

# METODOLOGIA DE CONSTRUÇÃO DA DOCUMENTAÇÃO DE UM SISTEMA JÁ EXISTENTE VIA UML<sup>1</sup>

*Pablo Parreiras Drumond Ferreira<sup>2</sup>  
Augusto dos Santos Moura Júnior<sup>3</sup>  
João Francisco Neiva de Carvalho<sup>4</sup>*

## Resumo

O presente trabalho tem por objetivo apresentar uma proposição metodológica para que as empresas possam conviver com o paradigma decorrente da necessidade de alteração de sistemas existentes versus alto custo de aprendizado, quando o domínio tecnológico ou da implementação destes sistemas não se encontra mais integralmente nas mãos da organização, criando o mito dos sistemas caixa preta.

O desenvolvimento do trabalho ocorreu em duas etapas. A primeira aponta a necessidade de viabilizar financeiramente uma atividade de documentação parcial no escopo da alteração do sistema. A segunda sugere a adoção da *Unified Modeling Language* como ferramenta de documentação, para que o contínuo processo de alterações conduza à reaquisição do domínio deste sistema por parte da organização. Como exemplo de adoção desta metodologia é apresentado um caso prático referente a um modelo matemático do conversor LD na Vallourec & Mannesmann Tubes do Brasil.

O investimento adicional para a execução de uma alteração em um sistema já existente, relativo à fase de documentação parcial proporcionará, a médio prazo, a recuperação da capacidade da organização em viabilizar prazos, custos e riscos associados a uma requisição de alteração. Concomitantemente a empresa também reduzirá a obsolescência dos sistemas, aumentará o tempo de amortização dos mesmos e minimizará a dependência de que alterações ocorram exclusivamente por alguns indivíduos que historicamente acompanharam o desenvolvimento do sistema.

**Palavras-chave:** UML, manutenção, TI, documentação.

---

<sup>1</sup> X Seminário de Automação de Processos, 04 a 06 de outubro de 2006, Belo Horizonte, Minas Gerais.

<sup>2</sup> Pablo Drumond é formado em Engenharia de Controle e Automação pela Escola de Minas – UFOP.

<sup>3</sup> Augusto dos Santos Moura Júnior é formado em Engenharia Elétrica com Mestrado em Inteligência Artificial aplicada à área financeira e Doutorando em Sistemas de Otimização pela UFMG.

<sup>4</sup> João Francisco Neiva de Carvalho é formado em Engenharia Elétrica pela UFMG e engenheiro de sistemas na Aciaria da Vallourec & Mannesmann Tubes do Brasil.

## 1. Introdução.

O presente trabalho sugere, o conceito de construção incremental da documentação de sistemas para que as empresas, sem domínio da implementação dos mesmos, possam desenvolver seus processos de manutenção. Propõe o detalhamento de acordo com intervenções parciais associadas aos pedidos de alteração, refinamento e estruturação dos modelos e diagramas da *Unified Modeling Language* (UML), permitindo melhoria contínua do entendimento e evitando submeter a mesma a um processo extensivo que não agregará valor 'a manutenção do software.

Adicionalmente, esse trabalho não tem a intenção de descrever todos os diagramas da UML e suas nuances, apenas serão propostos os diagramas que apresentaram um retorno sólido combinado aos conceitos de visão global, intervenções parciais e processo de construção por incrementos.

## 2. Viabilidade Financeira da Manutenção de Software, Através da Construção da Documentação.

Historicamente, a preocupação com o processo de se documentar um sistema de forma precisa e padronizada passou a ser considerada preponderante a partir da década de sessenta. Um marco relevante neste processo foi a documentação por Ivar Jacobson de um sistema de telecomunicação mediante a utilização de camadas de blocos (1). Embora seja perceptível que desde décadas passadas já existam sistemas com uma documentação adequada, a qual permitiria que alterações pudessem ser realizadas com o mínimo dispêndio de tempo e recursos financeiros, ainda hoje pode-se facilmente encontrar softwares essenciais às organizações que não dispõem de uma correta documentação.

