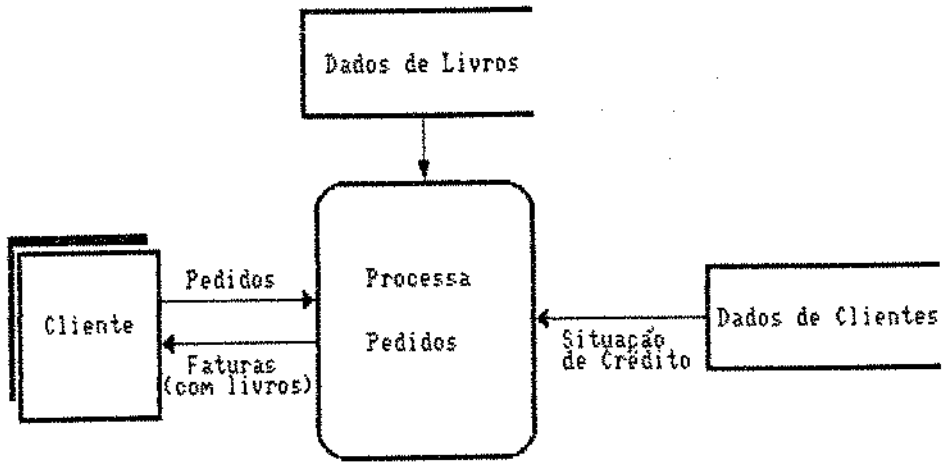


Figura 3.4 - Diagrama de Fluxo de Dados

Na Figura 3.5 é apresentado um Diagrama de Representação de Objetos que mostra um modelamento simplificado da metodologia Análise Estruturada de Sistemas. Para esse modelamento identificaram-se os Tipos de Objetos : DFD, FUNÇÃO, FLUXO-DE-DADOS, ARQUIVO-LÓGICO, ENTIDADE-EXTERNA, ESTRUTURA-DE-DADOS E ELEMENTO-DE-DADOS, os Tipos de Relacionamentos GERA oposto a GERADO-POR, RECEBE oposto a RECEBIDO-POR, etc., com as respectivas associações, tal como pode ser visto nessa figura.

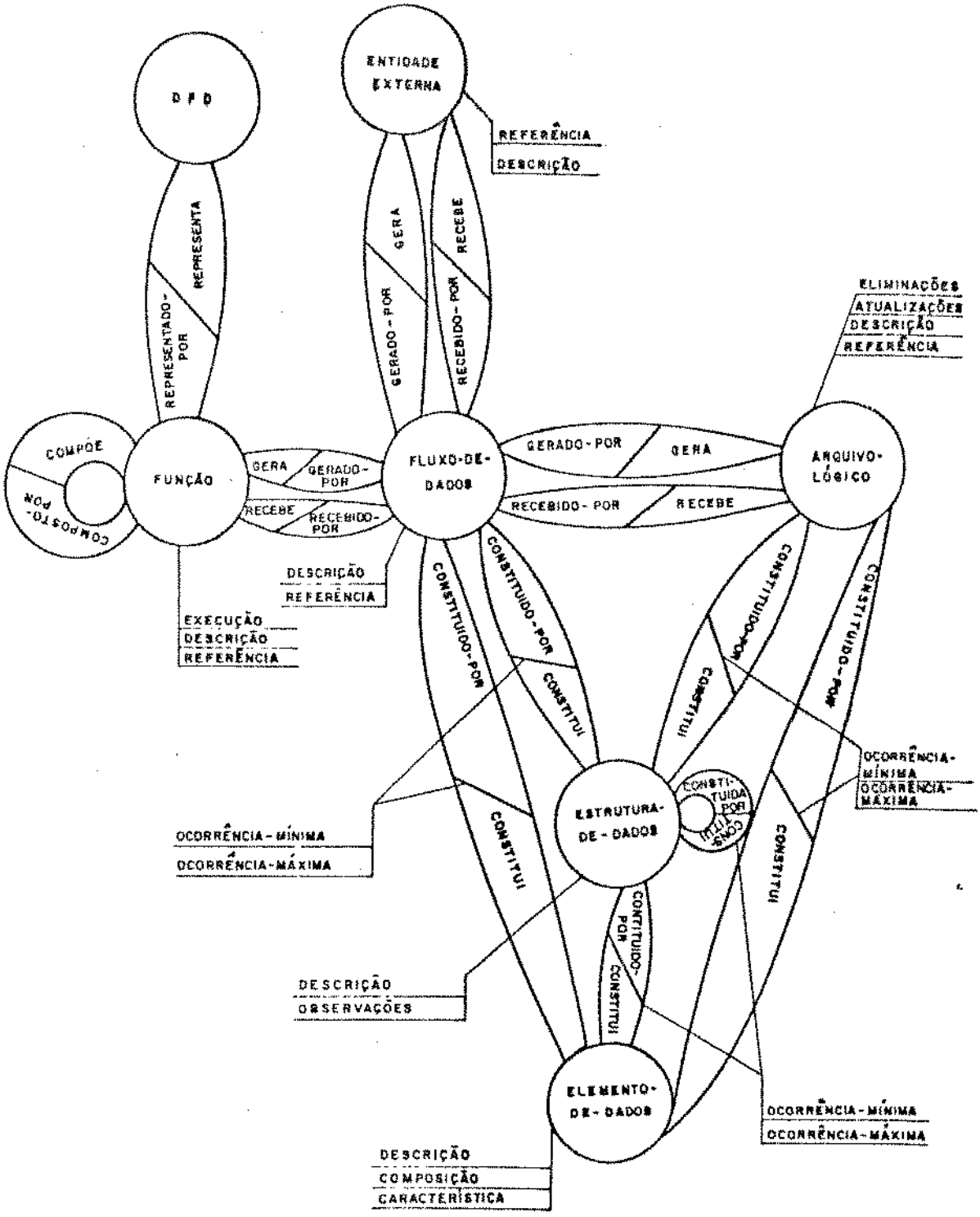


Figura 3.5 - Modelamento da Análise Estruturada de Sistemas usando o MRO

3.4 - Composição da Base de Dados do SIPS

O núcleo do SIPS consiste de primitivas que permitem às ferramentas acessar as Bases Meta e Sistema. Uma ferramenta pode somente ler os dados da Base Meta, uma vez que ela contém as informações que descrevem a metodologia que será usada para descrever o sistema alvo. A Base Sistema pode ser acessada pelas ferramentas para leitura, inclusão ou alteração de dados do sistema alvo em desenvolvimento.

Pode-se considerar que os seguintes componentes (arquivos) compõem a Base Sistema:

1 - Identificador (Nomes e Sinônimos de objetos),

2 - Objeto (Outros dados do objeto),

3 - Relacionamento,

4 - Atributo de Objeto e Relacionamento,

5 - Comentário e

6 - Classe (ver seção 3.6).

As operações fundamentais definidas sobre os primeiros cinco tipos de componente são as seguintes:

- incluir um novo tipo de componente;
- posicionar em um componente;
- obter dados de um componente posicionado;
- avançar de uma forma pré-determinada através dos dados de um componente;
- retirar dados de um componente;
- retirar todos os dados de um objeto.

Além disso, algumas operações aplicam-se somente a determinados componentes, tais como identificar ou alterar o nome de um objeto, aglutinar dois objetos em um (fundir todas as propriedades e relacionamentos dos dois objetos em um único objeto), etc.

Além das primitivas que manipulam diretamente as informações da Base Sistema, existem primitivas auxiliares que permitem algumas operações adicionais sobre os componentes da Base Sistema. Dentre essas operações existem primitivas que permitem:

- manipular arquivos de seleção. Arquivos de seleção são utilizados para armazenar códigos (identificadores internos) de objetos selecionados pelo projetista. Existem primitivas para criar um arquivo de seleção, abrir um arquivo já criado, fechar um arquivo de seleção, colocar e recuperar um código de objeto, ou colocar todos os objetos de uma classe em um arquivo; e ainda ordenar um arquivo de seleção, fazer a diferença, união ou intersecção entre dois arquivos de seleção;

- armazenar e restaurar uma marca para um dado objeto. Uma marca é uma informação qualquer que o projetista pode guardar na Base Sistema associada a um objeto;

- armazenar e restaurar registros de estados correntes em uma área de memória. Isto se faz necessário quando se quer manipular informações de dois objetos ao mesmo tempo. Para detalhes de Registro de Estado veja Seção 3.5.

- armazenar e restaurar registros de classes correntes em uma área de memória. Isto se faz necessário quando se quer manipular informações contidas em dois conjuntos de objetos constritos diferentes de uma mesma classe hospedeira. Para detalhes de classes veja Seção 3.6.

A implementação de uma ferramenta para o SIPS consiste no uso de primitivas que acessam as Bases Meta e Sistema e primitivas da Interface/Homem-Máquina que constituem o núcleo do SIPS. Dessa forma, nenhuma ferramenta acessa diretamente as informações das bases de dados do SIPS.

O SIPS possui várias metodologias automatizadas. Uma delas é a Análise Estruturada de Sistemas, cujo modelo está mostrado parcialmente na Figura 3.5. Para automatizar esta metodologia dentro do ambiente, as seguintes ferramentas foram implementadas: Editor Gráfico para Diagramas de Fluxo de Dados (DFD) e Editor de Formulários. Com estas ferramentas, os projetistas podem desenvolver sistemas usando a metodologia Análise Estruturada de Sistemas para inserir a descrição dos mesmos na Base Sistema.

Para exemplificar o uso das primitivas do núcleo do SIPS na construção de uma ferramenta, podemos pensar em uma ferramenta Analisador de Completeza para Dicionário de Dados, definida de acordo com a metodologia Análise Estruturada de Sistemas. Este Analisador pode ser visto como uma ferramenta usada para verificar se certas informações estão presentes na Base Sistema. Por exemplo, verificar se em um determinado Diagrama de Fluxo de

Dados existem fluxos desconectados, isto é, fluxos de dados que não estão sendo gerados ou não estão sendo recebidos por alguma função, arquivo lógico ou entidade externa.

Para implementar a ferramenta descrita acima, deve ser escolhido do conjunto de primitivas do núcleo do SIPS, aquela que posiciona objeto do tipo fluxo de dados. Para cada fluxo de dados posicionado deve ser verificado, através de primitivas de obter dados, se os relacionamentos "gerado-por" ou "recebido-por" estão presentes na descrição do sistema alvo. Se um dos dois relacionamentos não estiver presente na descrição, uma mensagem de erro deve ser gerada pela ferramenta. Para obter o próximo fluxo de dados, primitivas de avançar devem ser usadas. Para obter o nome do fluxo de dados, assim como as mensagens de erro, devem ser usadas as primitivas da interface homem/máquina.

Um procedimento para esta ferramenta pode ser escrito como indicado abaixo:

```
Obtém o nome do diagrama de fluxo de dados
Posiciona no primeiro fluxo de dados
Enquanto houver fluxos de dados no diagrama
Faça
    Obtém o relacionamento "gerado-por"
    Se o relacionamento não existe
    Então  Envia uma mensagem de erro
    Obtém o relacionamento "recebido-por"
    Se o relacionamento não existe
    Então Envia uma mensagem de erro
    Avança para o próximo fluxo de dados
```

3.5 - O Registro de Estado

Um conceito importante para o uso das primitivas de acesso a informações da Base Sistema é o de Registro de Estado. O objetivo é o conhecimento preciso de quais dados da base serão afetados em cada operação solicitada.

Sempre que se está posicionado em um dado objeto, pode-se considerar que esse objeto passa a ser o **Objeto Corrente**, sobre o qual as operações subsequentes serão efetuadas, até que se mude o objeto corrente novamente. Para efeito de visualização, pode-se considerar que existe um "apontador" indicando, dentre todos os objetos da base de dados, qual é o corrente.

Porém, não apenas objetos devem ser indicados dessa forma, mas também relacionamentos, sinônimos, comentários, linhas de comentários, atributos de objetos e atributos de relacionamentos. Assim, existem sete apontadores, um para cada um desses elementos, podendo existir então, além de objeto corrente, também um relacionamento corrente, um sinônimo corrente, um comentário corrente, uma linha de comentário corrente, um atributo de objeto corrente e um atributo de relacionamento corrente.

Note-se que os apontadores não são independentes um do outro, uma vez que existe uma hierarquia bem estabelecida entre estes apontadores, a qual será explicada a seguir.

Somente pode ser indicado como relacionamento corrente um dos relacionamentos que tenham o objeto corrente como objeto origem. Dessa maneira, para se ter um relacionamento corrente, é necessário primeiro que se indique qual é o objeto corrente, e a seguir, que se indique qual de seus relacionamentos é o corrente. Se alguma operação alterar o objeto corrente, então deixa de existir um relacionamento corrente, e o apontador correspondente fica invalidado.

O mesmo ocorre com atributo de objeto, comentário e sinônimo, os quais somente podem ter como corrente um atributo, comentário e/ou sinônimo associado ao objeto corrente. Quanto a linhas de comentário, existirá uma linha de comentário corrente sempre que existir um comentário corrente. Se um comentário for reposicionado, a linha corrente será a primeira linha desse comentário, e somente mudará quando for solicitado um avanço na linha de comentário corrente.

Quanto aos atributos de relacionamento, o apontador do atributo de relacionamento corrente deve estar sempre apontando para um dos atributos do relacionamento corrente, e, dessa forma, torna-se inválido, deixando de existir, um atributo de relacionamento corrente sempre que o relacionamento corrente for reposicionado ou se tornar inválido.

Dessa forma fica claro que, para existir um atributo de relacionamento corrente, deve existir um relacionamento corrente. Além disso, para que exista relacionamento corrente, comentário corrente, linha de comentário corrente, sinônimo corrente, ou atributo de objeto corrente, deve existir um objeto corrente.

O conjunto de todos esses sete apontadores constitui a estrutura de informação denominada Registro de Estado. O seu conteúdo não pode ser acessado diretamente por uma ferramenta,

mas apenas indiretamente através de chamadas de primitivas de acesso à Base Sistema. Por sua vez, muitas dessas primitivas atuam dependendo do conteúdo do registro de estado, e por isso é muito importante que o projetista de ferramentas, que esteja usando essas primitivas, conheça precisamente como cada primitiva afeta o seu conteúdo.

3.6 - O Conceito de Classes no SIPS

Quando se procura acessar um objeto armazenado na Base Sistema, se for conhecido o seu código (identificador interno) então o acesso é imediato, uma vez que o código identifica univocamente um objeto entre todos os objetos da Base Sistema. No entanto, quando a identificação é feita pelo nome (ou por um sinônimo), essa identificação não é unívoca, uma vez que pode existir mais de um objeto com um mesmo nome. Para que se possa identificar um objeto pelo seu nome univocamente, é necessário que se especifique o contexto onde esse nome está definido. Essa especificação é feita através da partição dos identificadores existentes na Base Sistema em classes, de maneira que um nome esteja definido somente no domínio de uma classe.

A identificação de objetos pelo seu código é normalmente efetuada pelas ferramentas; porém, o usuário nunca tem acesso direto a esses códigos, sendo que a sua interação com o SIPS sempre se dá através dos nomes e sinônimos dos objetos envolvidos. Assim, é desejável que o modelo de dados utilizado pelo seu gerenciador de base de dados esteja bastante próximo do que o usuário 'sente' que seja a informação que a Base de Dados manipula.

Pode-se dizer que um sistema, quando visto de maneira global, apresenta um pequeno grau de detalhe. Quando o interesse recai em um de seus componentes, o grau de detalhe aumenta, porém toda a informação que então se torna disponível refere-se diretamente àquele componente que está sendo observado. Pode-se então pensar que quando uma ferramenta efetua uma comunicação com o usuário, ela realiza a interpretação dos identificadores dos objetos envolvidos, levando em consideração somente o conjunto de informações que está sendo examinado, e exigindo que qualquer referência a outras informações tenha um teor de identificação maior.

Pode-se dotar o modelo de dados do SIPS dessa capacidade considerando que, quando se manipula a base de dados de um ponto de vista **global**, são poucos os objetos acessíveis. Indicando-se ao sistema que se pretende considerar a seguir um desses

objetos, indica-se uma região da Base de Dados em que se pretende obter ou manipular informações. Dessa forma tornam-se acessíveis novos objetos. Esse procedimento pode ser repetido, indicando-se para consideração um dos novos objetos tornados disponíveis, e dessa forma aumentando-se sucessivamente o nível de detalhe e, correspondentemente, a quantidade de objetos a que se tem acesso diretamente através de seus nomes (ou sinônimos).

Cada vez que se **coloca em consideração** um novo objeto, aumenta-se um nível de detalhe; porém, não são todos os objetos desse nível que se tornam disponíveis, mas apenas aqueles que detalham mais o objeto considerado.

O modelo de dados do SIPS pode então contar com essa capacidade usando-se os conceitos de Classe de Objetos, Objeto Considerado, e Classe Hospedeira.

Uma das classes terá que estar permanentemente disponível e será denominada **Classe Global**. Os objetos da Classe Global são os que estão disponíveis quando se está no nível menos detalhado de acesso ao sistema e permanecem sempre disponíveis. Correspondentemente existe uma Classe Hospedeira Global, a qual contém os tipos de objetos que existem na Classe Global. Dessa forma, todos os objetos dos tipos que pertencem à classe hospedeira global estarão permanentemente disponíveis.

Portanto, o conceito de classes permite que os objetos de uma Base de Dados do SIPS passem a ser acessíveis em blocos, sendo que cada bloco se refere não somente a um nível específico de detalhe de informação, mas também a um determinado aspecto de informação.

Para isso, os objetos de uma Base de Dados devem ser classificados segundo diferentes aspectos, sendo essa classificação feita usando os diferentes Tipos de Objetos definidos na Base Meta. Dessa forma, cada objeto pertence a apenas um bloco acessível, que é chamado de **Classe de Objetos**, que é determinado através do seu tipo de objeto.

Para a existência de Classes de Objetos, é necessário que se divida os tipos de objetos contidos na Base Meta nos vários aspectos de interesse. Pode-se considerar então que o conceito de Classes de Objetos estabelece uma partição no conjunto de Tipos de Objetos da Base Meta, sendo cada partição denominada **Classe Hospedeira**. Os objetos da Base Sistema estarão também particionados segundo os seus tipos, em uma correspondência com a partição que as classes hospedeiras estabelecem sobre os tipos dos objetos na Base Meta.

Os objetos de um mesmo tipo que estão agrupados em uma mesma classe, segundo o foco de interesse do usuário em um dado instante, são acessíveis quando a classe a que pertencem é aberta; porém, num dado instante, somente um conjunto de objetos constritos da Base Sistema pode ser hospedada para cada uma das classes hospedeiras.

Objeto Considerado é um objeto que se tem presentemente disponível e que se indica para consideração, o que faz com que os objetos que o especificam com maior grau de detalhe se tornem também disponíveis.

Quando se considera um objeto, indica-se implicitamente uma Classe de Objetos. Como foi visto, não são todos os objetos da classe hospedeira indicada que se tornam disponíveis, mas apenas os que pertencem ao **Conjunto de Objetos Constritos** pelo objeto indicado para consideração. Assim, cada classe de objetos que existe na Base Sistema é sempre constrito por um único objeto, denominado **Objeto Constritor** daquela classe.

Indicar um objeto para consideração caracteriza uma operação de **Abrir uma Classe de Objetos** para acesso, tornando acessíveis os objetos do Conjunto de Objetos Constritos pelo Objeto Considerado. Essa abertura pode permitir que os objetos se tornem acessíveis para consulta e modificação ou apenas para consulta. A indicação de um objeto para consideração tem por objetivo abrir novas classes de objetos para acesso, e a indicação para consideração de um objeto pode abrir mais de uma classe. Dessa forma, o Conjunto de Objetos Constritos por um objeto pode envolver uma ou mais classes de objetos.

Quando se indica para consideração um objeto cujos objetos constritos pertencem a uma Classe Hospedeira já aberta, esta é fechada, o que faz com que o Conjunto de Objetos Constritos anteriormente deixe de estar acessível; a seguir, a mesma Classe Hospedeira é novamente aberta, agora tornando disponíveis os objetos constritos pelo novo Objeto Considerado. Outros objetos, pertencentes a classes que não aquela focalizada implicitamente pela indicação de um novo objeto para consideração e pela correspondente operação de abertura de classe, não são afetados por essa operação.

Os objetos da Classe Global estarão sempre disponíveis, não sendo possível fechar essa classe.

Se o usuário desejar referir-se a qualquer dos objetos que estão disponíveis em qualquer uma das classes abertas, o nome do

objeto e a indicação de seu tipo serão suficientes para identificá-lo. No entanto, se for necessário se referir a algum objeto pertencente a uma classe não aberta, ou a um objeto pertencente a um conjunto de objetos constritos por um objeto que não o presentemente considerado, então será necessário que se identifique todo o caminho de acesso, desde o ponto de interesse corrente na base de dados até o ponto onde está o objeto a que se quer referir.

É com esses conceitos que o modelo de dados usado pelo gerenciador de base de dados do SIPS passa a permitir que objetos pertencentes a Conjuntos Constritos por Objetos distintos tenham nomes iguais e, mesmo assim possa manter-se a sua identificação inequívoca através da identificação do ponto de interesse do usuário em um dado instante.

A Figura 3.6 ilustra o conceito de classes descrito acima. Fazendo uma analogia com países, seus estados e suas cidades, a classe hospedeira global, neste exemplo, é composta de países, que por sua vez são compostos de estados que são compostos por cidades. Os países estão sempre acessíveis e não existem dois países com o mesmo nome. Dentro da classe hospedeira global, existe a classe hospedeira país, que contém os vários estados; e dentro da classe hospedeira país, existe a classe hospedeira estado, que contém as várias cidades. Um estado não possui duas cidades com o mesmo nome, mas pode haver dentro de um país mais de uma cidade com o mesmo nome mas em estados diferentes. Então, ao identificarmos o contexto de um determinado país e de um determinado estado, quando estivermos referenciando uma cidade esta pertence ao contexto especificado.

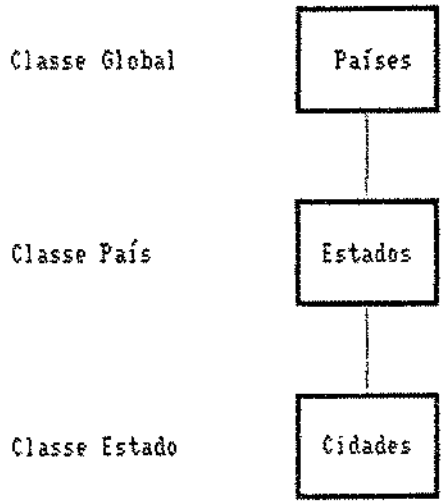
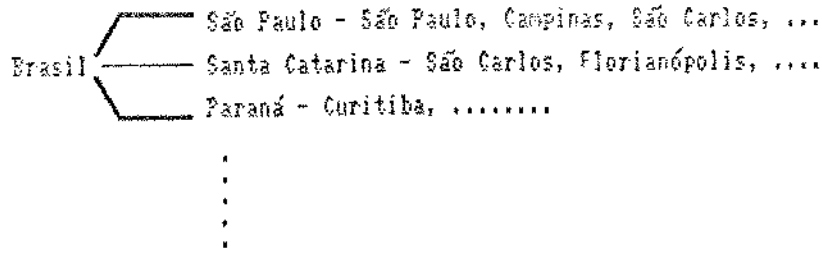


Figura 3.6 - Um Exemplo do Conceito de Classes no SIPS

Os países da classe global estão sempre disponíveis. Colocando em consideração o país Brasil, os estados do Brasil passam a ficar disponíveis. A partir de então, os objetos que podem ser colocados em consideração são os estados. Colocando se em consideração um estado, por exemplo São Paulo, ficam acessíveis todos as cidades do estado de São Paulo, que é o conjunto de objetos constritos por São Paulo. Neste ponto, pode-se fazer referência à qualquer país, aos estados do Brasil e às cidades de São Paulo, bastando para isso indicar o seu nome e seu tipo (estado ou cidade). No entanto, se for necessário fazer referência a alguma cidade que não pertença ao estado de São Paulo, será necessário que se identifique todo o caminho de acesso, indicando o estado e a cidade a que se quer referir.

No exemplo abaixo, se quisermos nos referir às cidades de Campinas ou São Carlos, do estado de São Paulo, basta indicar o seu nome, uma vez que São Paulo é o estado que está sendo considerado. Para referir-se a São Carlos de Santa Catarina ou a Florianópolis, deve-se especificar o estado Santa Catarina e a cidade de São Carlos ou Florianópolis.



Quando o SIPS está sendo usado para desenvolvimento de sistemas aplicando-se o GCS, em um primeiro nível de detalhe seriam reconhecíveis alguns tipos de objeto do tipo sistema e orgão que constituem a organização onde os sistemas estão sendo desenvolvidos. Quando se passa a considerar um determinado sistema, os ICSs que pertencem a este sistema tornam-se acessíveis, porém não os que pertencem aos demais sistemas da organização. A seguir, o usuário pode indicar um dos ICSs disponíveis para consideração e, a partir daí, terá acesso aos dados que compõem o ICS considerado. Este conceito será amplamente utilizado pelo GCS no SIPS, onde o objeto a ser colocado em consideração será um ICS. Estes ICSs podem ser refinados, ou seja, podem dar origem a novos ICSs que representam a evolução dos ICSs ao longo do ciclo de vida do sistema.

Note-se que com o refinamento de um determinado aspecto de um ICS, os demais aspectos não deixam de estar acessíveis, apenas não são por sua vez refinados. Assim, no exemplo acima, quando o usuário indica um sistema para consideração, ele não deixa de ter

acessíveis as informações gerais dos demais sistemas não considerados; porém as informações detalhadas que ele obterá corresponderão apenas ao sistema considerado. Ainda, quando leva-se em consideração um maior detalhe de um ICS, não se aumentam correspondentemente os detalhes dos outros ICSs especificados segundo outras metodologias, mesmo que estejam associados a esse ICS, mas tão somente os detalhes do ICS considerado.

Para implementar esses conceitos o SIPS dispõe de primitivas especiais para a manipulação de classes; além disso, todas as primitivas que acessam objetos através de seus nomes ou sinônimos levam em conta esses conceitos.

Para que o conceito de classes seja entendido dentro do SIPS inicialmente, ao se modelar uma metodologia, deve-se dividir os seus tipos de objetos nos vários aspectos de interesse, de forma a se estabelecer os tipos de objetos pertencentes a cada uma das classes hospedeiras da metodologia.

O passo seguinte é estabelecer a hierarquia na abertura destas classes hospedeiras. Isto se faz através da associação com uma palavra do tipo denominado "considere" que determina quais classes devem estar abertas, quais devem ser fechadas e quais devem ser abertas. Desta forma, uma palavra do tipo "considere" durante a operação do SIPS determina a execução das operações de verificação das classes que devem estar abertas, fechamento das que devem ser fechadas e abertura das que devem ser abertas.

Para exemplificar o uso de uma palavra do tipo "considere" associada à abertura de classes, examinaremos o "considere-dfd" já implementado no SIPS, associado à metodologia Análise Estruturada de Sistemas. A classe hospedeira para agrupar os tipos de objetos desta metodologia é denominada dfd. A hierarquia está estabelecida de tal forma que para se abrir um conjunto de objetos constritos da classe dfd, já deve estar aberta uma classe mais abrangente denominada sistema. Ao se abrir um conjunto de objetos constritos da classe hospedeira dfd primeiramente deve-se fechar algum outro cojunto de objetos constritos por esta classe que possa estar aberta neste instante. O considere-dfd contém os seguintes comandos:

```
deve-estar-aberta sistema
fecha dfd
abre dfd
```

A nível de implementação, um objeto constritor possui associado um registro do tipo classe que contém os endereços dos conjuntos de objetos constritos por ele. Desta forma, a operação de colocar em consideração um objeto constritor significa deixar disponível na memória o registro de classe correpondente tornando

acessíveis os conjuntos de objetos constritos por este objeto constritor.

Da mesma maneira como existe um registro de estado corrente que contém o estado atual do objeto corrente também existe um registro de classe corrente, onde ficam armazenados os endereços dos conjuntos de objetos constritos acessíveis em um dado instante.

3.7 - O Supervisor

O supervisor do SIPS é o responsável pela monitoração e direcionamento do acesso aos vários componentes do SIPS, com os quais o usuário interage. Permite a comunicação entre ferramentas do ambiente SIPS, encarregando-se de efetuar a transição entre elas de tal maneira que o conjunto de ferramentas utilizadas pelo projetista dentro do ambiente tenha a aparência de uma única ferramenta.

O supervisor é formado por um conjunto de primitivas que permite a comunicação entre ferramentas, recuperação de estado de uma ferramenta, passagem de mensagem entre ferramentas e ativação de uma ferramenta. Tais primitivas podem ser usadas pelo projetista de ferramentas.

Pode existir ferramentas que são ativadas independentemente de uma solicitação explícita do usuário. Ferramentas ativadas explicitamente, dependendo de subcomandos ou operações enviadas pelo usuário, ou dependendo de resultados de verificações efetuadas, podem originar solicitação de ativação de outras ferramentas, que são passadas para o supervisor. No momento adequado essas ferramentas são ativadas, independentemente da solicitação explícita. Dessa forma, as várias ferramentas aparecem para o usuário como um conjunto de comandos unificados e não estanques entre si.

A ferramenta passa o controle para o supervisor de duas maneiras.

A primeira é uma saída normal onde a operação da ferramenta é encerrada e o controle é passado para o supervisor o qual, a partir daí, seleciona a próxima ferramenta a ser executada.

A segunda maneira permite que uma ferramenta deixe de ser executada e passe o controle diretamente para outra ferramenta

(via seleção pelo supervisor), ativando-a para depois receber de volta o controle e informações para que prossiga sua execução. Outra maneira de ativar uma ferramenta é solicitando sua execução adiada para o supervisor o qual irá ativar essa ferramenta somente quando a ferramenta corrente tiver sido encerrada e não houver outra solicitação de ativação pelo usuário.

Um exemplo da utilização do supervisor no SIPS pode ser visto entre as ferramentas que automatizam a metodologia Análise Estruturada de Sistemas: o Editor Gráfico de Diagramas de Fluxo de Dados (DFD) e o Editor de Formulários. Suas funções são complementares. O projetista, após obter o diagrama de fluxo de dados, dicionariza esses dados usando o editor de formulários, uma vez que pelo editor gráfico somente ficam armazenadas as informações gráficas e sua identificação na base de dados. Para complementar as informações dos objetos do diagrama, o projetista posiciona em um dado objeto e através de um comando do próprio editor gráfico tem acesso ao formulário correspondente ao objeto; a partir daí pode complementar os dados desse objeto. Após essa operação, é automático o retorno ao diagrama que estava sendo editado.

Desta forma, o supervisor apresenta a facilidade de permitir que as ferramentas que vão sendo incorporadas ao SIPS possam comunicar-se entre si e principalmente, proporcionar um conforto maior ao projetista de sistemas alvo; o projetista não precisa fazer a chamada a outra ferramenta, o próprio editor onde ele está trabalhando o faz por ele. Além disso, o supervisor torna o uso de ferramentas mais eficiente por passar para a ferramenta que está sendo chamada o registro de estado corrente, ou seja, onde o projetista se encontra posicionado na Base Sistema. Sem o uso do supervisor, o projetista teria que sair do editor gráfico, chamar o editor de formulários e posicionar no objeto que vai dicionarizar.

3.8 - Ferramentas Genéricas do SIPS

Dois tipos de ferramentas podem ser gerados dentro do ambiente SIPS: ferramentas genéricas e ferramentas específicas.

Ferramentas específicas são construídas para automatizar metodologias específicas.

Uma ferramenta genérica atua sob uma descrição meta que descreve como as ocorrências do modelo meta da ferramenta estão associados ao modelo meta da metodologia. Estas informações estão

armazenadas na Base Meta.

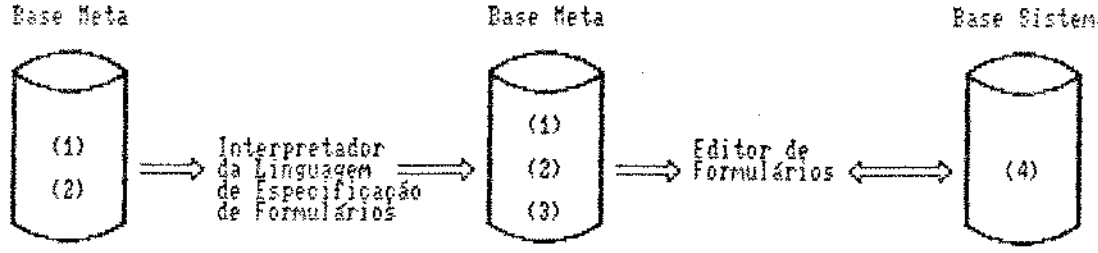
O editor de formulários é um exemplo deste tipo de ferramenta.

Esquematicamente, a Figura 3.7 ilustra como uma ferramenta genérica como o editor de formulários pode atender as necessidades de uma metodologia específica, como se a mesma tivesse sido feita com o propósito de automatizar especificamente a metodologia.

Este processo consiste de dois passos.

O primeiro passo consiste em obter a instanciação que corresponde ao Modelo Meta de uma Metodologia para o Editor de Formulários. Para isso necessita-se do modelo formal do editor de formulários e do modelo formal da metodologia que devem ser submetidos ao Interpretador de Linguagem de Especificação de Formulários.

O segundo passo é a utilização do editor de formulários que, a partir da instanciação obtida no primeiro passo, passa a atuar como uma ferramenta específica da metodologia em questão, permitindo que o projetista descreva seu sistema alvo, a ser armazenado na Base Sistema.



(1) Modelo Formal do Editor de Formulários

(2) Modelo Formal da Metodologia

(3) Instanciação que corresponde ao Modelo Meta de uma Metodologia para o Editor de Formulários

(4) Descrição do Sistema Alvo usando o Editor de Formulários

Figura 3.7 - O Editor de Formulários do SIPS

CAPÍTULO 4

AUTOMATIZAÇÃO DO GCS NO AMBIENTE SIPS

Este capítulo trata da definição da metodologia GCS para o ambiente SIPS descrito na seção anterior.

4.1 - Definição dos Termos da Metodologia GCS no SIPS

Para podermos definir a metodologia GCS para o SIPS, inicialmente, são definidos os termos que compõem a metodologia desenvolvida neste trabalho.

ICS (Item de Configuração de Software) - é um componente de software definido formalmente para acompanhamento e controle durante o desenvolvimento do sistema de software. Um ICS pode assumir três estados:

1 - em edição: quando o ICS é criado e está em desenvolvimento. Para o SIPS, um ICS neste estado é denominado de alternativa;
2 - efetivo: quando um ICS está em um estado transitório, ou seja, é uma alternativa cujo desenvolvimento se encerrou e que deve passar pelo crivo da auditagem. Caso o resultado da auditagem seja positivo, o ICS é congelado; caso contrário, o ICS retorna para o estado em edição, para ser corrigido;
3 - congelado: é um ICS que já passou pelos estados 1 e 2. Portanto, um ICS congelado já passou pelo crivo do GCS tendo obtido resultado positivo no processo da auditagem. Neste estado o ICS muda o seu nome de alternativa para versão. Uma versão de ICS não poderá mais ser alterada. Qualquer alteração que deva ser feita a esta versão gerará uma nova versão deste ICS. Veja definição de versão abaixo.

Instância de ICS - é uma representação associada a um ICS a nível de implementação. Um ICS na Base Sistema pode possuir várias instâncias associadas ao mesmo. As instâncias podem ser de três tipos: alternativa, revisão e variação. A implementação de ICSs por instâncias é que possibilita gerar um histórico que mantém a

estrutura, bem como o conteúdo das modificações de cada ICS, de forma que cada ICS consista em uma árvore de versões juntamente com todos os dados que definem as modificações ao longo do tempo.

Alternativa - é uma instanciação de um ICS que pode ser criada pelo projetista. é a partir de uma alternativa que é gerada uma revisão ou variação. Várias alternativas de uma versão podem ser criadas pelos projetistas durante o desenvolvimento do software. Esta caracterização de alternativa é a mesma adotada por Katz em [15].

Versão - caracteriza as várias formas alternativas, alterações sucessivas, adaptações sofridas por um ICS a partir do instante que o ICS é devidamente reconhecido e está sob o controle do GCS. Para o SIPS uma versão de ICS deve ser distinguida entre dois tipos básicos: revisão e variação. A distinção entre dois tipos de versões no SIPS apresenta as mesmas características de versões dada por Babich em [2].

Revisão - é uma versão de um ICS com o intuito de substituir a anterior. Revisões refletem progresso na depuração de um ICS, embora também possam refletir características funcionais adicionais ou melhorias no desempenho. As revisões aparecem em uma ordem linear, relacionada com a sequência em que elas foram geradas.

Variação - é uma versão de um ICS que preenche as mesmas funções para situações ligeiramente diferentes e, portanto, atuam como opções distintas para um mesmo ICS. Variação, ao contrário de revisão, não substitui uma versão anterior. Múltiplas variações podem coexistir como várias opções de um mesmo ICS. Variações se fazem necessárias se o sistema suporta vários tipos de hardware ou se preenche funcionalidades alternativas. Duas variações podem originar de uma mesma versão, o que não implica que as variações tenham que ser sempre originadas a partir da mesma versão, ou seja, as variações ocorrem como o resultado da necessidade de ramificação de um requisito.

Configuração Básica Parcial - representa um marco que reflete a evolução de um determinado ICS, ou seja, representa o término da evolução de um determinado ICS na fase seguinte do desenvolvimento do sistema. Ela é constituída pelo conjunto de ICSs produtos da fase seguinte da evolução do ICS a que está associada. A configuração básica parcial constitui uma base para uma próxima etapa da evolução do sistema. Para ela se constituir em uma base sólida, ela deve ser colocada sob o controle de configuração, isto é, mudanças subsequentes devem ser feitas somente com a aprovação do controle de configuração.

Configuração Básica Global - representa um marco bem definido no final de cada fase do ciclo de vida do sistema que é alcançado pela finalização e congelamento dos ICSs que constituem todas as configurações básicas parciais que a constituem.

PMS (Proposta de Modificação de Software) - é um documento que é gerado quando existe alguma necessidade de efetuar uma modificação em uma versão de um ICS. Este documento passa então a ser o elemento centralizador do controle que o sistema exerce sobre o que pode ou não ser modificado, adicionado, retirado ou expandido em uma descrição já em andamento, bem como fornece subsídios para um usuário de alto nível decidir sobre as autorizações ou rejeições que dirigem esse controle.

4.2 - Definição de ICS e Configuração Básica no SIPS

O Gerenciamento de Configuração de Software a ser automatizado, pretende seguir os conceitos de Bersoff [4] desenvolvendo os conceitos desta disciplina, dentro de uma abordagem cujo objetivo é o de armazenar todas as informações necessárias ao GCS na base de dados do SIPS, usando o MRO para a representação da informação. Dessa forma, torna-se possível obter auxílio automatizado para tomadas de decisões de gerenciamento, coleta de dados da equipe de produção e manutenção de sistemas de software, e controle automático sobre as atividades desta equipe de acordo com as decisões tomadas.

Para se entender melhor a automatização do GCS sob o SIPS, dois conceitos básicos desta metodologia devem ser situados dentro do ambiente: ICS e Configuração Básica.

O ICS é o componente básico sobre o qual o GCS do SIPS irá atuar. é sobre ele que será feito o controle de versões e de configurações.