Manter e aumentar tende a ocupar mais tempo na vida do software do que o tempo necessário para construí-lo, acrescenta SCOTT(1). A ausência da modelagem torna esses procedimentos extremamente onerosos e o custo do aprendizado da equipe leva, prematuramente, à consideração de técnicas de refatoração ou reconstrução do sistema. A falta do conhecimento global ou centralização da alteração na arquitetura acarreta não identificação de oportunidades de reuso, gerando maiores riscos para as intervenções consecutivas, tornando o processo de reaquisição do domínio da implementação mais distante e implicando na dificuldade de evolução do sistema. Em casos extremos estes fatores podem culminar na redução brusca do ciclo de vida do sistema.

Dessa forma, modelar software é a competência de simplificá-lo em uma linguagem que permita facilitar e reter informação de sua cobertura. Os motivos para viabilizar o modelamento são descritos por JONES(2). A seguir encontram-se destacados os motivos que incidem sobre sistemas já construídos:

- Inabilidade de conduzir as alterações nos requerimentos
- Módulo que não se conversam de forma inteligível
- Softwares difíceis de manter e estender
- Descoberta tardia de falhas graves de projeto
- Falta de confiança no processo de construção e implantação

Especificamente quanto à viabilidade econômica da manutenção de software, as organizações modernas usualmente aplicam a estes sistemas critérios similares aos utilizados para a avaliação de investimentos e manutenção dos seus demais ativos. Desta feita, faz-se necessário que os indicadores Taxa de Retorno, Tempo de Amortização do Investimento, Valor Presente Líquido sejam sempre favoráveis à imobilização do capital necessário para a realização da alteração, ou revisão, proposta para o sistema.

Uma vez que a metodologia aqui sugerida irá permitir a reconstrução da documentação do sistema de forma paulatina e progressiva, torna-se aceitável a hipótese de que este custo de manutenção não constituirá um acréscimo considerável ao escopo da manutenção, dado que, para sistemas caixa preta existentes, sempre existe um custo de estudo do sistema associado à manutenção. Concomitantemente, também se torna admissível a idéia de que a formalização do registro da documentação irá conduzir a uma aceleração do tempo análise, planejamento e realização da manutenção destes sistema. Conseqüentemente, a medida que o processo de documentação se consolida o custo total das manutenções irá inexoravelmente reduzir, tornando o processo de manutenção mais viável para as organizações.

### **3. Metodologia de Construção da Documentação de um Sistema Existente**

#### **3.1 Definição da UML**

A *Unified Modeling Language* (UML) é uma linguagem gráfica para visualizar, especificar, construir e documentar os artefatos envolvidos no processo de desenvolvimento de software (3).

Portanto, a documentação proposta para sistemas existentes, baseada na UML, será composta de elementos (entidades e relacionamentos), diagramas e visões (perspectivas de um modelo).

#### **3.2 Importância da UML**

Sumarizando todos os motivos apontados para a modelagem de software e a viabilização financeira da documentação, a UML torna-se ferramental indispensável para elevar a eficiência da comunicação entre os mantenedores da aplicação, isso significa:

- Proporcionar um padrão para abstração da complexidade ou simplificação da realidade;
- Auxiliar o planejamento de funcionalidades, preservando as features da arquitetura;
- Possibilitar a reutilização de componentes, pois sua carência torna dispendiosa a manutenção.

#### **3.3 Documentos Iniciais**

São três os documentos iniciais propostos por MEDEIROS(4) como alicerces para a construção dos modelos: Visão, Nomenclatura e Glossário

##### **Documento Visão**

O Documento Visão descreve, textualmente, aspectos relacionados ao propósito, tecnologia, linguagem, restrições e banco de dados da aplicação, além dos requisitos de documentação, e, graficamente, um modelo conceitual, abstraído por um diagrama de casos de uso, abordado nas próximas seções.

Diversos autores recomendam uma descrição sucinta e em alto nível nesse documento, alertando para não transformá-lo em uma especificação morosa e detalhada do sistema.

##### **Documento Nomenclatura**

O Documento Nomenclatura é obtido através de pequenas intervenções no código-fonte, verificando e apresentando o padrão utilizado para nomeação de palavras no corpo do mesmo.

É importante lembrar que, historicamente, sistemas que não possuem documentação, também não possuem organização adequada, no que se refere a nomenclatura. A constatação dessa, torna inviável a confecção do documento nomenclatura, sem levar em consideração técnicas de refatoração.