O conceito de classes descrito na Seção 3.6 é de grande importância para a implementação da metodologia GCS no SIPS. Isto porque o GCS entenderá qualquer outra metodologia dentro do SIPS através do tipo de objeto ICS que irá constringir a classe de objetos de uma metodologia no SIPS. Desta forma, ao controlarmos o ICS estaremos controlando o conjunto de objetos constritos pelo ICS.

Um ICS pode ser composto de outros ICSs de modo a refletir a evolução do sistema. Então, podemos considerar que existe um ICS mais abrangente que denominaremos de ICS-SISTEMA. Este, por sua vez, será composto de outros ICSs que, no caso do SIPS, se faz melhor representar por tipos de objetos de cada metodologia automatizada dentro do ambiente e que deva ser associada junto à metodologia GCS.

O desenvolvimento de um sistema abrangendo todas as fases do ciclo de vida dentro do SIPS é feito através do uso de ferramentas que automatizam as várias metodologias que cobrem essas fases do ciclo de vida. Se identificarmos dentro de cada metodologia os tipos de objetos que serão identificados como ICSs, durante o uso sequencial destas metodologias naturalmente esses ICSs vão sendo desenvolvidos, ou seja, evoluem. O papel da metodologia GCS dentro do SIPS, será então o de acompanhar a evolução desses ICSs.

Este controle deve ser feito sobre os ICSs congelados. A partir dos ICSs congelados, pode-se gerar configurações básicas. Vendo mais profundamente o conceito do GCS, percebe-se que a evolução do sistema é obtida através das várias configurações básicas estabelecidas ao longo do desenvolvimento do sistema.

Uma configuração básica do sistema segundo Bersoff [4] é um ponto de referência bem definido durante o desenvolvimento do sistema. As configurações básicas servem como marcos que dividem as várias fases do ciclo de vida do software.

Ao se tentar automatizar o conceito de configuração básica, que nos parece bastante importante no acompanhamento da evolução do sistema, percebe-se que a conceituação dada em [4] é bastante teórica. Um sistema em desenvolvimento não apresenta marcos globais estanques ao final de cada fase do ciclo de vida. Desta forma é impossível implementar este conceito, pois não é viável um sistema automatizado que obrigue o desenvolvimento de um sistema rigidamente dentro do ciclo de vida, por não permitir evoluir somente uma parte do projeto se as demais partes não estiverem terminadas.

A conceituação de configuração básica dentro do SIPS será então ligeiramente alterada, adotando-se os conceitos de configuração básica parcial e configuração básica global apresentados na Seção 4.1.

Dado que configuração básica consiste em um conceito muito importante para esta metodologia, seria desejável que configurações básicas pudessem ser geradas automaticamente ao longo da evolução dos ICSs. Da maneira como elas foram apresentadas e implementadas dentro do SIPS através do mecanismo de classes, isto é possível desde que se tenha um controle sobre o ato de se criar um ICS e sobre onde cada ICS criado deve se situar dentro do contexto do sistema em questão. O gerente ao criar um ICS deve situá-lo dentro do contexto correto, ou seja, se o ICS a ser criado reflete uma parte do detalhamento de um ICS mais abrangente, então ele deve estar contido dentro desse ICS mais abrangente.

Dado que configurações básicas parciais são constituídas a partir de um conjunto de ICSs congelados, então se o conjunto de ICSs do mesmo tipo derivados diretamente de um ICS estiverem todos congelados, consequentemente a configuração básica parcial associada a esse ICS mais abrangente terá se estabelecido.

Uma configuração básica global é uma composição de configurações básicas parciais; da mesma forma, a partir do momento que todas as configurações básicas parciais que a compõem estiverem estabelecidas então a configuração básica global também se estabelecerá automaticamente.

A Figura 4.1 apresenta um exemplo dos conceitos configuração básica parcial e global.

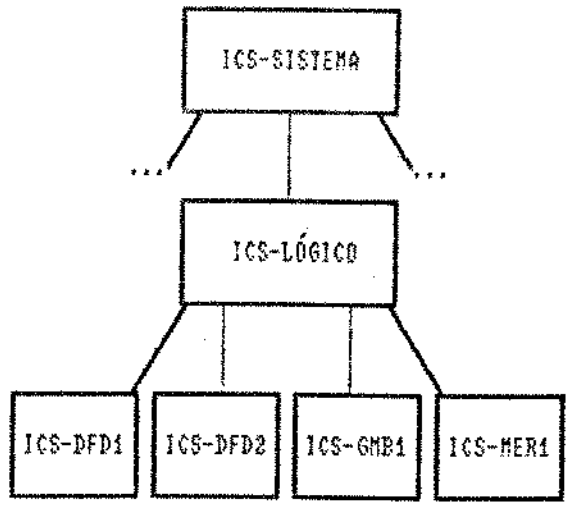


Figura 4.1 - Exemplo de Configuração Básica Parcial e Global

Se os ICS-DFD1 e ICS-DFD2 representam o desenvolvimento do sistema usando a metodologia Análise Estruturada de Sistemas, então, a partir do momento que estes dois ICSs estiverem congelados, a configuração básica parcial referente à metodologia Análise Estruturada de Sistemas é estabelecida.

Neste exemplo, ICS-GMB1 representa o ICS que se refere à avaliação da dinâmica do sistema obtida através do GMB da metodologia SARA [21] e ICS-MER1 representa o ICS referente ao modelamento de dados obtido através da metodologia MER [8].

Se pensarmos no desenvolvimento incremental deste sistema, podemos perfeitamente admitir que, concluído o desenvolvimento dos diagramas de fluxo de dados com o dicionário de dados da metodologia Análise Estruturada de Sistemas, podemos evoluí-lo para a fase de projeto estruturado, sem ainda ter concluído o desenvolvimento do ICS-GMB1.

Se o ICS-LÓGICO representa um marco na evolução do desenvolvimento do sistema, pois a finalização deste ICS garante que a fase de projeto lógico estará concluída, então a finalização deste ICS estabelece uma configuração básica global que corresponde à fase de projeto lógico. Mas esta configuração básica global somente existirá após a finalização das configurações básicas parciais ICS-DFD1, ICS-DFD2, ICS-GMB1 e ICS-MER1; ou seja, quando as configurações básicas parciais destes quatro ICSs que compõem o ICS-LÓGICO estiverem estabelecidas, automaticamente a configuração básica global passa a existir.

O mesmo ocorre também para o ICS-SISTEMA que somente se estabelecerá como configuração básica global quando todos os ICSs que o compõem estiverem congelados, portanto estando com todas as configurações básicas parciais estabelecidas. Neste momento o sistema representado por esse ICS-SISTEMA está pronto e íntegro segundo os princípos do GCS.

4.3 - Definição da Metodologia GCS para o SIPS

O GCS a ser introduzido no SIPS não consiste apenas na introdução de uma nova metodologia a ser automatizada dentro deste ambiente, ou voltada para uma fase específica do ciclo de vida de software. A metodologia GCS a ser introduzida pretende ser genérica o suficiente de forma que ela consiga desempenhar o

seu papel efetivamente sobre a base de dados do SIPS durante toda a evolução de um software desenvolvido dentro do ambiente, independentemente de quais metodologias estão sendo usadas para tal propósito.

Seguindo o enfoque que se quer dar para o GCS dentro do ambiente SIPS, novas primitivas devem ser acrescentadas ao núcleo do SIPS além de uma ferramenta denominada Gerenciador de Configuração de Software. As primitivas têm por objetivo aceitar esses novos conceitos introduzidos de forma a controlar as versões dos ICSs, bem como as configurações que vão sendo geradas durante o desenvolvimento de um sistema. A ferramenta tem por objetivo assegurar que as funções do GCS sejam desempenhadas dentro do SIPS.

A introdução de novas primitivas ao núcleo do SIPS significa que estas constituem um nível mais básico que o nível das ferramentas. Portanto, elas devem ser utilizadas pelas ferramentas desenvolvidas para o SIPS.

Para que elas funcionem de forma sincronizada com as primitivas já existentes no núcleo do SIPS, deve haver um esquema de proteção a nível de GCS na base de dados do SIPS. Este esquema de proteção está basicamente relacionado com a não permissão de atualização em um conjunto de objetos constritos por um ICS que já esteja congelado.

As novas primitivas, na verdade, são as responsáveis pelo crivo do GCS que se pretende colocar na base de dados do SIPS. Elas, conjuntamente com o esquema de proteção a nível de GCS, efetuam a função contida no bloco gerenciador de configuração de software que aparece na Figura 3.1.

Uma metodologia a ser automatizada no SIPS se beneficiará dos recursos do GCS do SIPS, se for modelada integrando a metodologia GCS juntamente com ela e com a utilização dos recursos oferecidos pelas primitivas de GCS através da(s) ferramenta(s) que a automatiza(m). Um exemplo de integração de uma metodologia com a metodologia GCS pode ser visto na Seção 4.5.

4.3.1 - A Evolução de um Projeto usando o GCS sob o SIPS

As quatro funções básicas do GCS no SIPS são executadas ao longo do desenvolvimento do sistema através do uso das várias ferramentas do SIPS juntamente com a ferramenta Gerenciador de Configuração de Software.

Os itens de a) a i) apresentam a maneira como o GCS deve ser usado dentro do SIPS bem como a interação que deve existir entre a ferramenta Gerenciador de Configuração de Software e as demais ferramentas do ambiente. Os itens abaixo mostram todos os passos que um ICS percorre durante a vida de um sistema, desde a sua criação até se tornar um ICS congelado e a sua manutenção.

a) Os ICSs são criados somente pela ferramenta Gerenciador de Configuração de Software, com o seu estado em edição, ou seja, são alternativas. No ato da criação de um ICS é estabelecido em qual contexto ele se situa;

b) Estando criado o ICS um projetista, através de uma ferramenta qualquer, desenvolve esta alternativa, podendo até criar novas alternativas para um mesmo ICS;

c) Uma vez que o projetista termina o desenvolvimento da alternativa, ele a efetiva através da ferramenta que estava usando para desenvolvê-la;

d) Uma alternativa é congelada somente pela ferramenta gerenciador de configuração de software. O gerenciador pode congelar uma alternativa criando uma revisão ou variação, se todos os resultados dos procedimentos aplicados a alternativa relacionados com a função de auditagem forem positivos, ou então devolvê-la para o estado em edição. É no momento da criação de uma versão que as configurações básicas do sistema vão sendo estabelecidas automaticamente.

Através dos passos de a) a d) os ICSs vão sendo gerados e evoluem ao longo do sistema, sendo que o controle de se estabelecer as configurações básicas é obtido automaticamente.

Mas um sistema real sofre muitas modificações durante a fase de manutenção e mesmo durante o seu desenvolvimento. Para esta tarefa, é muito importante a atuação do GCS de forma a evitar o caos do sistema, principalmente de grandes sistemas, onde várias pessoas estão envolvidas.

A Figura 4.2 mostra como é exercida a função de controle de configuração.

O processo de modificação de um ICS:

e) começa quando se detecta um problema com o software que já está sob o controle do GCS. O projetista preenche um formulário de modificação (PMS) associado ao ICS que está envolvido com o problema usando a ferramenta Gerenciador de

Configuração de Software. Caso o problema já tenha sido detectado anteriormente, este formulário é recusado;

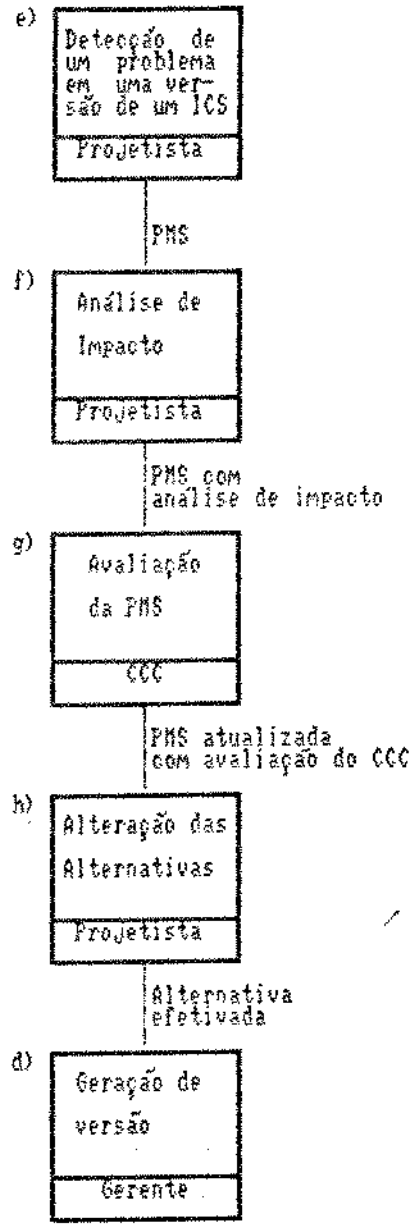


Figura 4.2 - O Controle de Configuração do SIPS

f) o gerente de configuração atribui a algum projetista ou a uma equipe a tarefa de analisar o impacto da modificação sobre o software envolvido. Para esta tarefa que lhe foi atribuída, o

projetista pode utilizar ferramentas automatizadas, caso existam; ou fazê-la de forma manual. O resultado desta análise deve apresentar quais os ICSs envolvidos com a modificação proposta e, se possível, quais os efeitos reais que cada um destes ICSs sofrerá. O objetivo desta análise serve para ajudar a determinar tanto os efeitos técnicos como os efeitos gerenciais. Estando a análise de impacto pronta, as conclusões resultantes devem fazer parte da PMS, que é conseguida através da atualização da PMS pelo projetista;

g) a partir de então esta PMS deve ser levada ao conhecimento do Comitê de Controle de Configuração (CCC) que aceitará ou rejeitará as alterações a serem feitas no software. O CCC é composto por pessoas responsáveis pelo sistema e que tenham conhecimento a respeito do mesmo. O CCC reúne-se periodicamente e analisa os problemas de acordo com suas prioridades. Após a reunião deve ficar claro o impacto total da modificação a ser feita. Duas possíveis decisões podem surgir:

- A modificação está dentro do escopo - isto é, está dentro do escopo do trabalho autorizado pelo cliente ou gerente. Uma modificação dentro do escopo corrige o software; portanto significa que ambas as partes, o projetista e o cliente, concordam que deve ser feita.
- A modificação está fora do escopo. Isto representa uma modificação no planejamento original do software. Pode ser uma modificação significativa que não foi analisada anteriormente, mas que todos concordam que deva ser feita, ou pode ser uma modificação para implementação futura, ficando esta arquivada para uma futura análise e implementação. Após a reunião, o gerente atualiza na PMS o estado das modificações concluídas pelo CCC. Uma modificação arquivada implica somente na sinalização que esta PMS não terá seu prosseguimento para o momento. Caso a modificação deva ser implementada, de acordo com o seu tipo fica já estabelecido que se trata de uma revisão ou uma variação que será gerada. Esta informação deve ser atualizada na PMS associada;

h) a partir de então, o projetista ou a equipe responsável pela correção do sistema corrige os ICSs relacionados com a modificação que foi determinada pela análise de impacto e autorizada pelo CCC, gerando-se alternativas dos mesmos;

i) a forma de efetivar uma alternativa e de se criar um ICS congelado bem como estabelecer configurações básicas está apresentado nos itens c) e d) descritos acima.

Uma vez que foram geradas revisões ou variações, estas informações atualizadas que estão dentro da base de dados do SIPS passam a ficar disponíveis e podem ser acessadas por qualquer ferramenta, desde que o usuário tenha a permissão necessária para tal acesso. Por exemplo, podem ser acessadas via editor de formulários ou gerador de relatórios recuperando as informações e exercendo-se a função de Administração de Estado.

É importante notar os vários passos que foram seguidos antes de implementar uma modificação no software. Pode parecer um trabalho moroso, mas é amplamente necessário e a utilização de um sistema automatizado para estas tarefas torna-o muito mais rápido.

4.3.2 - As Atividades de GCS sob o SIPS

A seguir são apresentadas as quatro funções do GCS sendo executadas pela ferramenta Gerenciador de Configuração de Software juntamente com as demais ferramentas do SIPS.

Identificação de Configuração de Software

A função de identificação consiste na criação do ICS dentro do contexto correto de forma que possam ser geradas as configurações básicas automaticamente como apresentado no Seção 4.2. Ao se criar um ICS deve-se também estabelecer o relacionamento **identifica** com o objeto da metodologia que deverá ser controlado por esse ICS, o relacionamento **é_uma_evolução_de** com o ICS constritor da classe à qual o ICS que está sendo criado pertence, bem como relacionar quais os objetos de outros ICSs que serão **importados** para o ICS que está sendo criado.

Controle de Configuração de Software

A função de controle de configuração de software visa controlar as alterações que vão sendo feitas nos ICSs do sistema através da geração de PMSs para controle de preparação, avaliação e controle de execução de alterações.

Uma PMS no SIPS pode ser preenchida através do Editor de Formulários. Desta forma, uma PMS pode conter os mais diversos campos que se julgue necessário armazenar como histórico de uma PMS.

O exemplo de formulário PMS apresentado abaixo consiste dos seguintes campos:

- identificação da PMS;

- ICS a ser modificado;

- versão do ICS que será modificada;

- descrição da modificação;

- data da geração da PMS;

- autor da PMS;

- ICSs afetados pela análise de impacto;

- custo estimado pela análise de impacto;

- cronograma estimado pela análise de impacto;

- responsável pela análise de impacto;

- observações da análise de impacto;

- data da reunião do CCC;

- observações do CCC;

- responsável pela modificação;

- observações após a modificação;

- data de fechamento da PMS;

- estado da PMS.

O documento PMS é o centralizador de todas as informações referentes a uma modificação proposta para um ICS. Para tanto, os campos acima referem-se às informações necessárias que devem ser obtidas e armazenadas de forma a registrar a situação em que uma modificação de um ICS se encontra.

O estado da PMS pode assumir valores de 1 a 5. O seu significado é:

1 - gerada a identificação da PMS;

2 - efetuada a análise de impacto da modificação;

3 - analisada pelo CCC;

4 - analisada pelo CCC e arquivada;

5 - concluídas as modificações requeridas pela PMS.

Uma PMS ao ser gerada deve fornecer as seguintes informações: uma identificação para que ela possa ser referenciada posteriormente; um ICS, mais precisamente uma versão de ICS que está com algum problema e é o causador da geração de uma PMS; a descrição da modificação requerida; a data da geração da PMS; o autor da geração desta PMS e a sinalização de seu estado como sendo igual a um.

A partir de então, esta PMS deve ser encaminhada para que seja efetuada a análise de impacto referente à modificação descrita. Estando a modificação encaminhada para a análise de impacto, deve-se incluir na PMS o responsável por esta tarefa. Finalizada esta tarefa, a PMS deve ser atualizada com os ICSs afetados pela modificação requerida, o custo e o cronograma estimados para realização desta modificação, observações resultantes da análise de impacto e a sinalização de seu estado como sendo igual a dois.

Esta PMS, está pronta para ser submetida ao CCC, que a analisará. Após a reunião, a PMS deve ser novamente atualizada com a data da reunião do CCC, observações do CCC e se a modificação requerida pela PMS deve ser feita, sinalizá-la passando o seu estado igual a três ou, se deve ser arquivada, sinalizá-la passando o seu estado igual a quatro.

Caso a modificação deva ser feita, então ela deve ser encaminhada ao projetista ou equipe responsável por tal tarefa. Estando a modificação encaminhada para o responsável por esta tarefa, deve-se atualizar na PMS o responsável por esta tarefa. Somente depois que as modificações forem concluídas e as novas configuraçõoes básicas referentes aos ICSs afetados forem estabelecidas é que deve se concluir esta PMS atualizando-a com as observações resultantes da modificação, data de fechamento da PMS e sinalizar o seu estado igual a cinco.

Auditagem de Configuração de Software

O processo de auditagem de um software poderia ser feito através de várias ferramentas que auxiliariam na verificação, validação, análise de consistência, completeza, clareza e testabilidade de um ICS.

A função de auditagem para o SIPS poderia ser traduzida pela avaliação dos resultados obtidos de ferramentas automatizadas genéricas e específicas para cada metodologia e por

técnicas utilizadas para a obtenção da análise que se julgue necessária empregar para auditar um produto de software.