## **Documento Glossário**

O Documento Glossário, esclarece os termos utilizados do negócio no desenvolvimento e modelamento do software, sendo crucial para organizações que possuem alta rotatividade e que desejam minimizar o custo do aprendizado nas alterações.

### **3.4 Semelhanças com o Processo Unificado**

Assim como o Processo Unificado de desenvolvimento, o processo de documentação proposto também tem como premissas a centralização da arquitetura, a direção por casos de uso e a construção incremental da mesma. Essas características são descritas nas seções seguintes.

Então o que deve ser descrito primeiro? Se por um lado os casos de uso dirigem a arquitetura de um sistema, pois dirigem o esforço de desenvolvimento e documentação. Por outro lado, a arquitetura guia a seleção e exploração de casos de uso.

A segunda estratégia deve ser priorizada pelos seguintes motivos:

- O principal artefato já está construído, ou seja, o próprio software, portanto, o esforço de desenvolvimento já foi aplicado ao mesmo;
- A descrição da arquitetura requisita o Documento Visão, proposto como um dos documentos iniciais, e então esse exigirá o diagrama de caso de uso que agrega valor ao documento;
- Torna-se claro que somente a partir da visão superficial podemos aprofundar na documentação, em consequente nos casos de uso;

### **3.5 Definição de Arquitetura**

Arquitetura é a organização fundamental do sistema como um todo. Entre os aspectos de uma arquitetura, estão incluídos elementos estáticos, elementos dinâmicos, o modo como estes elementos trabalham juntos e o estilo arquitetônico total que guia a organização do sistema. A arquitetura também se refere a questões como desempenho, escalabilidade, reuso e restrições econômicas e tecnológicas (1). Deve-se então avaliar os seguintes questionamentos:

#### **Porque a documentação deve ser centrada na arquitetura?**

Entender a Visão Global: A descrição da arquitetura tem como principal objetivo facilitar o entendimento global da aplicação.

Organizar o esforço de documentar: a discretização do sistema em porções permite focalizar o esforço no incremento da documentação, tornando eficiente a comunicação.

Facilitar as possibilidades de reuso: explicitar as interfaces dos componentes, facilitando a identificação de oportunidades de reuso para que as alterações ocorram de modo elegante.

Facilitar a evolução do sistema: todos os benefícios relacionados à centralização da arquitetura, culminam na construção dos pilares que tornará possível e saudável a manutenção do sistema.

#### **Como descrever a arquitetura?**

A descrição da arquitetura de um sistema estará baseada em uma série de documentos e diagramas, sendo eles o documento Visão, Diagrama de Caso de Uso referente à Visão, também chamados arquitetonicamente significativos, Diagrama de Componentes e Implantação, que permitirão a avaliação da mesma quanto a flexibilidade, adaptação de novas propriedades e riscos relacionados aos pedidos de alterações.

### **3.6 Casos de Uso**

#### **3.6.1 Elementos do diagrama de casos de uso**

Dois conceitos devem ser inseridos antes da definição de caso de uso, são eles Ator e Requisito.

### Ator

Ator pode ser entendido como uma pessoa ou um sistema (conhecido como entidade externa na análise estruturada), que interaje com o software, realizando uma atividade e atuando sobre um caso de uso. Sua representação está definida na figura 3.1

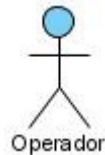


Figura 1 - Ator.

Fonte: Elaborado pelo autor.

### Requisito

Requisito é uma condição ou capacidade que um software deve ter

### Caso de Uso

Um caso de uso pode ser explicado como a representação visual de um requisito funcional, uma macroatividade que encerra diversas tarefas ou atividades. Sua representação está definida na figura 3.2

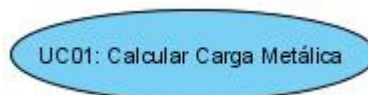


Figura 2 - Caso de Uso.

Fonte: Elaborado pelo autor.

### 3.6.2 Diagrama de Caso de Uso

O diagrama de caso de uso reúne os elementos citados anteriormente, e adiciona associações entre atores e casos de uso, além do relacionamento entre os mesmos, como descreve a figura 3.3

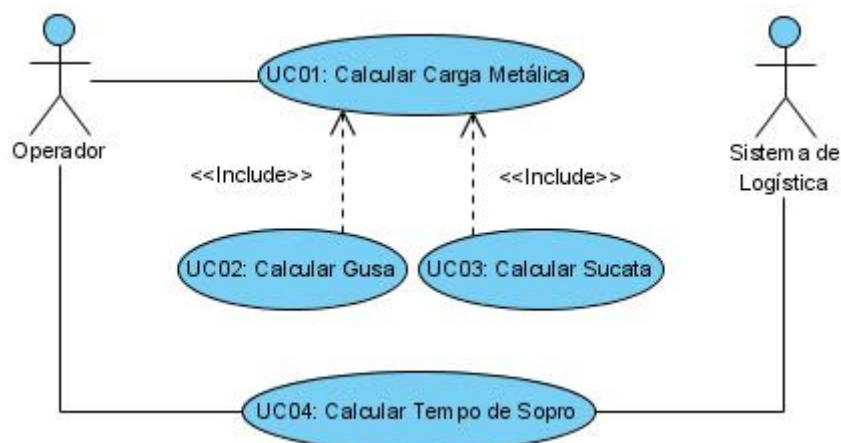


Figura 3 - Diagrama de caso de Uso.

Fonte: Elaborado pelo autor.

Alguns autores defendem a idéia de representação de linhas que delimitam o sistema onde os casos de uso estão inseridos, como também a nomeação desse módulo.

### 3.6.3 Identificação de casos de uso em um sistema existente

Em um sistema existente, no estilo caixa-preta, a identificação de casos de uso é uma tarefa que consiste em relatar ações do usuário associadas a uma lista de resultados obtidos através dessa interação.

### 3.6.4 Porque a documentação deve ser dirigida por casos de uso?

Deve-se utilizar casos de uso como força condutora da documentação, pois são eles os verdadeiros responsáveis pela discretização do sistema em porções de fácil entendimento. Adicionalmente, SCOTT(1) apresenta diversas vantagens dessa direção, dentre as quais podemos destacar:

- São expressos sob as perspectivas do usuário do sistema;
- São escritos em uma linguagem natural e por isso, são intuitivamente óbvios;
- Oferecem uma habilidade de compreensão consideravelmente melhor, quando comparados as descrições textuais morosas e redundantes.

### 3.6.5 Como estruturar o diagrama?

Esta tarefa compreende desmembrar os casos de uso na busca por casos de maior simplicidade. A UML disponibiliza três relacionamentos entre casos de uso: *Extend*, *Realization* e *Include*, esse último exibido na figura 3.3, sob o estereótipo <<include>>

Deve-se ainda salientar que os diagramas de caso de uso presentes no Documento Visão não apresentam tais desmembramentos, pois o objetivo desse documento é diminuir a complexidade adjacente.

### 3.6.6 Como detalhar os casos de uso?

Detalhar um caso de uso significa, descrever textualmente o curso das atividades exercidas pelo usuário para a produção dos resultados. Em sistemas existentes, essa etapa está implicitamente ligada à identificação de casos de uso, anteriormente explanada.

## 3.7 Diagrama de Componentes e Implantação

Segundo MEDEIROS(4), um componente é qualquer arquivo que contenha uma parte necessária à execução de um software. Sendo importante salientar que a crença de que componentes são somente executáveis é errônea.

Sua representação expõe suas interfaces (métodos públicos) para os demais componentes, como representado na figura 3.4, provendo a identificação de oportunidades de reuso e mantendo características vitais como:

- Coesão: as operações e atributos trabalham juntos, de forma compacta, tanto nas classes e, obviamente, nos componentes;
- Acoplamento: dependências mínimas entre as classes e entre os componentes do software.

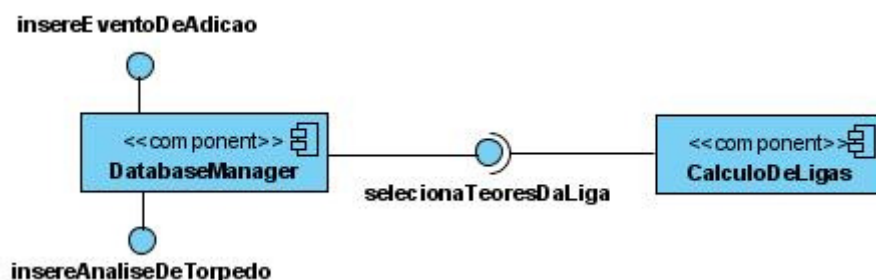


Figura 4 - Diagrama de Componentes.