Visto que este processo de auditagem pode ser tão amplo quanto se possa imaginar, então podem ser geradas ferramentas voltadas a analisar aspectos tão diversos de uma metodologia quanto se julgue necessário. Para o GCS basta ter o conhecimento de quais ferramentas integram o conjunto responsável pelo processo de auditagem para que se possa executá-las; se o resultado obtido pela execução de cada uma destas ferramentas for positivo, um ICS poderá ser congelado.

Vale ressaltar que o GCS não se preocupa com os critérios para se avaliar um ICS, mas com os resultados obtidos desta avaliação para que o GCS gere um ICS congelado ou não. Desta forma, estas ferramentas não fazem parte mas auxiliam a metodologia GCS implementada. Para um ambiente como o SIPS, isto seria de grande importância, uma vez que o objetivo de um ambiente de produção de software é o de fornecer o maior número possível de ferramentas automatizadas que auxiliam a tarefa de desenvolvimento de um projeto de software.

Diante disso, supondo-se a existência de várias ferramentas desenvolvidas para este propósito, para cada variação de ICS gerada no SIPS, deveriam existir associados alguns tipos de atributos relacionados com os critérios de avaliação do processo de auditagem. Somente quando todos os critérios estivessem satisfeitos, o processo de auditagem teria se concretizado para um dado ICS.

Este trabalho não trata da automatização dos vários métodos e técnicas utilizadas na realização da auditagem. Desta forma, no estágio atual do SIPS, os métodos e técnicas de auditagem de sofware devem ser aplicados manualmente. Somente após a avaliação dos resultados obtidos do processo de auditagem, duas seguintes operações do Gerenciador de Configuração de Software poderão ser solicitadas:

- congelar uma versão de ICS a partir de uma alternativa em estado efetivo, se o resultado da auditagem for positivo;

- devolver ao estado em edição uma alternativa de um ICS que estava em estado efetivo, se o resultado da auditagem for negativo.

A implementação da abordagem descrita acima consistiria em alterar a ferramenta Gerenciador de Configuração de Software para que o comando de auditagem apresentasse subcomandos correspondentes aos vários critérios de auditagem dos ICSs da metodologia particular. Assim, a ferramenta Gerenciador de

Configuração de Software só geraria uma versão de um ICS que tivesse sido examinado sob todos os critérios e cujos resultados tivessem sido positivos. Somente neste momento o estado do ICS passaria de efetivo (2) para congelado (3).

Se alguns desses critérios não forem satisfeitos, o ICS não passou pelo processo de auditagem. Nesse caso, o estado do ICS passaria de efetivo (2) para em edição (1).

Estes critérios poderiam ser estabelecidos durante o modelamento da metodologia quando poderiam ser definidos os métodos responsáveis pela auditagem para a fase do projeto associada à metodologia. Desta forma, os critérios também fariam parte da Base Meta.

Administração de Estado da Configuração

A função de administração de estado é obtida automaticamente ao se atualizar ICSs pelas demais ferramentas do SIPS, uma vez que todas as informações associadas a estes ficam armazenadas na Base Sistema. Tanto o registro de estado como o histórico dos ICSs e das configurações básicas são obtidos via ferramentas de saída genéricas do SIPS como o gerador de relatórios. Desta forma, este trabalho não se preocupa com os detalhes desta função uma vez que ela será de fato exercida por outra ferramenta do SIPS.

A implementação do GCS neste trabalho é feita de forma que tanto o registro de estado corrente como o histórico desenvolvido para cada ICS permaneçam na Base Sistema.

A implementação desta função para este trabalho está restrita à apresentação de configurações básicas parciais e globais, revisões, variações e alternativas de um ICS, sem se preocupar em relacionar estas informações juntamente com os dados que os compõem de modo a refletir o andamento real das atividades dentro de um projeto em desenvolvimento no SIPS.

Como exemplos de relatórios que podem ser obtidos para a função de administração de estado, através de ferramentas de saída genéricas do SIPS, podemos relacionar relatórios que apresentem:

- todas as PMSs do sistema bem como os seus respectivos estados;
- PMSs em andamento;
- PMSs arquivadas;
- PMSs concluídas, ou seja, aprovadas e implementadas;
- PMSs geradas que devem ser encaminhadas para análise de impacto;
- PMSs a serem analisadas pelo CCC;
- PMSs já analisadas pelo CCC;
- ICSs afetados por uma dada PMS;
- ICSs que possuem versões em desenvolvimento;
- o histórico de versões geradas para um ICS;
- todas as alternativas relacionadas com um ICS;
- ICSs que compõem uma configuração básica parcial;
- ICSs que compõem uma configuração básica global;
- para cada projetista relacionar:
  - quais as PMSs que gerou;
  - de quais PMSs efetuou a análise de impacto;
  - de quais PMSs efetuou a modificação nos ICSs afetados;
- qual o motivo da geração de uma versão para um dado ICS;
- a configuração básica corrente do software operacional.

As ferramentas do SIPS são as responsáveis pela manipulação de uma alternativa, podendo desenvolver, criar, efetivar e corrigir alternativas e consultar versões de ICSs. A ferramenta Gerenciador de Configuração de Software também pode criar, efetivar alternativas, bem como consultar versões de ICSs. Uma tarefa que o Gerenciador de Configuração de Software não está apto a executar seria a de desenvolver e corrigir alternativas, uma vez que o conteúdo das alternativas é especifico da metodologia adotada para o desenvolvimento das mesmas.

4.4 - Modelamento da Metodologia GCS usando o MRO (Modelo de Representação de Objetos)

O SIPS pode ser caracterizado como um ambiente que suporta a automatização de várias metodologias, independentemente do tipo de aplicação ou da fase do ciclo de vida a que elas se destinam.

O GCS no SIPS, portanto, também deve possuir uma abordagem genérica de forma que possa atender a todas essas novas metodologias que vão sendo incorporadas ao ambiente. Seguindo esse objetivo, foi feita uma modelagem genérica do GCS para o SIPS. Esta abordagem permite que, ao se introduzir uma nova metodologia no SIPS, o especialista da metodologia possa ter a liberdade de introduzir a metodologia tal como ela é, ou então eleger tantos objetos desta metodologia quanto julgue necessários para se exercer o controle do GCS.

Os objetos da metodologia eleitos para ter atenção do GCS podem ter as suas versões controladas e acompanhadas pelo SIPS. E isto ocorre de uma forma genérica, ou seja, o controle que o GCS faz é sobre objetos da metodologia que o especialista da metodologia escolheu, independentemente da metodologia; consequentemente, independentemente da fase do ciclo de vida a que este objeto pertence. Como exemplo, um objeto pode ser um documento da especificação funcional, um programa fonte ou um caso de teste.

A seguir é apresentado o modelamento do GCS para o SIPS usando o MRO. A Figura 4.3 representa este modelamento.

O principal tipo de objeto a se destacar na metodologia GCS a ser adotada é o ICS. Os objetos do tipo ICS correspondem à contra-partida do GCS em relação aos objetos factuais da metodologia de auxílio ao desenvolvimento de software adotada. É através dos ICSs que todas as funções do GCS são refletidas em uma metodologia, qualquer que seja ela.

O conjunto de tipos de relacionamentos necessário para a integração do GCS com a metodologia adotada é dependente dessa metodologia; porém, deve conter relacionamentos para permitir a representação das informações necessárias ao GCS sobre os objetos da metodologia.

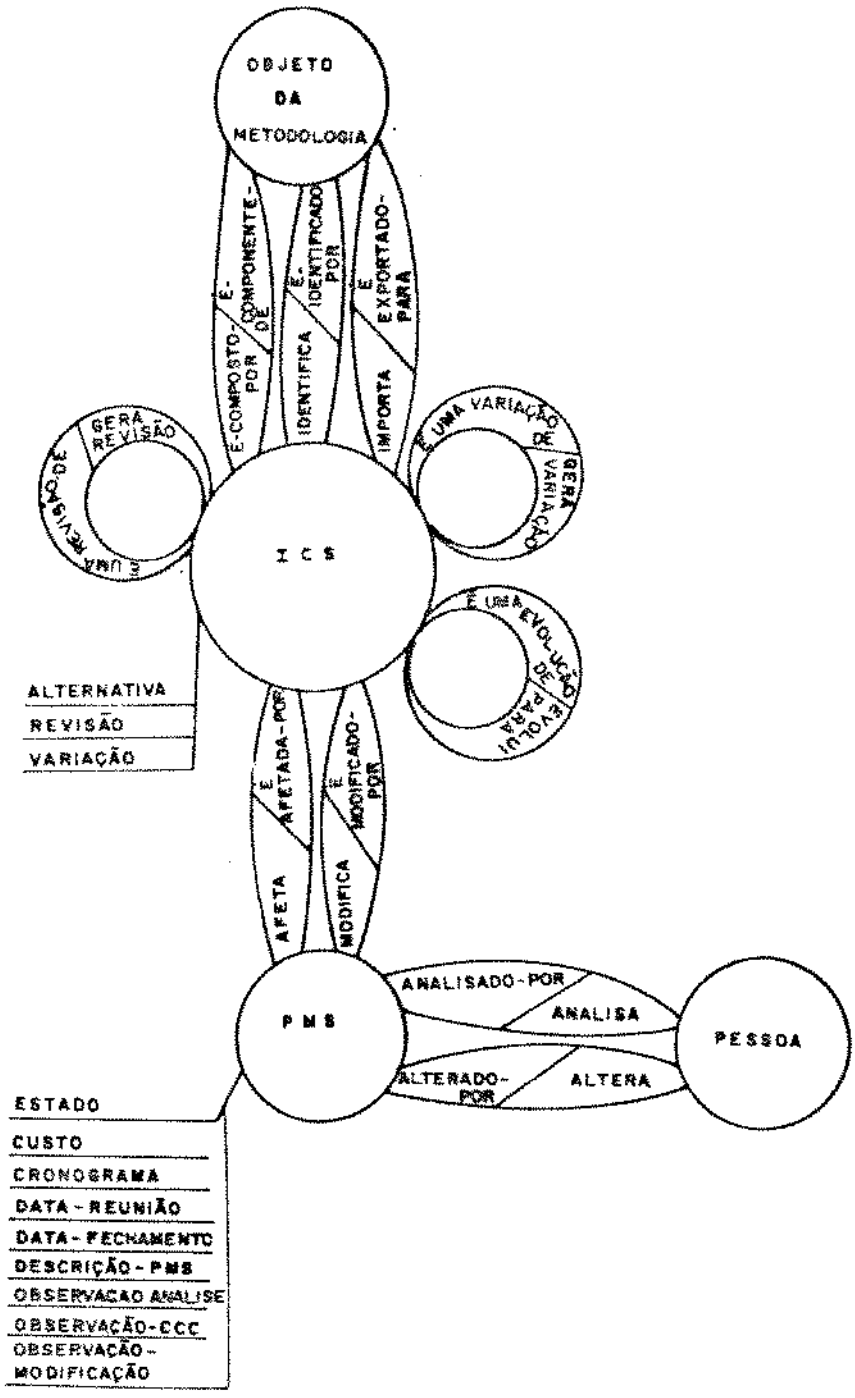


Figura 4.3 - Modelamento da Metodologia GCS para o SIPS usando o MRO

O relacionamento **identifica** deve ser aplicado entre um objeto do tipo ICS e um objeto da metodologia que deva ser controlado explicitamente pelo GCS através do ICS pelo qual é identificado univocamente.

O relacionamento **importa** deve ser aplicado entre um objeto do tipo ICS e todos os objetos da metodologia que possam estar presentes em mais de um ICS de mesmo nível de abstração, ou que correspondam à interface entre o interior e o exterior de um ICS.

Para que o **controle de versões** possa ser efetuado sobre ICSs de forma genérica, o controle deve ser exercido sobre as instâncias de um ICS.

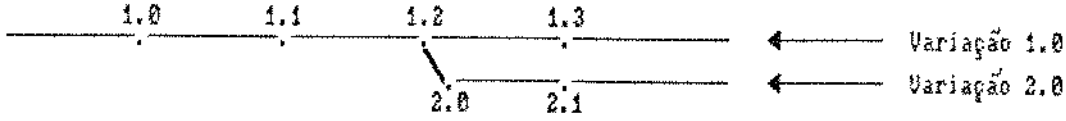
As instâncias estão representadas no modelamento através dos tipos de sinônimos associados ao ICS: alternativa, revisão e variação. Desta forma, um ICS ao longo de sua existência apresenta várias instâncias na Base Sistema, de forma a refletir o seu histórico.

O relacionamento **composto_por** identifica o conjunto de objetos da metodologia constritos por uma instância de um ICS.

O relacionamento **é_uma_variação_de** permite a implementação do conceito de variações, de forma a permitir visualizar a sequência em que as variações vão sendo geradas.

O relacionamento **é_uma_revisão_de** permite mostrar a sequência em que as revisões são geradas dentro de uma variação. Isto permite reconstituir o histórico associado a cada variação de um ICS do sistema.

Exemplificando os conceitos acima, temos:



Neste exemplo:

Revisão 1.3 **é uma revisão** da revisão 1.2,
variação 2.0 **é uma variação** da revisão 1.2 e
revisão 2.1 **é uma revisão** da variação 2.0.

Com as instâncias de um ICS formalizadas, bem como os relacionamentos **é_uma_variação_de** e **é_uma_revisão_de** pode-se então obter o controle sobre todas as instâncias relacionadas com

um ICS ao longo de sua existência, desde a geração de um histórico relacionado ao mesmo, bem como apresentar o registro corrente de sua evolução.

O **controle de geração de configuração básica** no SIPS fundamenta-se também sobre os objetos do tipo ICS. Desta forma, novos atributos devem ser acrescentados a um ICS para garantir que uma configuração básica possa ser estabelecida automaticamente através da análise desses atributos.

Os atributos intrínsecos relacionados a um ICS e responsáveis pelo estabelecimento de configurações básicas parciais e globais são:

- quantidade de alternativas. Para cada versão de um ICS, é feito um controle de quantidade de alternativas associadas. Isto se faz necessário porque o GCS do SIPS permite gerar várias alternativas para uma mesma versão de um ICS. Mas o fato de existirem duas alternativas para uma mesma versão de um ICS significa que somente uma versão de ICS está sendo atualizada;

- quantidade de versões em aberto. Para cada ICS, é feito um controle de versões que possuem alternativas em desenvolvimento. Uma configuração básica é estabelecida somente se todos os ICSs que a compõem não apresentarem nenhuma versão em aberto.

O relacionamento **é_uma_evolução_de** é o responsável para gerar configurações básicas globais. Isto porque é este relacionamento que determina quais os ICSs que compõem um ICS mais abrangente que se constituirá em uma configuração básica global.

Assim, uma configuração básica global é estabelecida quando todos os ICSs que têm o relacionamento **é_uma_evolução_de** com um ICS origem não apresentarem versões em aberto para atualizações. Como exemplo, na Figura 4.1, os ICSs ICS-DFD1, ICS-DFD2, ICS-GMB1 e ICS-MER1 apresentam o relacionamento "é_uma_evolução_de" com o ICS-LóGICO, ou seja, os ICSs ICS-DFD1, ICS-DFD2, ICS-GMB1 e ICS-MER1 compõem o ICS-LóGICO. Quando estes quatro ICSs não apresentarem versões em aberto, a configuração básica global ICS-LóGICO é estabelecida.

Uma configuração básica parcial é estabelecida quando todos os ICSs de um mesmo tipo não apresentarem versões em aberto para atualizações. Referindo-se ao exemplo da Figura 4.1, a configuração básica parcial relacionada com a metodologia Análise Estruturada de Sistemas se estabelecerá somente quando os ICS-DFD1 e ICS-DFD2 que a representam não apresentarem versões em aberto.

A **função de controle de configuração** no SIPS é feita sobre o objeto do tipo PMS. Uma PMS é gerada a partir de um ICS que deve ser modificado. Desta forma, o seu modelamento apresenta o relacionamento PMS **modifica** um ICS. Mas uma PMS ao modificar um ICS pode **afetar** outros ICSs.

Para gerar um histórico de uma PMS pode ser associada a esta tantas propriedades diferentes quanto se julgue necessário. Neste modelo, estão relacionados os atributos:

- estado da PMS, que apresenta o estado em que se encontra uma PMS ao longo do processo de controle de configuração;
- custo associado a modificação proposta pela PMS;
- cronograma associado à modificação proposta pela PMS;
- data da reunião do CCC para analisar a PMS;
- data de fechamento da PMS;
- descrição da alteração requerida pela PMS;
- observações geradas após a análise de impacto feita na PMS;
- observações geradas após a reunião do CCC sobre a PMS;
- observações geradas após a modificação requerida pela PMS.

Um objeto do tipo pessoa está relacionado com a PMS. Isto porque uma PMS **é_analisada_por** uma equipe estando o seu responsável registrado na PMS. Da mesma forma, uma PMS **é_alterada_por** uma equipe estando o seu responsável registrado na PMS. Esta característica, não relevante ao GCS, de certa forma é uma introdução ao gerenciamento de projeto, pois forneceria informação para o controle das atividades realizadas por cada um dos membros do projeto.

4.5 - Modelamento do GCS com a Análise Estruturada de Sistemas

Nesta seção é apresentado o exemplo de modelamento implementado para este trabalho que engloba a metodologia GCS com a metodologia Análise Estruturada de Sistemas. Para isso, foi utilizado o modelo apresentado na Figura 3.5 que modela a metodologia Análise Estruturada de Sistemas usando o MRO.

A Figura 4.4 apresenta o modelamento da Análise Estruturada de Sistemas juntamente com o GCS.

O tipo de objeto da Análise Estruturada de Sistemas selecionado para receber atenção especial pelo GCS foi o tipo de objeto DFD (Diagrama de Fluxo de Dados). Desta forma, a figura apresenta o relacionamento **identifica** entre os objetos do tipo ICS-DFD e DFD.

Os objetos do tipo fluxo de dados podem ser componentes de mais de um DFD, pois um mesmo fluxo de dados pode estar presente em um DFD como entrada ou saída de uma dada função bem como no DFD que representa o detalhamento da função que recebe ou gera o fluxo de dados. Isto está representado no modelo através do relacionamento ICS-DFD **importa** fluxo de dados, ou seja, o fluxo de dados **é_exportado_para** um ICS-DFD.

O relacionamento ICS-DFD **é_composto_por** função, fluxo de dados, entidade externa, arquivo lógico, estrutura de dados e elemento de dados identifica o conjunto de tipos de objetos da Análise Estruturada de Sistemas constritos pelo ICS-DFD.

Para ilustrar a evolução de ICSs neste exemplo é considerado que o sistema evolui através do desenvolvimento de diagramas de fluxo de dados. Isto é representado como: o tipo de objeto ICS-DFD **é_uma_evolução_do** tipo de objeto ICS-SISTEMA.

Neste exemplo, uma configuração básica global se estabelecerá quando todos as versões de ICSs do tipo ICS-DFD estiverem com suas versões congeladas. Uma configuração básica parcial se estabelecerá quando todas as versões de ICSs do tipo ICS-DFD estiverem com suas versões congeladas. Como pode se notar neste exemplo, uma configuração básica global se confunde com uma configuração básica parcial. Isto porque este é um exemplo simples de modelamento, uma vez que envolve somente uma metodologia junto com a metodologia GCS.

Isto é um conceito importante para o SIPS que em seu estado atual comporta outras metodologias. Não será dado o modelamento completo das metodologias automatizadas pelo SIPS pois, para efeito de ilustração, não nos importam os detalhes destas metodologias. Consideremos que além da Análise Estruturada de Sistemas, o modelamento envolve a metodologia MER (Modelo Entidade-Relacionamento) [8]. Assim, o ICS-MER também é uma evolução do ICS-SISTEMA.