Fonte: Elaborado pelo autor.

De maneira geral, esse diagrama aplicado à descrição da arquitetura, sugere ao mantenedor, conservar as responsabilidades de cada componente, garantindo que somente estes executarão as tarefas relacionadas. Parece simples, mas se mostra fundamental para alterações sadias na aplicação.

O diagrama de implantação da UML, exibido na figura 3.5, encerra a organização física do sistema como recursos de infra-estrutura computacional, rede ou artefatos. Ele é composto de elementos denominados nós (representação de hardware que fornece estrutura para os componentes) e o relacionamento entre esses.

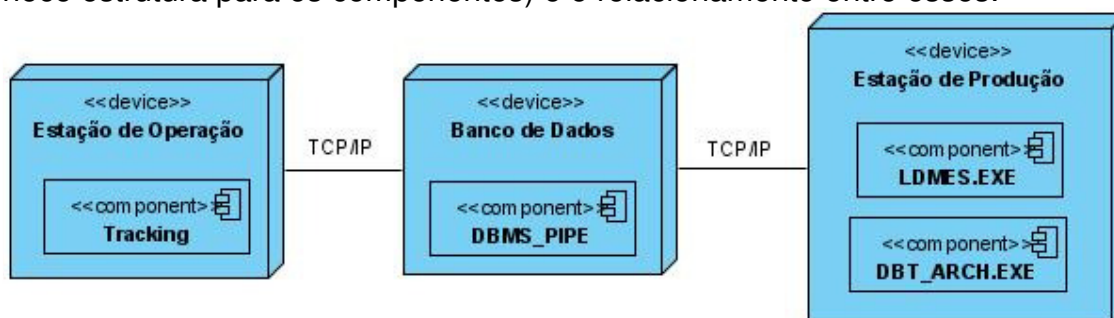


Figura 5 - Diagrama de Implantação.

Fonte: Elaborado pelo autor.

O avanço que o foco em sistemas distribuídos trouxe para a computação é incontestável, como também tornou o software desenvolvido bem mais complexo. É dentro desse enfoque que o diagrama de implantação demonstra valor para descrição da arquitetura existente.

### 3.8 Documentação Incremental

Para a descrição da arquitetura os documentos e diagramas apresentados até o momento são suficientes. Aproveitando o ensejo, segundo Ivar Jacobson, um dos pais da UML, cerca de 20% do que a linguagem de modelamento propõe resolve 80% dos problemas do dia-a-dia.

O conceito de documentação incremental propõe aprofundar passo a passo de acordo com intervenções parciais associadas aos pedidos de alteração, permitindo melhoria contínua do entendimento, e evitando submeter a mesma a um processo extensivo que não agregará valor a manutenção do software.

Os incrementos serão cunhados, nessa fase, a partir da estruturação dos diagramas de caso de uso, descrito anteriormente, e obtenção dos diagramas de atividades e diagramas de classes.

### 3.9 Diagrama de Atividades

O Diagrama de Atividades descreve o fluxo das diversas atividades associadas aos casos de uso, auxiliando a descrição dos mesmos. Adicionalmente, MEDEIROS(4) destaca a colaboração do esquema para ajustes durante a confecção dos diagramas de classes e objetos.

A representação se mostra intuitiva pelo fato de se basear nos fluxogramas tradicionais, sendo interessante destacar sua semelhança com o conhecido diagrama de fluxo de dados (DFD) da análise estruturada.