Então, uma configuração básica global para o ICS-SISTEMA se estabelecerá quando estiverem estabelecidas as configurações básicas parciais para o ICS-DFD e para o ICS-MER. Uma configuração básica parcial para o ICS-DFD se estabelecerá quando todas as versões de ICSs do tipo ICS-DFD estiverem com suas versões congeladas. Da mesma forma, uma configuração básica parcial para o ICS-MER se estabelecerá quando todas as versões de ICSs do tipo ICS-MER estiverem com suas versões congeladas.

Com este modelamento, todos os diagramas de fluxo de dados que forem sendo gerados poderão ter suas versões e configurações controladas.

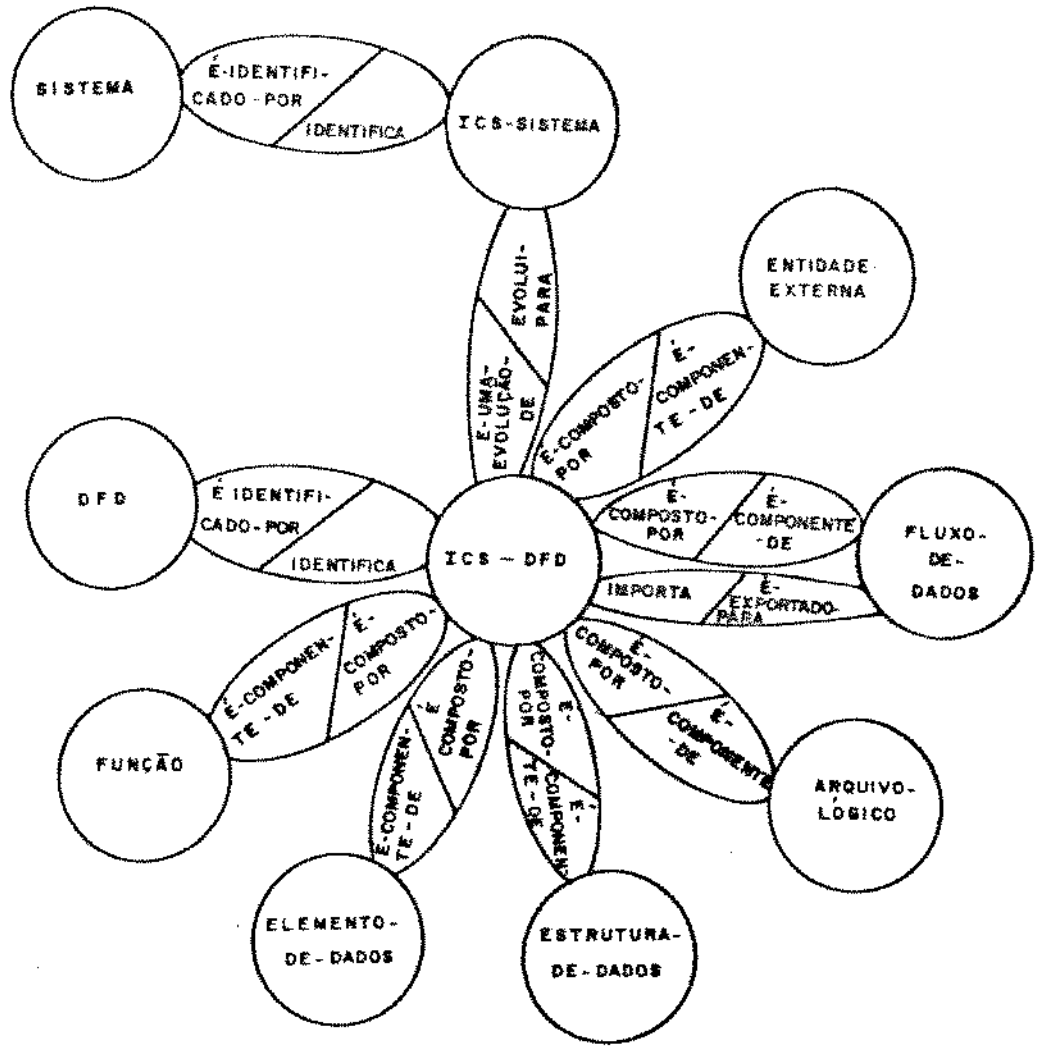


Figura 4.4 - Modelamento de GCS e Análise Estruturada de Sistemas para o SIPS

CAPÍTULO 5

IMPLEMENTAÇÃO DO GCS NO AMBIENTE SIPS

Este capítulo trata da implementação do GCS no SIPS. Para isto, inicialmente, é apresentada a maneira pela qual as informações de GCS são armazenadas na Base Sistema. Para a manipulação destas informações novas primitivas, denominadas de primitivas de GCS foram geradas; suas definições são apresentadas na seção 5.2 e suas especificações na seção 5.3. Uma ferramenta denominada Gerenciador de Configuração foi gerada para que o usuário do ambiente SIPS pudesse utilizar-se destes novos recursos introduzidos; a especificação das funções desta ferramenta está descrita na seção 5.4 e a especificação de seus módulos está descrita na seção 5.5.

5.1 - Armazenamento das Informações de GCS na Base de Dados do SIPS

No modelamento da metodologia GCS para o SIPS apresentado na Seção 4.4, as instâncias de um ICS aparecem como tipos de sinônimos para um ICS. Mas para que o controle de versões e de configurações de ICSs pudesse ser efetivamente efetuado no SIPS foi necessário alterar algumas estruturas internas da Base Sistema para introduzir o conceito de instâncias de ICS, como será apresentado a seguir.

Dentre os componentes da Base Sistema já citados na Seção 3.4, os que devem ter suas estruturas internas alteradas são aqueles relacionados especificamente com as informações sobre objetos: Identificador e Objeto, uma vez que ICS é um tipo de objeto.

Um objeto armazenado na Base Sistema possui sempre um registro do tipo Identificador, um registro do tipo Objeto e tantos registros adicionais do tipo Identificador quantos forem os sinônimos associados a este objeto.

Para implementar o conceito de versões de ICSs foi alterada esta característica quando o objeto em questão for um ICS. Um ICS passa então a ter um registro do tipo Identificador que o identifique e tantos registros do tipo Objeto quantos forem as alternativas, revisões e variações relacionados com este ICS. Ainda, para cada alternativa e versão gerada existirá um registro do tipo Identificador associado, mas este armazena o sinônimo do ICS, ou seja, o identificador para que cada alternativa ou versão possa ser referenciada univocamente.

Um ICS ao ser gerado na Base Sistema contém somente um registro de cada tipo (Identificador e Objeto). Cada nova alternativa gerada resulta na criação de um novo registro do tipo Objeto. Um registro do tipo Objeto de um ICS passa a armazenar todas as informações relacionadas com o histórico desta alternativa e posteriormente o histórico da versão, uma vez que uma versão é originada sempre a partir de uma alternativa. Ao se criar uma alternativa deve ser dado um nome que a identifica. Dado que podem ser criadas várias alternativas para uma mesma versão de ICS então, ao se criar uma versão, um novo nome deve ser dado para a versão, e o nome da alternativa neste momento deixa de existir.

Os atributos contidos em um registro do tipo Objeto que auxiliam a implementação das funções do GCS no SIPS, são:

- data da criação da alternativa;
- data da criação da versão;
- estado do objeto (1 = em edição, 2 = efetivo, 3 = congelado);
- código do autor da criação da alternativa;
- código do autor da criação da versão.

Para se distinguir as instâncias associadas a um ICS, devem ser observadas algumas características:

- uma alternativa possui seu estado igual a um (em edição), se está em desenvolvimento, ou igual a dois (efetivo) se está em estado transitório para ser congelada. Para se examinar as várias alternativas que constituem uma versão deve-se seguir um apontador criado especialmente para este propósito, de forma a criar uma lista de alternativas associadas com cada versão de um ICS;
- uma revisão ou variação de um ICS apresenta seu estado igual a três (congelado). Para se obter as várias revisões associadas com uma variação de um ICS, deve-se percorrer a lista de

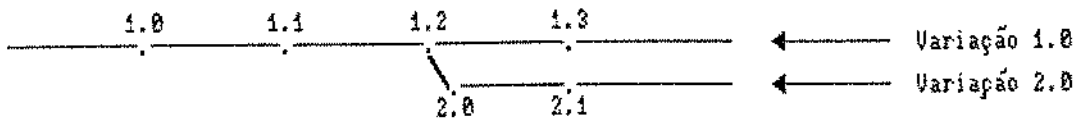
relacionamentos do tipo **é_uma_revisão_de**. Da mesma forma que para revisão, para se obter todas as variações associadas com um ICS, percorre-se a lista de relacionamentos **é_uma_variação_de** associada com este ICS. Vale ressaltar que o último objeto destino da lista de relacionamentos **é_uma_revisão_de** corresponde a última revisão desta variação de ICS, ou seja, a revisão mais atualizada associada a esta variação.

Para efeito de implementação, a primeira versão de um ICS é obtida através de um apontador que se origina do registro do tipo Identificador e aponta para um registro do tipo Objeto associada à versão em questão.

Um outro apontador que se origina do registro do tipo Identificador se faz necessário para que possa ser armazenada qual foi a última instância do ICS acessada pelo usuário. A partir do posicionamento em uma das instâncias do ICS, este registro torna-se a instância corrente do ICS para efeito de operações a serem realizadas sobre o mesmo.

O armazenamento de relacionamentos associados a um objeto na Base Sistema foi implementado em forma de lista ligada de forma que, para cada tipo de relacionamento associado ao objeto, os relacionamentos vão sendo posicionados na lista de acordo com a ordem de entrada. Utilizando-se desta característica, ao se percorrer as listas de relacionamentos **é_uma_revisão_de** e **é_uma_variação_de** obtém-se todas as revisões associadas a uma variação e todas as variações de um ICS, respectivamente, exatamente na sequência em que elas foram geradas.

No exemplo abaixo, a revisão 1.3 é a revisão mais atualizada da variação 1.0 e a revisão 2.1 é a revisão mais atualizada da variação 2.0. Como variações deste ICS, podem conviver as variações 1.0 com sua revisão mais atualizada 1.3 e a 2.0 com sua revisão mais atualizada 2.1.



A Figura 5.1 apresenta como as versões de um ICS do exemplo descrito acima ficam armazenadas na base de dados do SIPS. Nesta figura, a revisão 1.3 apresenta duas alternativas, estando uma em edição e outra efetivada e a revisão 2.1 apresenta uma alternativa em edição. Estas alternativas podem se tornar uma variação ou uma revisão dependendo do motivo pelas quais elas

estão sendo geradas.

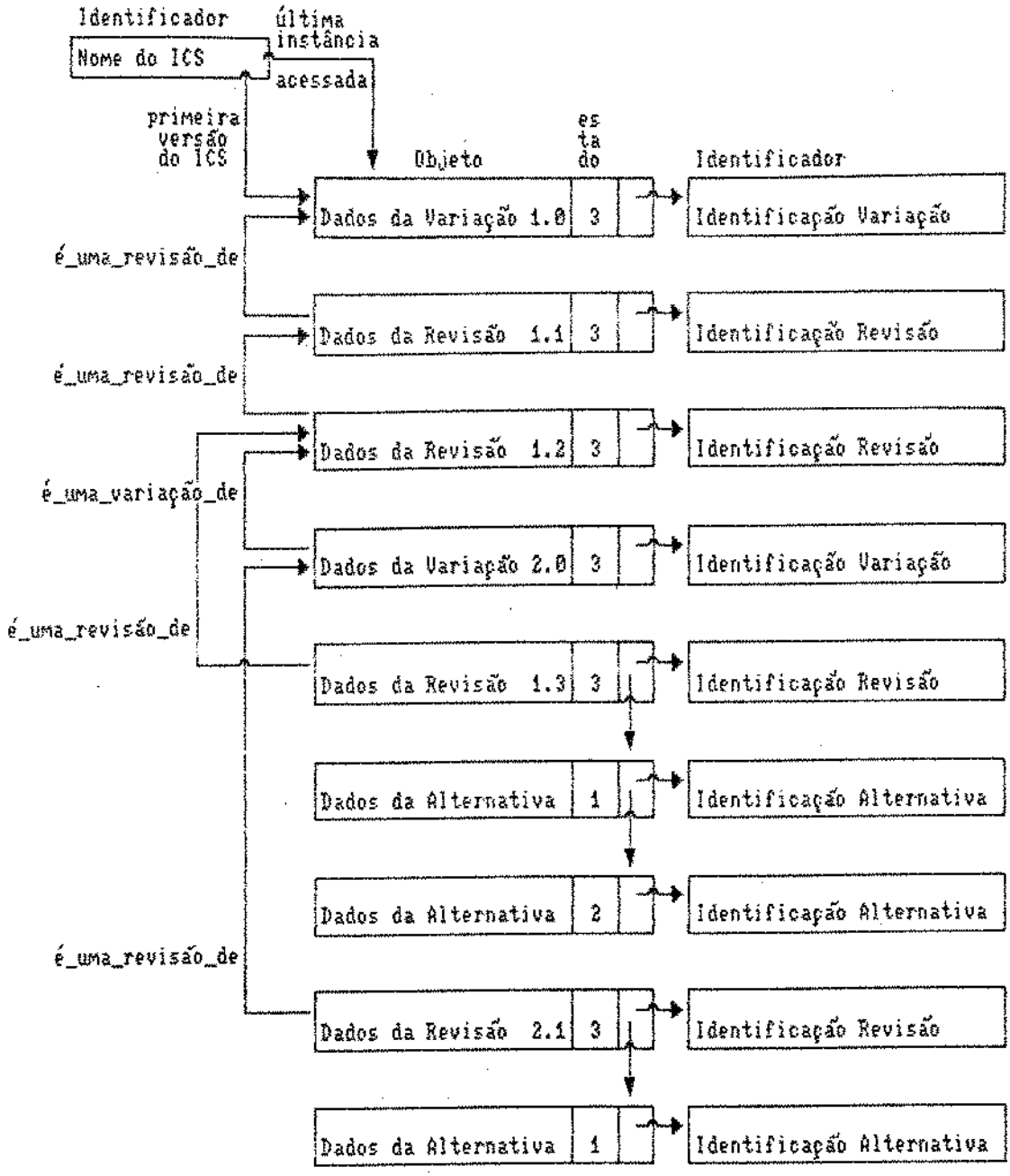


Figura 5.1 - Esquema de Armazenamento de Instâncias de um ICS na Base Sistema

Desta forma, é possível gerar um histórico que mantém a estrutura bem como o conteúdo das modificações de cada ICS. Assim, cada ICS consiste de uma árvore de versões juntamente com todos os dados que definem as modificações ao longo do tempo.

Uma variação é criada a partir de uma versão colocando-se o relacionamento **é_uma_variação_de** entre a variação que está sendo criada e a versão que a originou. Uma revisão é criada a partir de uma revisão pertencente a mesma variação colocando se o relacionamento **é_uma_revisão_de** entre a revisão que está sendo criada e a revisão que a originou.

O relacionamento ICS **é_composto_por** objetos de uma metodologia (apresentado na Figura 4.3) é implementado no SIPS usando se o conceito de classes. Então, um ICS constringe objetos que pertencem a classe hospedeira de objetos da metodologia específica que está sendo modelada juntamente com a metodologia GCS.

Como visto na Seção 3.6, um objeto constritor possui um registro do tipo classe associado, que contém os endereços dos conjuntos de objetos constritos por ele. Desta forma, uma instância de um ICS por se tratar de um objeto constritor para a metodologia GCS, possui um registro do tipo classe associado ao mesmo.

Portanto, a operação de se criar um ICS ou criar uma alternativa para este ICS consiste na criação do objeto do tipo ICS (registro do tipo Identificador e um registro do tipo Objeto) mais um registro do tipo de classe que contém o endereço de conjuntos de objetos constritos pelo ICS.

Esquematicamente, a Figura 5.2 ilustra o armazenamento de informações de uma instância de um ICS.

Observe que, se na abertura de uma instância de um ICS, estiver associado a ela mais de um conjunto de objetos constritos, então no registro de classe associado a esta instância haverá mais de um endereço de conjuntos de objetos constritos pela mesma. Exemplificando, se para um mesmo ICS for definido que este constringe um conjunto de objetos constritos referentes ao detalhamento do fluxo de dados e um outro conjunto de objetos constritos referentes ao detalhamento do fluxo de controle, então associado a uma instância deste ICS existirá no registro de classe o endereço destes dois conjuntos de objetos constritos.

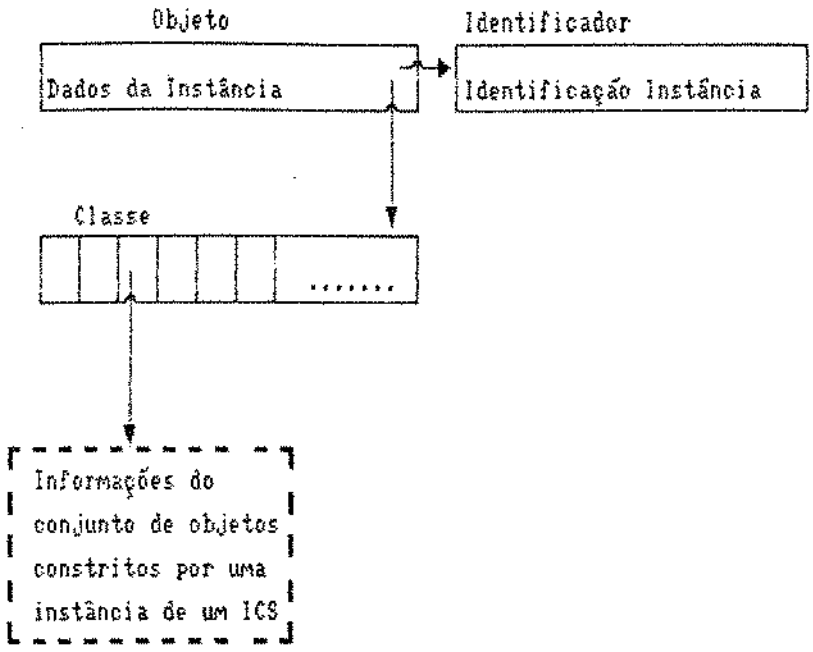


Figura 5.2 - Armazenamento de Informações de Uma Instância de Um ICS

5.2 - Definição das Primitivas de GCS

As alterações efetuadas na Base Sistema descritas na Seção 5.1 tornou necessária a existência de novas primitivas para que o projetista pudesse utilizar-se desses recursos.

As primitivas de GCS descritas nesta seção são voltadas exclusivamente para manipular alternativas, revisões e variações de ICSs.

Estas primitivas podem ser classificadas em duas categorias, de acordo com o usuário a que se destina. Para tanto, existem primitivas para uso do projetista que desenvolve uma ferramenta e primitivas que são utilizadas exclusivamente pela ferramenta gerenciador de configuração de software.

As primitivas manipulam as instâncias de um ICS. Então, existem primitivas que permitem: criar e recuperar um ICS, uma

alternativa ou uma versão de um ICS; eliminar uma alternativa de uma versão de um ICS; alterar os estados de uma alternativa de modo que ela passe do estado em edição para o estado efetivo e, posteriormente, para o estado congelado ou voltar ao estado em edição; e relacionar os ICS que estabelecem uma configuração básica parcial ou global. Estas primitivas foram criadas como complementares às primitivas que manipulam a base de dados do SIPS. Desta forma, elas devem atuar conjuntamente. As operações sobre os objetos, propriedades e relacionamentos de uma alternativa ou de uma versão são efetuadas através das primitivas já existentes do núcleo do SIPS.

A criação de uma alternativa de uma versão de um ICS é feita através da duplicação de todos os dados (objetos, propriedades e relacionamentos) da versão do ICS. Isto é feito devido à forma como a base de dados do SIPS foi implementada inicialmente. A priori, não nos parece a melhor solução, uma vez que dados redundantes permanecem na Base Sistema. Soluções melhores existem, como é o caso da implementação de versões armazenando-se apenas as diferenças entre as mesmas; mas, pela maneira como a base de dados do SIPS se encontra implementada, esta solução não é viável.

A implementação do modelo adotada para este trabalho não se preocupa com o armazenamento externo do histórico dos ICSs. Desta forma, todas as versões geradas para um ICS permanecem na Base Sistema. Por não existir uma área de trabalho onde as alternativas possam ser desenvolvidas, estas também permanecem na Base Sistema. Desta forma, todas as instanciações geradas de um ICS convivem na Base Sistema podendo-se tanto obter o registro de estado atual de um ICS como o seu histórico.

Apesar de poder existir mais de uma alternativa para uma mesma versão de um ICS, sempre haverá uma alternativa ou versão que será considerada como corrente, a última alternativa ou versão acessada.

Deve sempre existir uma identificação para uma alternativa uma revisão ou variação, a ser fornecida no ato de sua criação.

As primitivas para o projetista de uma ferramenta são:

- Cria uma alternativa de uma versão de um ICS;
- Efetiva uma alternativa de uma versão de um ICS;
- Elimina uma alternativa de uma versão de um ICS;
- Posiciona na primeira alternativa de uma versão de um ICS;

- Avança para a próxima alternativa de uma versão de um ICS;

- Posiciona na primeira revisão ou variação de um ICS;

- Avança para a próxima revisão ou variação de um ICS;

- Posiciona em uma dada versão de um ICS;

- Posiciona em uma dada alternativa de um ICS.

Como pode-se notar, o projetista de uma ferramenta pode basicamente manipular somente alternativas, dado que alternativas são geradas pelo usuário da ferramenta durante o desenvolvimento do software. Com isso pode-se notar também que alternativas são partes do sistema em desenvolvimento que, apesar de estarem na Base Sistema, ainda não estão sob controle do GCS, embora este reconheça a sua existência. O gerenciador de configuração de software somente atua a partir de alternativas que o usuário efetivou, pois estas são os elementos usados pelo gerenciador de configuração de software para gerar novas versões de ICSs.

O usuário da ferramenta tem permissão somente para consultar uma revisão ou uma variação de um ICS.

As primitivas a serem usadas pelo projetista de ferramentas, são aquelas que devem ser incluídas nas ferramentas de metodologias sobre as quais o GCS deve atuar.

As primitivas da ferramenta Gerenciador de Configuração de Software são:

- Cria um ICS;

- Cria uma revisão ou variação de um ICS;

- Muda o estado da alternativa de efetivo para em edição;

- Coloca em um vetor todos os ICS que estabelecem uma configuração básica parcial composta pelo ICS corrente;

- Coloca em um vetor todos os ICSs que evoluem do ICS corrente e que são configurações básicas parciais.

5.3 - Especificação das Primitivas de GCS

A seguir, são apresentados os algoritmos das primitivas

descritas na seção anterior. A descrição de cada uma das primitivas consiste de comentários para facilitar o seu entendimento, caso sejam necessários, e do pseudo-código correspondente.

Cria uma Alternativa para a Versão Corrente de um ICS

Esta primitiva foi desenvolvida usando os seguintes recursos do núcleo do SIPS:

- primitivas do núcleo do SIPS para manipular as informações do conjunto de objetos constritos da versão do ICS que será duplicado, e para manipular informações da alternativa em desenvolvimento. Desta forma, usa-se bastante o conceito de registro de estado corrente e registro de classe corrente;

- arquivo de seleção para armazenar todos os objetos que pertencem ao ICS e que deverão ser duplicados;

- salvar e restaurar registro de estado e de classe corrente. Para cada classe hospedeira é possível ter somente um conjunto de objetos constritos abertos ao mesmo tempo e só é possível ter um objeto corrente ao mesmo tempo. Para o caso de duplicação de informações de um ICS para uma alternativa, é necessário trabalhar com dois diferentes objetos constritores para uma mesma classe hospedeira e dois objetos correntes ao mesmo tempo. Estes recursos serão utilizados cada vez que deve ser lida uma informação de um objeto da classe constrita pelo ICS e deve ser gravada uma informação no objeto da classe constrita pela alternativa;

- marcação de objetos. É colocado uma marca no objeto da classe ICS que corresponde ao código do objeto na classe alternativa, pois esta marca será necessária para a duplicação dos relacionamentos e propriedades deste objeto. Uma outra marca é colocada para sinalizar que o objeto já foi duplicado. Esta sinalização é necessária durante a duplicação de relacionamentos para determinar se o relacionamento associado a este objeto já foi colocado ou não.

Esta primitiva pressupõe que, ao ser chamada, já se esteja posicionado numa versão de um ICS; caso contrário retorna uma mensagem de erro.

- verifica se existe objeto corrente
- verifica se objeto corrente é uma versão de ICS
- cria um registro do tipo objeto para a alternativa
- salva o código da versão corrente do ICS
- liga a alternativa que está sendo criada com a versão corrente do ICS

- assinala usuário corrente como autor da criação da alternativa
- assinala a data do sistema como a data da criação da alternativa
- assinala o estado da alternativa sendo criada como em edição
- identifica as classes hospedeiras que a versão do ICS constringe para sua duplicação
- cria um registro de classe para a alternativa
- posiciona no primeiro objeto da versão do ICS a ser duplicado

- **enquanto** houver objetos na versão do ICS
  faça (duplica todos os objetos da versão do ICS)
  - obtém o código do tipo do objeto
  - obtém o nome do objeto
  - coloca o objeto na alternativa
  - guarda o objeto no arquivo de seleção
  - avança para próximo objeto da versão do ICS

- obtém um objeto do arquivo de seleção

- **enquanto** houver objeto no arquivo de seleção
  faça (duplica todos os relacionamentos, comentários, atributos e sinônimos de todos os objetos da versão do ICS)
  - o objeto obtido do arquivo de seleção passa a ser o objeto corrente
  - posiciona no 1º sinônimo, 1º relacionamento, 1º atributo, 1º comentário do objeto corrente

  - duplica todos os sinônimos do objeto corrente
    - obtém dados do 1º sinônimo
    - **enquanto** houver sinônimos no objeto corrente
      faça
      - não duplica os sinônimos do tipo variação e revisão
      - coloca o sinônimo na alternativa
      - avança para próximo sinônimo do objeto corrente
      - obtém dados do sinônimo posicionado

  - duplica todos os atributos do objeto corrente
    - obtém dados do 1º atributo
    - **enquanto** houver atributos no objeto corrente
      faça
      - coloca o atributo na alternativa
      - avança para próximo atributo do objeto corrente
      - obtém dados atributo posicionado

  - duplica todos os comentários do objeto corrente
    - obtém dados da 1ª linha do 1º comentário
    - **enquanto** houver comentários no objeto corrente
      faça
      - coloca a linha de comentário na alternativa
      - avança para próxima linha ou para próximo comentário do objeto corrente
      - obtém dados da linha de comentário posicionada

  - duplica todos os relacionamentos do objeto corrente
    - obtém dados do 1º relacionamento

- **enquanto** houver relacionamentos no objeto corrente
  faça
  - não duplica os relacionamentos do tipo é_uma_variação_de e é_uma_revisão_de
  - verifica se o objeto destino pertence a mesma versão de ICS
    - **se** objeto destino pertence ao ICS que está sendo duplicado
      então
      - obtém código objeto destino novo
      senão (objeto destino não pertence a mesma versão de ICS e não será duplicado)
        - objeto destino na alternativa é igual ao objeto destino da versão do ICS
  - verifica se já foi colocado este relacionamento
    - **se** ainda não foi colocado este relacionamento
      então
      - coloca o relacionamento na alternativa

- duplica todos atributos do relacionamento corrente
  - posiciona no 1º atributo do relacionamento
  - obtém dados do 1º atributo de relacionamento
  - **enquanto** houver atributo de relacionamento
    faça
    - posiciona no relacionamento da alternativa
    - coloca o atributo de relacionamento no relacionamento posicionado
    - avança para próximo atributo do relacionamento corrente
    - obtém dados atributo de relacionamento posicionado
- avança para próximo relacionamento do objeto corrente
- obtém dados do relacionamento posicionado

- sinaliza o objeto duplicado do ICS que já foi colocado os seus relacionamentos
- obtém o próximo objeto do arquivo de seleção

- finalização da duplicação
- atualiza a área global para retornar dentro da alternativa
- atualiza a quantidade de alternativas para a versão de ICS que está sendo criada esta alternativa
- atualiza a quantidade de versões abertas para a configuração básica que a versão de ICS, cuja alternativa está sendo criada, constitui
- cria um registro do tipo identificador para guardar a identificação da alternativa
- retorna o código da alternativa criada
- retorna

Efetiva a Alternativa Corrente da Versão Corrente de um ICS

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionada em uma alternativa; caso contrário, retorna uma mensagem de erro.

- verifica se existe alternativa corrente
- efetiva a alternativa - passando o estado da alternativa de 1 para 2
- retorna

Elimina a Alternativa Corrente da Versão Corrente de um ICS

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionada em uma alternativa; caso contrário, retorna uma mensagem de erro. Utiliza-se das primitivas do núcleo do SIPS para eliminar as informações associadas à alternativa que será eliminada.

- verifica se existe alternativa corrente
- elimina todos os relacionamentos da alternativa
- elimina todos os atributos da alternativa
- elimina todos os comentários da alternativa
- elimina o sinônimo identificador da alternativa
- elimina o registro de classe relacionado com a alternativa
- atualiza o apontador de alternativa, revisão, ou variação corrente que passa a apontar para a primeira alternativa, revisão ou variação do ICS
- atualiza o apontador que liga a lista de alternativas da versão corrente do ICS
- decrementa a quantidade de alternativas para a versão de ICS que esta alternativa pertencia
- elimina a alternativa
- invalida os apontadores globais, de forma a sinalizar que não tem mais alternativa corrente
- retorna

Posiciona na Primeira Alternativa da Versão Corrente de um ICS

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionada numa versão de um ICS; caso contrário, retorna uma mensagem de erro.

- verifica se existe objeto corrente
- verifica se objeto corrente é uma versão de ICS
- verifica se o ICS possui alternativa
- salva o código da versão corrente do ICS
- obtém a primeira alternativa da versão corrente do ICS
- a alternativa obtida passa a ser a alternativa corrente
- atualiza os apontadores globais
- retorna

Avança para a Próxima Alternativa da Versão Corrente de um ICS

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionada em uma alternativa; caso contrário, retorna uma mensagem de erro.

- verifica se existe alternativa corrente
- avança para a próxima alternativa
- a alternativa obtida passa a ser a alternativa corrente
- atualiza os apontadores globais
- retorna

Posiciona na Primeira Variação ou Primeira Revisão do ICS Corrente Dependendo da Opção de Entrada

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionado em um objeto do tipo ICS; caso contrário, retorna uma mensagem de erro.

Esta primitiva trabalha em conjunto com a primitiva que avança para a próxima revisão ou variação. É a partir da sequência de posicionamento estabelecida no posicionamento na primeira variação ou na primeira revisão de um ICS que é estabelecido o tipo da próxima versão que a rotina de avança deve tentar posicionar.

- verifica se existe objeto corrente
- verifica se objeto corrente é um ICS
- verifica se opção de entrada é válida
  - opção = 0 => variação opção = 1 => revisão
- salva o código do ICS corrente
- verifica se existe revisão ou variação para o ICS
- lê a identificação da versão (sinônimo) para ver se é uma revisão ou variação

- se opção = 0 (procura pela primeira variação)

```
  então
    - verifica se tipo de sinônimo é variação
       - se não é variação
         então
            - posiciona no primeiro relacionamento 'gera_variação'
            - se  não  encontrou o relacionamento
              então
                  - erro: não existe variação para este ICS
                  - retorna
              senão
                  - lê  o  objeto  destino do  relacionamento  que
                    corresponde a variação do ICS
    - atualiza a sequência de posicionamento para que a  primitiva
      de avança posicione na próxima variação
    - atualiza a área global que passa a ter uma variação corrente

- se opção = 1  (procura pela primeira revisão)
  então
    - verifica se tipo de sinônimo é uma revisão
       - se não é revisão
         então
            - posiciona no primeiro relacionamento 'gera_revisão'
            - se  não  encontrou o relacionamento
              então
                  - erro:  não existe revisão para este ICS
                  - retorna
             senão
                  - lê  o  objeto destino  do  relacionamento  que
                    corresponde a revisão do ICS
    - atualiza  a sequência de posicionamento para que a primitiva
      de avança posicione na próxima revisão
    - atualiza a área global que passa a ter uma revisão corrente

- a variação ou revisão posicionada passa a ser a versão corrente
- atualiza os apontadores globais
- retorna
```

Avança para a Próxima Variação ou Revisão do ICS Corrente de Acordo Como Foi Feita a Opção no Posicionamento pela Primitiva Anterior

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionada em uma versão de um ICS, através da primitiva de posicionar na primeira versão do ICS corrente; caso contrário retorna uma mensagem de erro.

Esta primitiva testa se a revisão ou variação posicionada corresponde ao primeiro registro do tipo objeto. Isto porque a primeira revisão ou variação é obtida através de um apontador interno. As demais variações e revisões de um ICS são obtidas via relacionamento **é_uma_variação_de** e **é_uma_revisão_de**, respectivamente.

```
- verifica se foi posicionada na primeira revisão ou variação de
  um ICS
- verifica se a revisão ou variação posicionada corresponde ao
  primeiro registro do tipo objeto
    - se revisão ou variação corresponde ao primeiro registro
      então atualiza variável indicando que é o primeiro
      senão atualiza variável indicando que não é o primeiro
- posiciona na próxima revisão ou variação
    - se sequência de posicionamento corresponde a variação
      então
        - posiciona na variação corrente se esta não é o
          primeiro registro do tipo objeto ou posiciona na
          próxima variação se esta é o primeiro registro do
          tipo objeto
        - se não tem mais variação
          então
            - invalida a sequência de posicionamento
            - invalida a área global que passa a não ter mais
              variação corrente
            - erro: não existe mais variação para este ICS
            - retorna
        - se variação não é o primeiro registro do tipo objeto
          então
            - avança para a próxima variação
                - se não é fim da lista de relacionamentos
                  então
                    - se o objeto destino é uma variação
                      então
                        - objeto destino passa a ser a
                          variação corrente
                      senão (ICS não tem mais variação)
                        - invalida a sequência de posiciona-
                          mento
                        - invalida a área global que passa a
                          não ter mais variação corrente
                        - erro: não existe mais variação para
                          este ICS
                        - retorna
                  senão (fim da lista de relacionamentos - ICS
                         não tem mais variação)
                    - invalida a sequência de posicionamento
                    - invalida a área global que passa a não
                      ter mais variação corrente
                    - erro: não existe mais variação para este
                      ICS
                    - retorna
        - variação passa a ser o objeto corrente
      senão (sequência de posicionamento corresponde a revisão)
        - posiciona na revisão corrente se esta não é o
          primeiro registro do tipo objeto ou posiciona na
          próxima revisão se esta é o primeiro registro do
          tipo objeto
        - se não tem mais revisão
```

```
            então
              - invalida a sequência de posicionamento
              - invalida  a área global que passa a não ter  mais
                revisão corrente
              - erro: não existe mais revisão para este ICS
              - retorna
        - se revisão não é o primeiro registro do tipo objeto
          então
            - avança para a próxima revisão
                - se não é fim da lista de relacionamentos
                  então
                    - se o objeto destino é uma revisão
                      então
                        - objeto  destino  passa  a  ser  a
                          revisão corrente
                      senão (ICS não tem mais revisão)
                        - invalida a sequência de  posiciona-
                          mento
                        - invalida a área global que passa  a
                          não ter mais revisão corrente
                        - erro:  não existe mais revisão para
                          este ICS
                        - retorna
                  senão  (fim da lista de relacionamentos - ICS
                          não tem mais revisão)
                     - invalida a sequência de posicionamento
                     - invalida  a área global que passa a  não
                       ter mais revisão corrente
                     - erro:  não existe mais revisão para este
                              ICS
                   - retorna
        - revisão passa a ser o objeto corrente

- a variação ou revisão posicionada passa a ser a versão corrente
- atualiza apontadores globais
- retorna
```

Posiciona em uma Dada Revisão ou Variação do ICS Corrente Dependendo da Opção de Entrada

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionado em um objeto do tipo ICS; caso contrário, retorna uma mensagem de erro.

Utiliza-se das primitivas de posicionar na primeira versão e avançar para a próxima versão de um ICS descritas nesta seção.

```
- verifica  se o nome da revisão ou variação de entrada é um nome
  válido
- verifica se opção de entrada é válida
  - opção = 0  => variação   opção = 1 => revisão
```

```
- verifica se objeto corrente é um ICS
- posiciona na primeira revisão ou variação do ICS corrente
  dependendo da opção de entrada
- enquanto não encontrar a versão requerida e não for fim da
                  lista de relacionamentos
  faça
    - compara a versão posicionada com a versão requerida
    - se encontrou a versão
      então
        - atualiza a area global que passa a ter essa versão como
          corrente
        - retorna
      senão
        - posiciona na próxima revisão ou variação
(fim da lista de relacionamentos)
- erro: não existe a versão especificada para o ICS corrente
- retorna
```

Posiciona em uma Dada Alternativa da Versão Corrente de um ICS

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionada em uma versão de um ICS; caso contrário, retorna uma mensagem de erro.

Utiliza-se das primitivas de posicionar na primeira alternativa e avançar para a próxima alternativa descritas nesta seção.

```
- verifica se o nome da alternativa de entrada é um nome válido
- verifica se objeto corrente é uma versão de ICS
- posiciona na primeira alternativa da versão corrente de um ICS
- enquanto não encontrar a alternativa requerida e não for fim da
                  lista de alternativas
  faça
    - compara a alternativa posicionada com a alternativa requerida
    - se encontrou a alternativa
      então
        - atualiza a area global que passa a ter essa alterntiva
          como corrente
        - retorna
      senão
        - posiciona na próxima alternativa
(fim da lista de alternativas)
- erro: não existe a alternativa especificada para a versão
        corrente
- retorna
```

Cria um Objeto do Tipo ICS

- cria um registro do tipo identificador para guardar a identificação da alternativa que está sendo gerada
- cria um registro do tipo objeto para a alternativa
- assinala usuário corrente como autor da criação da alternativa
- assinala a data do sistema como a data da criação da alternativa
- assinala o estado da alternativa sendo criada como em edição
- inicializa a quantidade de versões abertas para a configuração básica que este ICS constituirá
- inicializa a quantidade de alternativas para o ICS
- coloca este objeto constrito dentro da classe ICS mais abrangente
- atualiza a área global sinalizando que tem uma alternativa corrente
- retorna o código do ICS criado
- retorna

Congela a Alternativa Corrente que Está em Estado Efetivo, Criando uma Variação ou uma Revisão Dependendo da Opção de Entrada

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionada em uma alternativa com seu estado efetivo; caso contrário, retorna uma mensagem de erro.

- verifica se existe alternativa corrente
- verifica se alternativa está com seu estado efetivo
- verifica se opção de entrada é válida
  - opção = 0 => variação opção = 1 => revisão
- elimina o sinônimo que identifica a alternativa
- cria um registro do tipo identificador (sinônimo) para armazenar a identificação da versão (revisão ou variação) que está sendo gerada
- assinala este sinônimo como identificador da versão que está sendo gerada
- cria o relacionamento é_uma_variação_de ou é_uma_revisão_de dependendo da opção de entrada e se não for a primeira versão do ICS que está sendo criada
- atualiza data da criação da revisão ou variação
- atualiza autor da criação da revisão ou variação
- inicializa a variável que contém a quantidade de alternativas para esta versão que está sendo gerada
- congela a alternativa - passa o estado da alternativa de 2 para 3
- decrementa a quantidade de alternativas para a versão de ICS que esta alternativa pertencia

- decrementa a quantidade de versões abertas do ICS correspondente
- retorna

Retorna a Alternativa Corrente Efetiva de Uma Versão de um ICS para o Estado em Edição

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionada em uma alternativa com seu estado efetivo; caso contrário, retorna uma mensagem de erro.