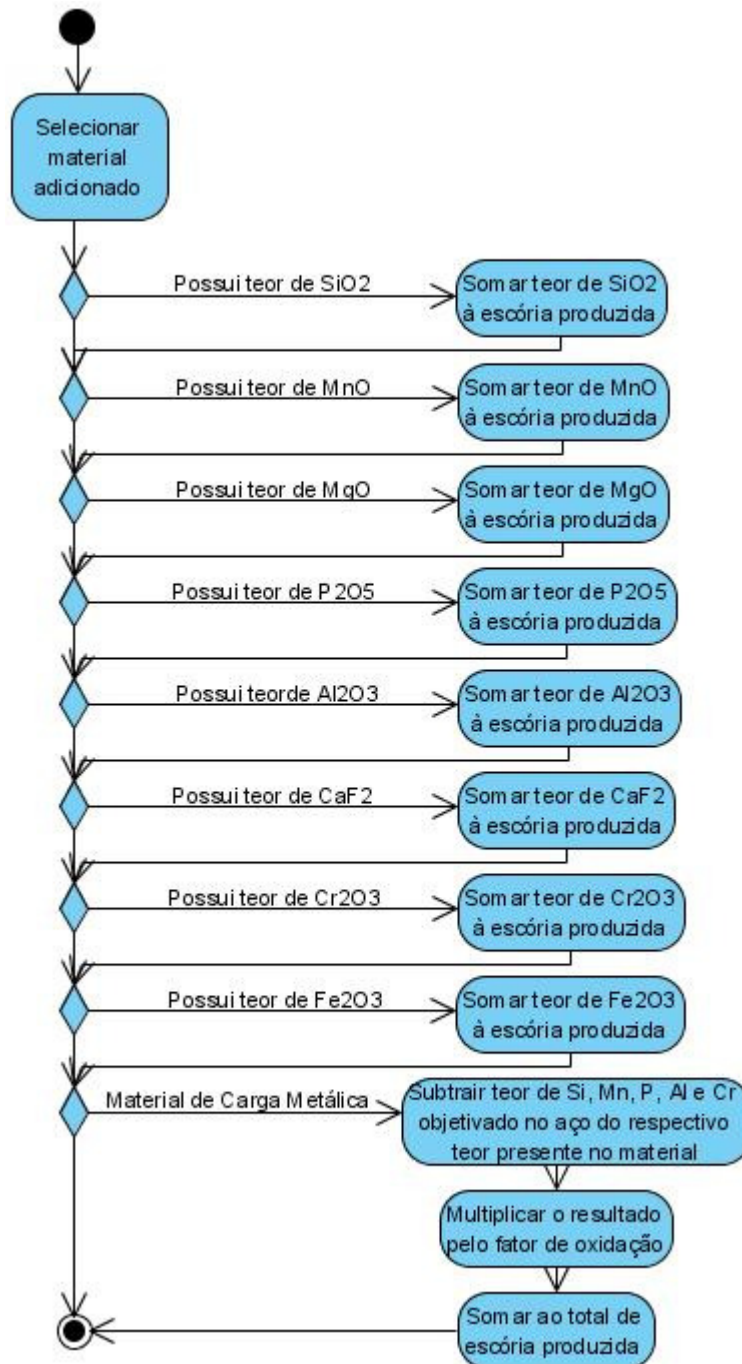


Figura 6 - Diagrama de Atividades.  
Fonte: Elaborado pelo autor.

A figura 3.6 apresenta um diagrama de atividades para cálculo da quantidade de escória gerada para 1000 Kg de adição de material.

### 3.10 Diagrama de Classes

O diagrama de classes é provavelmente o mais conhecido, e é utilizado para representação dessas e relacionamentos, conforme exemplifica a figura 3.7, convergindo na realização dos casos de uso apontados anteriormente, além de apresentar conceitos essenciais do mundo real.

É importante lembrar que não entra no escopo desse trabalho definir toda a teoria de classe envolvida no paradigma de orientação ao objeto, a preocupação aqui é somente expor a notação utilizada pela UML e seu valor para a documentação.

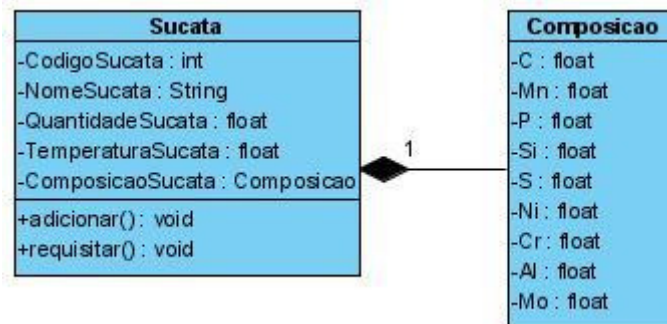


Figura 7 - Diagrama de Classe.  
Fonte: Elaborado pelo autor.