- verifica se existe alternativa corrente
- verifica se alternativa está com seu estado efetivo
- coloca a alternativa no estado em edição - passa o estado da alternativa de 2 para 1
- retorna

Coloca em um Vetor Todos os ICS que Estabelecem Uma Configuração Básica Parcial composta pelo ICS Corrente

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionado em um objeto do tipo ICS; caso contrário, retorna uma mensagem de erro.

Utiliza-se de primitivas do núcleo do SIPS para posicionar no primeiro objeto do tipo ICS da classe do ICS corrente e para obter o próximo ICS do mesmo tipo do ICS corrente.

- verifica se existe objeto corrente
- verifica se objeto corrente é um ICS
- posiciona no primeiro objeto do tipo ICS da classe do ICS corrente
- **Enquanto** houver ICSs do tipo ICS corrente na classe **faça**
    - verifica a quantidade de versões abertas do ICS posicionado
    - **se** quantidade de versões abertas igual a zero **então**
        - coloca o ICS posicionado no vetor
        - incrementa a quantidade de ICSs no vetor
        - obtém o próximo ICS do mesmo tipo do ICS corrente

      senão
        - erro: ICS corrente não faz parte de uma configuração básica parcial
- retorna

Coloca em um Vetor todos os ICSs que Evoluem do CS Corrente e são Configurações Básicas Parciais

Esta primitiva pressupõe que, ao ser chamada, já esteja posicionado em um objeto do tipo ICS; caso contrário, retorna uma mensagem de erro.

Utiliza-se de primitivas do núcleo do SIPS para percorrer a lista de relacionamentos **evolui_para** do ICS corrente.

- verifica se existe objeto corrente
- verifica se objeto corrente é um ICS
- verifica se ICS corrente estabelece uma configuração básica
- verifica se existe o relacionamento "evolui_para" no ICS corrente
- **Enquanto** houver relacionamento "evolui_para" e não for fim da lista de relacionamentos

  faça
  - verifica se o objeto destino (ICS que evolui do ICS corrente) do relacionamento "evolui_para" estabelece uma configuração básica parcial
    - **se** quantidade de versões abertas do objeto destino é igual a zero

      então
      - coloca o objeto destino no vetor
      - incrementa a quantidade de ICSs no vetor
      - obtém o próximo relacionamento "evolui_para"

      senão
      - erro: ICS corrente não estabelece uma configuração básica global

- retorna

5.4 - Especificação das Funções da Ferramenta Gerenciador de Configuração de Software

A ferramenta Gerenciador de Configuração de Software foi criada com o objetivo de suprir todas as funções necessárias para o desempenho da metodologia GCS no SIPS. Para tal propósito, foram criados vários comandos e subcomandos para esta ferramenta. Os subcomandos, quando vistos sob a ótica da filosofia de desenvolvimento de ferramentas para o SIPS, podem ser encarados como módulos que podem ser acessados por outras ferramentas do SIPS, via supervisor.

Este enfoque nos parece bastante interessante, visto que certos subcomandos devem ser executados pela maioria das ferramentas do SIPS. Usando esta filosofia, estes módulos (subcomandos) podem ser reutilizados pelas ferramentas que deles necessitarem. Como vantagem desta abordagem, vale destacar que o projetista da ferramenta não precisa conhecer detalhes de implementação das primitivas de GCS; basta-lhe ter ciência de que sua ferramenta precisa manipular alternativas e, para isso, utilizar-se dos módulos da ferramenta Gerenciador de Configuração de Software.

Desta forma, a ferramenta Gerenciador de Configuração de Software, por si só, não executa todas das funções necessárias ao GCS, mas executa outras funções básicas de GCS que poderiam estar embutidas nas ferramentas que automatizam outras metodologias no SIPS.

Esta parece ser a melhor solução visto que esta ferramenta deve ser integrada ao ambiente SIPS. Então, o enfoque dado à ferramenta permite suprir necessidades de outras ferramentas no que se refere a GCS para o SIPS, mas utiliza-se de recursos disponíveis no ambiente para suprir necessidades próprias.

A ferramenta Gerenciador de Configuração de Software pode ser esquematizada como apresenta a Figura 5.3, com seus comandos e subcomandos.

A seguir são detalhados os comandos e subcomandos da ferramenta.

Os comandos da ferramenta foram assim classificados pois consistem nas operações que podem ser efetuadas sobre os componentes da metodologia GCS no SIPS.

Cria

O comando **cria** permite criar: um ICS; uma alternativa, revisão ou variação de um ICS; e uma PMS, para alteração de um ICS.

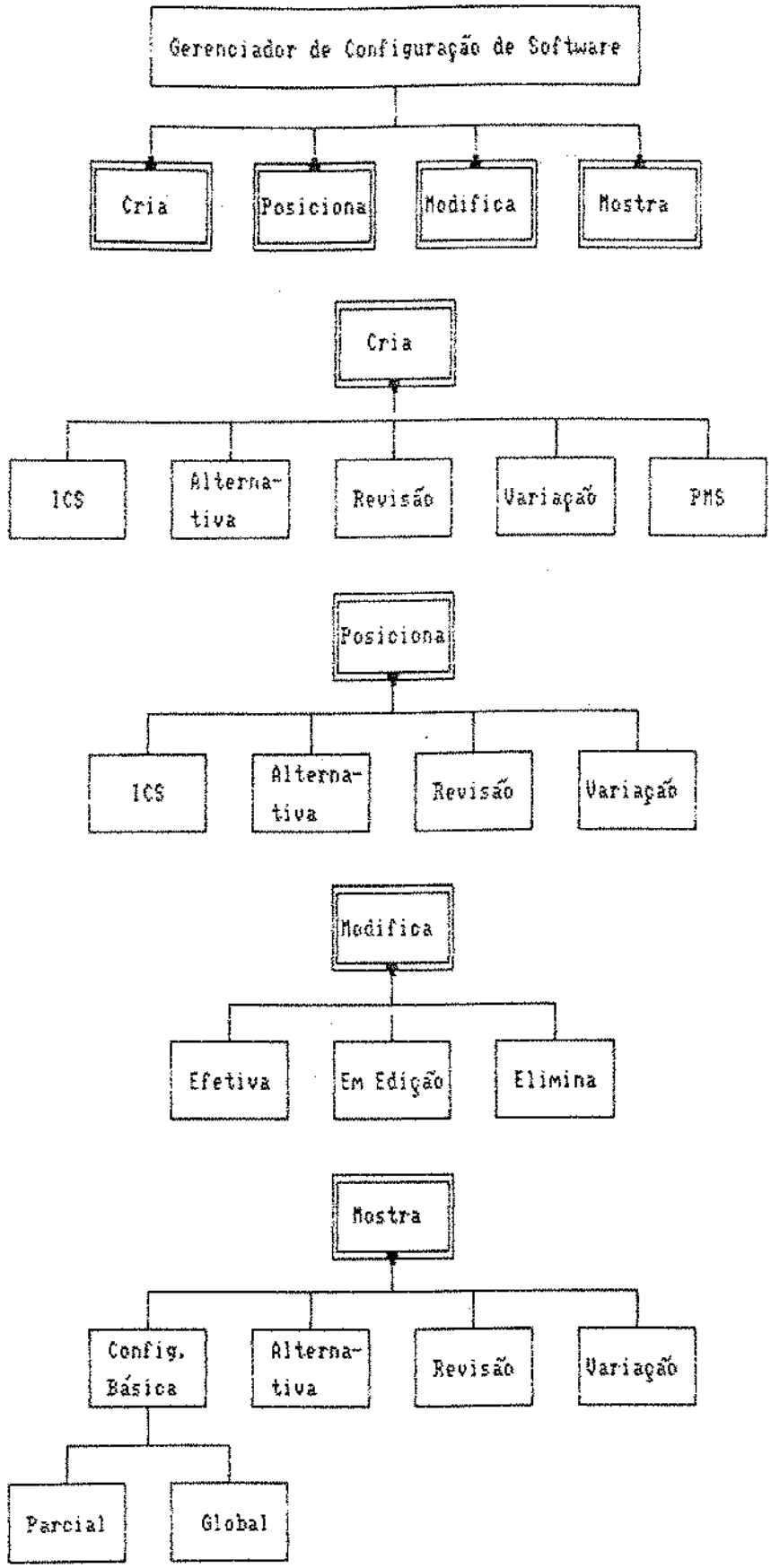


Figura 5.3 - Comandos da Ferramenta Gerenciador de Configuração de Software

O subcomando **cria ICS** é o responsável pela função de identificação de configuração de software. Com este subcomando deve ser associado o objeto a ser identificado univocamente com o ICS criado. Desta forma, é neste subcomando que se deve estabelecer o relacionamento **identifica** entre o objeto da metodologia e o ICS correspondente. É ainda neste subcomando que se estabelece o relacionamento **evolui_para**, entre o ICS que está sendo criado e o ICS contexto corrente; o ICS criado passa a ter o relacionamento **é_uma_evolução_de** com o ICS contexto corrente.

Devem também ser identificados os objetos de outros ICSs que devem ser importados para o ICS que está sendo criado. Esta função não está implementada na versão atual da ferramenta Gerenciador de Configuração de Software.

Cabe observar que, ao se criar um ICS, na verdade está-se criando um objeto do tipo ICS, mas com o estado em edição, ou seja, estabelece-se uma alternativa de um ICS.

O subcomando **cria alternativa** permite criar uma nova alternativa para um ICS já criado pelo subcomando cria ICS. Podem ser criadas alternativas para um ICS recém-criado ou para versões de ICSs, não podendo ser criadas alternativas de alternativas de ICSs.

O subcomando **cria revisão** e **cria variação** permitem criar uma revisão ou uma variação respectivamente para uma alternativa que esteja em seu estado efetivo. Estes subcomandos não se preocupam com o processo da auditagem pelo qual a alternativa deve ter passado antes de um desses subcomandos serem executados.

O subcomando **cria PMS** é o responsável pela criação de uma PMS quando da necessidade de alguma alteração sobre um ICS congelado. No SIPS, os dados de uma PMS podem ser relacionados em um formulário a ser preenchido usando o editor de formulários. Assim, este subcomando somente relaciona a PMS criada com a versão do ICS que deve sofrer a alteração requerida. Os demais dados podem ser preenchidos usando o editor de formulários do SIPS ou qualquer outra ferramenta genérica de entrada de dados do SIPS.

Posiciona

Os subcomandos do comando **posiciona** são bastante importantes para a execução de operações sobre os objetos posicionados; fazem

parte, portanto, do conjunto de módulos acessados por outras ferramentas que desejam efetuar estas operações.

O subcomando **posiciona ICS** permite posicionar em um objeto do tipo ICS para a execução de operações que necessitem de um ICS posicionado. Os subcomandos que necessitam de um ICS posicionado são os do comando mostra para apresentar a configuração básica parcial de que o ICS posicionado faz parte, para apresentar a configuração básica global que o ICS posicionado pode estabelecer e para mostrar todas as revisões ou variações do ICS posicionado.

O subcomando **posiciona alternativa** permite posicionar em uma dada alternativa. Os subcomandos desta ferramenta que necessitam de uma alternativa posicionada são aqueles do comando modifica: efetiva, em edição e elimina, que manipulam exclusivamente alternativas.

Os subcomandos **posiciona revisão** e **posiciona variação** permitem posicionar em uma dada revisão ou variação, respectivamente. Estes subcomandos podem ser usados para posicionar em uma revisão ou variação para a qual se quer criar uma alternativa, ou da qual se quer apresentar todas as alternativas geradas.

Modifica

Os subcomandos do comando **modifica** manipulam exclusivamente alternativas. Estes subcomandos ao serem chamados, portanto, pressupõem que já se esteja posicionado na alternativa sobre a qual vão atuar. Os subcomandos do modifica podem ser acessados por outras ferramentas do SIPS, porque realizam atualizações somente em alternativas.

O subcomando **efetiva** permite efetivar uma alternativa cujo desenvolvimento o projetista terminou e que deve ser colocado sob o GCS. é a partir deste ponto que a alternativa passa a merecer atenção do GCS devendo, portanto, passar pelo crivo da auditagem.

O subcomando **em edição** permite retornar ao estado em edição a alternativa que esteja em estado efetivo e que não passou pelo crivo da auditagem devendo, portanto, ser alterada novamente.

O subcomando **elimina** permite eliminar uma alternativa qualquer de um ICS.

Mostra

O comando **mostra** apresenta de uma forma resumida somente os elementos do GCS que são gerados ao longo do desenvolvimento de um software. Não detalha o conteúdo específico de cada um dos elementos uma vez que são específicos de cada metodologia. Desta forma, procura-se apresentar através deste comando as configurações básicas parciais e globais que se estabelecem no sistema, bem como para cada ICS do sistema, as suas revisões, variações e alternativas geradas. Com isso procura cobrir uma parte da função de administração de estado do GCS.

Maiores detalhes de cada elemento não são gerados por esta ferramenta, uma vez que estas informações podem ser obtidas por outras ferramentas de saída do SIPS. O uso coordenado da ferramenta Gerenciador de Configuração de Software juntamente com outras ferramentas de saída do SIPS exercem a função de administração de estado.

Para que os subcomandos do mostra sejam executados ao serem chamados, já se deve estar posicionado em um objeto do tipo ICS.

O subcomando **mostra configuração básica parcial** relaciona todos os ICSs que, juntamente com o ICS corrente, estabelecem uma configuração básica parcial.

O subcomando **mostra configuração básica global** relaciona todos os ICSs que evoluem do ICS corrente de forma que seja estabelecida uma configuração básica global pelo ICS corrente.

O subcomando **mostra revisão** apresenta todas as revisões geradas para um dado ICS.

O subcomando **mostra variação** apresenta todas as variações geradas para um dado ICS.

O subcomando **mostra alternativa** apresenta todas as alternativas geradas para uma dada versão de ICS.

5.5 - Especificação dos Módulos da Ferramenta Gerenciador de Configuração de Software

A ferramenta Gerenciador de Configuração de Software deve ser mais uma das ferramentas que deve ser incorporada ao ambiente SIPS. Desta forma, ela utiliza-se dos recursos disponíveis do núcleo do SIPS. Utiliza-se também da interface homem/máquina padronizada para este ambiente. A forma de utilização da interface apresentada no algoritmo da ferramenta está simplificada somente aos comandos necessários para o seu entendimento.

Os comandos e subcomandos implementados contêm principalmente as primitivas de GCS. Erros são tratados, mas para melhor apresentação, estes estão omitidos nos algoritmos.

Os algoritmos dos comandos do Gerenciador de Configuração de Software, apresentados na seção anterior, estão descritos abaixo na forma de módulos. Cada subcomando da ferramenta corresponde exatamente a um módulo.

Módulo Gerenciador de Configuração de Software

- apresenta os comandos da ferramenta para que o usuário da ferramenta selecione um dos comandos
- **caso** o usuário selecione o comando cria
  - chama o módulo CRIA
- **caso** o usuário selecione o comando posiciona
  - chama o módulo POSICIONA
- **caso** o usuário selecione o comando modifica
  - chama o módulo MODIFICA
- **caso** o usuário selecione o comando mostra
  - chama o módulo MOSTRA
- **caso** o usuário deseja sair da ferramenta
  - retorna ao SIPS principal para selecionar outra ferramenta

Módulo CRIA

- apresenta os subcomandos do módulo CRIA para que o usuário da ferramenta selecione um destes subcomandos
- **caso** o usuário selecione o subcomando ICS
  - chama o módulo CRIA-ICS
- **caso** o usuário selecione o subcomando alternativa
  - chama o módulo CRIA-ALTERNATIVA

- **caso** o usuário selecione o subcomando revisão
  - chama o módulo CRIA-REVISÃO
- **caso** o usuário selecione o subcomando variação
  - chama o módulo CRIA-VARIAÇÃO
- **caso** o usuário selecione o subcomando PMS
  - chama o módulo CRIA-PMS
- **caso** o usuário deseja sair do comando CRIA
  - retorna ao módulo Gerenciador de Configuração de Software para selecionar outro comando da ferramenta

Módulo CRIA-ICS

- apresenta as palavras do tipo "considere" para que o usuário identifique os conjuntos de objetos constritos que o ICS a ser criado deve constringir
- pede o nome do objeto que deve ser identificado pelo ICS que está sendo criado
- verifica se o objeto já está na Base Sistema
- **se** não existe o objeto na Base Sistema
  então
  - pede o tipo do objeto para que possa ser criado
  - coloca o objeto especificado na Base Sistema
- obtém o tipo de ICS que identifica o tipo do objeto especificado
- cria o objeto do tipo ICS com o nome "ICS-(nome do objeto especificado)"
- estabelece o relacionamento "identifica" entre o ICS e o objeto especificado
- cria o registro de classe correspondente ao ICS criado
- estabelece o relacionamento "é_uma_evolução_de" entre o ICS criado e o ICS que constringe o ICS criado
- retorna ao módulo CRIA

Módulo CRIA-ALTERNATIVA

- pede o nome da alternativa que será criada
- cria a alternativa para a versão do ICS corrente
- retorna ao módulo CRIA

Módulo CRIA-REVISÃO

- pede o nome da revisão que será criada
- cria a revisão da alternativa corrente
- retorna ao módulo CRIA

Módulo CRIA-VARIAÇÃO

- pede o nome da variação que será criada
- cria a variação da alternativa corrente
- retorna ao módulo CRIA

Módulo CRIA-PMS

- pede o nome da PMS que será criada
- coloca o objeto PMS na Base Sistema
- pede a identificação do ICS e da versão do ICS que será modificada pela PMS que está sendo gerada
- posiciona na versão do ICS que será modificada pela PMS
- verifica se a versão posicionada é a última revisão de uma variação de um ICS
- estabelece o relacionamento "modifica" entre a PMS gerada e a versão do ICS que será modificada
- retorna ao módulo CRIA

Módulo POSICIONA

- apresenta os subcomandos do módulo POSICIONA para que o usuário da ferramenta selecione um destes subcomandos
- **caso** o usuário selecione o subcomando ICS
  - chama o módulo POSICIONA-ICS
- **caso** o usuário selecione o subcomando alternativa
  - chama o módulo POSICIONA-ALTERNATIVA
- **caso** o usuário selecione o subcomando revisão
  - chama o módulo POSICIONA-REVISÃO
- **caso** o usuário selecione o subcomando variação
  - chama o módulo POSICIONA-VARIAÇÃO
- **caso** o usuário deseja sair do comando POSICIONA
  - retorna ao módulo Gerenciador de Configuração de Software para selecionar outro comando da ferramenta

Módulo POSICIONA-ICS

- pede o tipo do ICS a ser posicionado
- pede o nome do ICS a ser posicionado
- posiciona no ICS especificado tornando-o como objeto corrente e ICS corrente para as futuras operações que necessitem de um ICS posicionado
- retorna ao módulo POSICIONA

Módulo POSICIONA-ALTERNATIVA

- pede o nome da alternativa a ser posicionada
- posiciona na alternativa especificada tornando-a como objeto corrente e alternativa corrente para as futuras operações que necessitem de uma alternativa posicionada
- retorna ao módulo POSICIONA

Módulo POSICIONA-REVISÃO

- pede o nome da revisão a ser posicionada
- posiciona na revisão especificada tornando-a como objeto corrente e revisão corrente para as futuras operações que necessitem de uma revisão posicionada
- retorna ao módulo POSICIONA

Módulo POSICIONA-VARIAÇÃO

- pede o nome da variação a ser posicionada
- posiciona na variação especificada tornando-a como objeto corrente e variação corrente para as futuras operações que necessitem de uma variação posicionada
- retorna ao módulo POSICIONA

Módulo MODIFICA

- apresenta os subcomandos do módulo MODIFICA para que o usuário da ferramenta selecione um destes subcomandos
- **caso** o usuário selecione o subcomando efetiva
  - chama o módulo EFETIVA
- **caso** o usuário selecione o subcomando em edição
  - chama o módulo EM-EDIÇÃO
- **caso** o usuário selecione o subcomando elimina
  - chama o módulo ELIMINA
- **caso** o usuário deseja sair do comando MODIFICA
  - retorna ao módulo Gerenciador de Configuração de Software para selecionar outro comando da ferramenta