As classes são divididas em três partes claras, correspondendo ao nome, atributos e, por último, as operações. O relacionamento exibido acima, associa as classes por composição, ocorrendo quando os objetos da classe parte (Composicao) não “sobrevivem” quando o todo (Sucata) é destruído.

O modelo de domínio, proposto pelo Processo Unificado, e adotado nesse trabalho, é composto pelos diagramas de classes, reproduzindo o espaço do problema em um nível de abstração razoavelmente alto, como explica SCOTT(1).

Apesar da obtenção desse modelo se dar de forma trivial a partir da análise do sistema existente, ele representa a base sólida para o entendimento, reuso e refinamento.

#### 4. Aplicação da Metodologia e Conclusão

A aplicação dessa metodologia ao modelo matemático do conversor LD da Vallourec & Mannesmann Tubes do Brasil, mostrou que os diagramas, e modelos apresentados até o instante foram satisfatórios para a reanálise do controle da aplicação. Os incrementos da documentação foram vinculados aos pedidos de alterações e documentados através dos diagramas de seqüência e estado, seguindo o conselho de AMBLER(5), de só criar um modelo quando ele fornece algum valor positivo para os esforços envolvidos. Como resultado final, foi comprovada a eficiência da metodologia na reconstrução da documentação do sistema, a qual era praticamente inexistente, bem como na obtenção de maior previsibilidade do processo de planejamento de alterações, redução no tempo e custo das manutenções do sistema e diminuição do risco de efeitos colaterais associados ‘a intervenções no sistema.

Pode-se assim concluir que, se por um lado máquinas, equipamentos e instalações apresentam rendimento decrescente mediante ao período de utilização e exigem maiores despesas de manutenção (6), por outro, o software com documentação adequada terá seu tempo de amortização elevado e, simultaneamente, poderá ser mantido com menores custos e riscos.

Deve-se ainda ser destacada a importância de manter a documentação atualizada continuamente, de maneira análogo ao que já é praticado com os demais documentos da empresa, tornando o processo de documentar sistemático e iterativo, conforme solicitação de alterações. Por fim, deve-se ter em mente que a

formalização da documentação deve passar a integrar, de forma obrigatória, o processo de desenvolvimento e manutenção de software.

## 5. Referências Bibliográficas

- 1 SCOTT, Kendal. O processo unificado explicado. Porto Alegre: Bookman, 2003.
- 2 JONES, Caper. Patterns of software systems failure and success. London: International Thompson Computer Press, 1996.
- 3 The Object Management Group. "OMG Unified Modeling Language Specification", versão 2.0, jul. 2004. Disponível em: <http://www.omg.org/technology/documents/formal/uml.htm>. Acesso em: 18 abr. 2006.
- 4 MEDEIROS, Ernani Sales de. Desenvolvendo software com UML 2.0: definitivo. São Paulo: Pearson Makron Books, 2004
- 5 AMBLER, Scott W. Principle of maximize stakeholder investment agile modeling Disponível em: <http://www.agilemodeling.com/style/stateChartDiagram.htm>. Acesso em 20 abr. 2006.

### Abstract

The following work aims to present a methodological proposal to let companies live with the paradox which comes from the necessity of altering existing systems versus the high cost of learning. This paradox becomes even more evident when technology domain of system implementation does not lie utterly in the hands of the organization, thus creating the 'Black Box' myth.

The project development was divided into two phases. The first one points out the need to financially justify the act of partially documenting the system alteration. The second phase suggests the adoption of the Unified Modeling language as a documenting tool so that the continuous process of alteration leads to the reacquisition of the system domain by the organization. As an example, it is discussed the adoption of this methodology associated with a practical mathematical problem, at the LD converter of Vallourec & Mannesmann Tubes do Brasil.

The investment pretends to pay off in the medium term by improving the organizations capacity (time, cost and associated risks) to deal with future systems modifications. Meanwhile the company will also decrease the amount of obsolete systems, increase the amortization time of these systems and minimize the systems dependence from first implementation specialists.

**Key-Words:** UML, maintenance, IT, documentation.