Módulo EFETIVA

- efetiva a alternativa corrente
- retorna ao módulo MODIFICA

Módulo EM-EDIÇÃO

- retorna a alternativa corrente para o estado em edição
- retorna ao módulo MODIFICA

Módulo ELIMINA

- elimina a alternativa corrente
- retorna ao módulo MODIFICA

Módulo MOSTRA

- apresenta os subcomandos do módulo MOSTRA para que o usuário da ferramenta selecione um destes subcomandos
- **caso** o usuário selecione o subcomando conf-básica
  - chama o módulo CONF-BÁSICA
- **caso** o usuário selecione o subcomando alternativa
  - chama o módulo MOSTRA-ALTERNATIVA
- **caso** o usuário selecione o subcomando revisão
  - chama o módulo MOSTRA-REVISÃO
- **caso** o usuário selecione o subcomando variação
  - chama o módulo MOSTRA-VARIAÇÃO
- **caso** o usuário deseja sair do comando MOSTRA
  - retorna ao módulo Gerenciador de Configuração de Software para selecionar outro comando da ferramenta

Módulo CONF-BÁSICA

- apresenta os subcomandos do módulo CONF-BÁSICA para que o usuário da ferramenta selecione um destes subcomandos
- **caso** o usuário selecione o subcomando parcial
  - chama o módulo CONF-PARCIAL
- **caso** o usuário selecione o subcomando global
  - chama o módulo CONF-GLOBAL
- **caso** o usuário deseja sair do comando CONF-BÁSICA

- retorna ao módulo MOSTRA para selecionar outro subcomando do módulo mostra

Módulo CONF-PARCIAL

- coloca em um vetor todos os ICSs que constituem uma configuração básica parcial composta pelo ICS corrente
- mostra na tela todos os ICSs contidos no vetor
- retorna ao módulo CONF-BÁSICA

Módulo CONF-GLOBAL

- coloca em um vetor todos os ics's que evoluem do ICS corrente e que constituem configurações básicas parciais
- mostra na tela o ICS corrente que estabelece a configuração básica global
- mostra os ICSs do vetor como as configurações básicas parciais da configuração básica global estabelecida pelo ICS corrente
- retorna ao módulo CONF-BÁSICA

Módulo MOSTRA-ALTERNATIVA

- posiciona na primeira alternativa da versão corrente de um ICS
- **Enquanto** houver alternativas para a versão corrente de um ICS **faça**
  - obtém o nome da alternativa corrente
  - mostra a alternativa corrente na tela
  - posiciona na próxima alternativa da versão corrente de um ICS
- retorna ao módulo MOSTRA

Módulo MOSTRA-REVISÃO

- posiciona na primeira revisão do ICS corrente
- **Enquanto** houver revisões para o ICS corrente **faça**
  - obtém o nome da revisão corrente
  - mostra a revisão corrente na tela
  - posiciona na próxima revisão do ICS corrente
- retorna ao módulo MOSTRA

Módulo MOSTRA-VARIAÇÃO

- posiciona na primeira variação do ICS corrente
- **Enquanto** houver variações para o ICS corrente
  faça
    - obtém o nome da variação corrente
    - mostra a variação corrente na tela
    - posiciona na próxima variação do ICS corrente
- retorna ao módulo MOSTRA

CAPÍTULO 6

CONCLUSÕES

6.1 - Considerações Gerais

A disciplina GCS surgiu da necessidade de se controlar e acompanhar o desenvolvimento de um sistema de software. Este, muitas vezes não se comportava conforme o esperado, seu custo ultrapassava em muito o estimado e normalmente seu desenvolvimento requeria mais tempo que o estipulado no cronograma.

A aplicação desta disciplina impõe um certo nível de sistematização em uma tarefa que, até alguns anos atrás, era considerada quase uma arte: a de desenvolver sistemas de software.

Os benefícios gerados por esta disciplina, no entanto, requerem um investimento de tempo e dinheiro gastos com as atividades de GCS. A experiência tem demonstrado, porém, que estes investimentos são muito menos custosos do que as tentativas de correção do produto final.

O Gerenciamento de Configuração de Software é uma disciplina que deve ser aplicada ao longo do ciclo de vida de um software.

A automatização desta disciplina nos parece a fórmula para garantir que ela possa de fato vir a ser utilizada em desenvolvimento de projetos. Pois, sem um suporte computacional adequado, é uma tarefa trabalhosa e que requer tempo, sendo portanto negligenciada pelos projetistas.

Atualmente, existem poucas ferramentas implementadas que auxiliam a disciplina de GCS. Ferramentas ainda não integradas a um ambiente integrado de desenvolvimento de software, como por exemplo, RCS [29] e SCCS [22] existentes para o Unix, automatizam

o processo de controle de versões de programas.

Nota-se na bibliografia apresentada, que a automatização desta disciplina não segue um padrão, sendo portanto feito das mais diversas maneiras e automatizando parcialmente os procedimentos desta disciplina.

Existem vários esforços para gerar ferramentas automatizadas para apoiar todas ou quase todas as atividades do GCS de forma integrada.

Há uma expectativa quanto à utilização efetiva desta disciplina, principalmente em ambientes integrados para o desenvolvimento de software, onde os procedimentos de GCS estão embutidos dentro da filosofia do ambiente.

Neste trabalho procurou-se implementar a metodologia GCS para o ambiente SIPS.

A implementação aqui apresentada consistiu do modelamento da metodologia dentro do SIPS, da geração de novas primitivas para o núcleo do SIPS e de uma ferramenta denominada Gerenciador de Configuração de Software. Isto constitui apenas a implementação básica para a introdução desta metodologia no ambiente SIPS.

O grande esforço dispendido neste trabalho foi o de introduzir na base de dados do SIPS os conceitos de alternativa, revisão e variação associados a um ICS para permitir que o SIPS passe a efetuar o controle de versões e de configurações básicas.

A ferramenta Gerenciador de Configuração de Software, neste trabalho, além de executar as funções necessárias para que os conceitos introduzidos na base de dados do SIPS possam ser entendidos, estabelece o início de uma série de melhorias que poderiam ser acrescentadas para prover recursos adicionais de grande potencial ao ambiente SIPS.

Este trabalho apresenta um enfoque diferente dos encontrados na bibliografia pesquisada; o que se tem observado quanto à automatização de GCS compreende ferramentas isoladas que auxiliam principalmente no controle de versões de programas. Um GCS automatizado que pretenda cobrir todas as fases do ciclo de vida não aparece na bibliografia consultada, principalmente um que seja genérico o suficiente para atuar dentro de um ambiente de produção de software abrangente no que se refere a metodologias e técnicas de desenvolvimento.

Como vantagens do GCS automatizado do SIPS vale destacar:

- A documentação é atualizada simultaneamente com a criação dos programas, não a posteriori.

- O processo de modificação é independente de geração manual de documentos;

- rastreabilidade ao longo da evolução do sistema através da análise das evoluções das configurações básicas;

- controle de versões das partes do sistema;

- integridade dos dados do sistema a nível de GCS, dado que ICSs congelados não podem mais ser atualizados;

- geração automática de configurações básicas.

6.2 - Desenvolvimento de Novos Recursos

Nota-se que este trabalho apenas originou uma série de melhorias que podem ser adicionadas à implementação feita, de forma a beneficiar tanto a metodologia GCS no SIPS como o próprio ambiente SIPS.

A implementação deste trabalho não se preocupa com o armazenamento externo do histórico dos ICSs. Desta forma, todas as versões geradas para um ICS permanecem na Base Sistema. Por não possuir uma área de trabalho onde as alternativas possam ser desenvolvidas, estas também permanecem na Base Sistema. Desta forma todas as instâncias geradas de um ICS convivem na Base Sistema podendo-se tanto obter o registro de estado atual de um ICS como o seu histórico.

Esta implementação trouxe à tona novamente discussões quanto à forma de implementação feita da base de dados do SIPS. Isto porque hoje não é permitido a eliminação de objetos inseridos na Base Sistema. Com a implementação de alternativas como áreas de trabalho para o projetista, a filosofia da implementação da Base Sistema deve ser mudada para eliminar fisicamente as alternativas descartadas.

Este problema pode ser resolvido quando da implementação já prevista de se transportar o SIPS para um ambiente mainframe com estações de trabalho para os projetistas. A ferramenta Gerenciador de Configuração de Software ficaria no mainframe, os

projetistas desenvolveriam as alternativas em suas estações de trabalho, em suas áreas de trabalho, e somente no momento de se congelar estas alternativas é que estas seriam passadas para a base de dados do mainframe. Desta forma, somente os dados de versões de ICSs ficariam na Base Sistema.

Um outro fato relacionado à base de dados do SIPS que poderia ser melhorado, refere-se a deixar na Base Sistema somente a última revisão de cada variação de um ICS. O histórico (versões anteriores) poderia ser passado periodicamente para um arquivo à parte. O histórico para o GCS é importante, porque PMSs não resolvidas podem ser consultadas e resolvidas, modificações efetuadas podem vir a ser pesquisadas, configurações básicas anteriores podem ser requisitadas para consulta, etc.

O esquema de proteção à base de dados do SIPS, já previsto, muito contribuirá para um melhor desempenho da metodologia GCS implementada.

Um aspecto do esquema de proteção, de grande importância para o GCS implementado, refere-se à proteção contra escrita sobre determinados componentes (objetos, propriedades e relacionamentos) de uma metodologia; a escrita seria permitida somente a alguns usuários com permissão para tal acesso. Isto permite que a função de controle de configuração possa ser obtida seguramente, uma vez que todos os dados envolvidos com a modificação de um ICS estão centralizados em uma PMS. Desta forma, diferentes níveis de usuários podem acessar um mesmo documento, mas conseguem atualizar somente propriedades deste objeto que o seu nível de proteção permite. Ao projetista cabe atualizar informações técnicas envolvidas com a PMS e, ao gerente, informações gerenciais.

O processo de auditagem de um software poderia ser feito através de várias ferramentas que auxiliariam na verificação, validação, análise de consistência, completeza, clareza e testabilidade do software.

A função de auditagem para o SIPS poderia ser traduzida pela avaliação dos resultados obtidos de ferramentas automatizadas genéricas e específicas para cada metodologia e por técnicas utilizadas para a execução da análise que se queira empregar para auditar um produto de software.

Visto que este processo de auditagem pode ser muito amplo, podem ser geradas ferramentas voltadas a analisar aspectos tão diversos de uma metodologia quanto se julgue necessário. Para o GCS, basta ter-se o conhecimento de quais ferramentas integram o conjunto responsável pelo processo de auditagem; se o resultado

obtido de cada uma destas ferramentas for positivo, um ICS poderá ser congelado e uma configuração básica se estabelecer.

Vale ressaltar que o GCS não se preocupa com os critérios de avaliação de um ICS mas com os resultados obtidos desta avaliação. Desta forma, estas ferramentas, embora não façam parte do GCS auxiliam a metodologia GCS implementada. Para um ambiente como o SIPS, isto seria de grande importância, uma vez que o objetivo de um ambiente de produção de software é o de fornecer o maior número possível de ferramentas automatizadas que auxiliam a tarefa de desenvolvimento de um projeto de software.

Diante disso, supondo a existência de várias ferramentas desenvolvidas para este propósito, para cada variação de ICS gerada no SIPS, deveriam existir associados alguns tipos de atributos relacionados com os critérios de avaliação do processo de auditagem. Somente quando todos os critérios estivessem satisfeitos, o processo de auditagem teria se concretizado para um dado ICS.

O processo de auditagem, como descrito acima, não foi tratado neste trabalho.

A implementação da abordagem descrita acima consistiria em alterar a ferramenta Gerenciador de Configuração de Software para que o comando de auditagem apresentasse subcomandos que corresponderiam aos vários critérios de auditagem dos ICSs da metodologia particular. Assim, a ferramenta Gerenciador de Configuração de Software só geraria uma versão de um ICS que tivesse sido examinado sob todos os critérios e cujos resultados tivessem sido positivos. Somente neste momento o estado do ICS passaria de efetivo (2) para congelado (3).

Se alguns desses critérios não forem satisfeitos, o ICS não passou pelo processo de auditagem. Nesse caso, O estado do ICS passaria de efetivo (2) para em edição (1).

Estes critérios poderiam ser estabelecidos durante o modelamento da metodologia qunado poderiam ser definidos os métodos responsáveis pela auditagem dos produtos associados à metodologia. Desta forma, os critérios também fariam parte da Base Meta.

O trabalho conjunto e sincronizado destas ferramentas com a ferramenta Gerenciador de Configuração de Software, aliadas às características já existentes no SIPS tais como a representação interna uniforme da base de dados, o esquema de proteção e o supervisor, garantem que os procedimentos do GCS possam ser

rigorosamente seguidos e, principalmente, que possam ser rastreados.

Uma outra atividade que não é de responsabilidade do GCS, mas cuja automatização traria grandes benefícios ao ambiente SIPS, é a análise de impacto causada pela modificação proposta a um ICS. Isto a priori está sendo considerado como uma atividade manual cujo resultado é a relação de todos os ICSs afetados pela modificação proposta.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] A. Alderson, M. F. Bott and M. E. Falla, "The Eclipse Object Management System," Software Engineering Journal, Vol. 1, No. 1, Jan. 1986, pp. 39-42.

[2] W. A. Babich, Software Configuration Management - Coordination for Team Productivity, Addison-Wesley, 1986.

[3] N. Belkhatir and J. Estublier, "Experience with a Data Base of Programs," Proc. of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, Dec. 1986, pp. 84-91.

[4] E. H. Bersoff, "Elements of Software Configuration Management", IEEE Trans. on Software Engineering, Vol. SE 10, No. 1, Jan. 1984, pp.79-87.

[5] E. H. Bersoff, V. D. Henderson and S. G. Siegel, Software Configuration Management - An Investment in Product Integrity, Prentice-Hall, 1980.

[6] W. L. Bryan, C. Chadbourne and S. G. Siegel, Tutorial: Software Configuration Management, Eds., Los Alamitos, CA IEEE Comp. Soc. CAT EHO-169-3, 1981.

[7] W. L. Bryan, S. G. Siegel, and G. L. Whiteleather, "Auditing Throughout the Software Life Cycle: A Primer", Computer, Mar. 1982, pp. 57-67.

[8] P. P. Chen, "The Entity-Relationship Model: Toward a Unified View of Data", ACM Trans. on Database Systems, Mar. 1976, pp. 9-36.

[9] J. Estublier, "A Configuration Manager: The Adele Data Base of Programs," Proc. of Workshop on Software Engineering Environments for Programming-in-the-large, Jun. 1985, pp. 140-147.

[10] J. Estublier, S. Ghoul and S. Krakowiak, "Preliminary Experience with a Configuration Control System for Modular Programs," Proc. of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, Apr. 1984, pp. 149-156.

[11] S.I. Feldman, "MAKE - A Program for maintaining computer programs," Software - Practice and Experience, vol. 9, pp.255-265, 1979.

[24] D. P. Schwartz, "4Th IEEE SCM WG Meeting", ACM Sigsoft Software Engineering Notes, Vol. 10, No. 1, Jan. 1985, pp. 69-73.

[25] M. E. Singleton, Automating Code and Documentation Management, Prentice-Hall, 1987.

[26] "The STARTS Guide: Software Tools for Application to Real-Time Systems", UK Department of Trade and Industry/National Economic Development Office Guide, National Computing Centre, Manchester, England, 1984.

[27] I. Thomas and J. Loerscher, "Mosaix - A Version Control And History Management System," Proc. of Workshop on Software Engineering Environments for Programming-in-the-large, Jun. 1985, pp. 128-139.

[28] W. F. Tichy, "Design, Implementation, and Evaluation of a Revision Control System", Proc. of 6Th International Conference on Software Engineering , Sept. 1982, Tokyo, Japan, pp. 58-67.

[29] W. F. Tichy, "RCS - A System for Version Control", Software - Practice and Experience, Vol. 15(7), July 1985, pp. 637-654.

[30] C. Traina Jr., L. F. Capretz, M. B. Carvalho, M. Jino, e M. A. M. Capretz, "Gerenciamento de Configuração de Software usando o Modelo Entidade-Relacionamento", Anais do Segundo Congresso Nacional de Automação Industrial, São Paulo, Brasil, Nov. 1985, pp. 387-395.

[31] C. Traina Jr. F. N. Akkras, L. F. Capretz, M. B. Carvalho, M. Jino e M. A. M. Capretz, "SIPS - Estado Atual de Desenvolvimento", Anais do XIX Congresso Nacional de Informática, Rio de Janeiro, Ago. 1986, pp.297-306.

[32] C. Traina Jr., Máquina e Modelo de Dados Dedicados para Aplicações de Engenharia, Tese apresentada ao IFQSC-USP para obtenção do Título de Doutor, São Carlos, Dez. 1986.

[33] C. Traina Jr. e J. F. W. Staets, "Um Modelo de Representação de Objetos", Anais do III Simpósio Brasileiro de Banco de Dados, Recife, Mar. 1988.

[34] A. N. Tsukumo, C. Traina Jr., C. B. Sampaio, L. F. Capretz, M. B. Carvalho, M. Jino e M. A. M. Capretz, "Um Sistema Expansível para Produção de Software", Anais do II Congresso Nacional de Automação Industrial, São Paulo, Nov. 1985, pp. 379-386.

[35] S. Zucker, "Automating the Configuration Management Process", SOFTFAIR, Arlington, VA, June 1983, pp. 164-172.

[12] C. Gane and T. Sarson, Structured System Analysis: Tools and Techniques, Prentice-Hall, 1979.

[13] E. L. Gomes, Gerência de Configuração de Software, Tese apresentada ao Departamento de Informática da PUC/RJ, para obtenção do Título de Mestre, Rio de Janeiro, Ago. 1981.

[14] G.E. Kaiser and A. N. Habermann, "An Environment For System Version Control," Compcon'83 IEEE Computer Society, San Francisco, Fev. 1983, pp. 415-420.

[15] R. H. Katz and T. J. Lehman, "Database Support for Versions and Alternatives of Large Design Files, IEEE Trans. on Software Engineering, vol. SE-10, n. 2, Mar. 1984.

[16] A. van Lamsweerde, M. Buyse, B. Delcourt, E. Delor, M. Ervier, M. C. Schayes, J. P. Bouquelle, R. Champagne, P. Nisole and J. Seldeslachts, " The Kernel of a Generic Software Development Environment," Proc. of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, Dez. 1986, pp. 208-217.

[17] D. B. Leblang and G. D. McLean Jr, "Configuration Management For Large-Scale Software Development Efforts," Proc. of Workshop on Software Engineering Environments for Programming-in-the-Large, Jun. 1985, pp. 122-127.

[18] P. C. Lockemann, "Analisys of Version and Configuration Control in a Software Engineering Environment", Entity-Relationship Approach to Software Engineering, C. G. Davis, S. Jajodia, P. A. Ng and R. T. Yeh (eds.), Elsevier Science Publishers B. V. (North-Holland), 1983.

[19] D. Mackay, G. Ball, M. Crowe, M. Hughes, D. Jenkins and C. Nicol, "A Unix - Based System for Software Configuration Management", The Computer Journal, vol. 29, No. 6, 1986, pp. 527-530.

[20] R. McCarthy, "Applying the Techinique of Configuration Management to Software", Tutorial: Software Configuration Management, Out. 1980, pp. 42-47.

[21] R. R. Razouk, M. Vernon, and G. Estrin, "Evaluation Methods in SARA - The Graph Model Simulator", Proc. of the 1979 Conference on Simulation, Measurement and Modeling of Computer Systems, 1979.

[22] M. J. Rochkind, "The Source Code Control System", Tutorial: Automated Tools for Software Engineering, Edward Miller, 1979.

[23] P. Rook, "Controlling Software Projects", Software Engineering Journal, Jan. 1986, pp.7-16.