# Processo de Redocumentação: Uma Necessidade

Nicolas Anquetil, Káthia Marçal de Oliveira
Universidade Católica de Brasília
SGAN 916, módulo B, Asa Norte CEP 70790-160
Brasília, DF – Brazil
Phone/Fax: (61)340-5550
{anquetil, kathia}@ucb.br

**Resumo**

A maioria dos engenheiros de software conhece e entende a importância da documentação de um software. Infelizmente, pressões de tempo e custos muitas vezes levam esses engenheiros à não realizar a atividade fundamental de documentação. Somente, quando realizando manutenções posteriores a necessidade de melhor documentação surge como fator essencial. Para manter ativos sistemas antigos sem documentação ou mal documentados, as empresas precisam cada vez mais recursos na manutenção de software. É necessário, portanto, redocumentar esses software para possibilitar uma melhor manutenção atual e futura. Este artigo apresenta um processo de redocumentação de software projetado levando em consideração dois aspectos: a escassez de recursos para realizar essa atividade e o uso de uma abordagem baseada em engenharia reversa, ou seja, a partir do código fonte para níveis mais abstratos de documentação. Este artigo apresenta ainda lições aprendidas obtidas do acompanhamento de uso desse processo em empresas.

**Palavras Chave:** Processo de Software, Qualidade de Processo de Software

**Abstract**

Most software engineers know and understand the importance of software documentation. Unfortunately, in many cases, time and budget constraints prevent them ro perform the fundamental activity of documentation. A few maintenance operations later, the need for better documentation arises as an essential factor. To keep active old software systems with inexisting or bad documentation, companies need to spend more resources in software maintenance. It is essential, therefore, to re-document those software systems to be possible a better actual and future maintenance. Tthis paper presents a software process redocumentation designed taking into accoount two aspects: the scarcity of resources to perform such a task and the use of reverse engineering approach, that is, from source code to more abstract documentation. This paper also presents lessons learned from the use of software process in some companies.

**Keywords:** Software Process, Quality of Software Process

## 1. Introdução

A Documentação de sistema é um requisito mínimo de uma boa gestão de desenvolvimento de sistemas. Por isso, processos de desenvolvimento de software geralmente apresentam um conjunto de artefatos que correspondem a vários documentos para cada fase de desenvolvimento. A própria norma de processos de ciclo de vida de software ISO/IEC 12207 [1] define a documentação como um processo de apoio que deve ser utilizado em

vários outros processos de software. No entanto, os padrões nem sempre foram estes. Durante várias décadas o desenvolvimento de software se deu de forma desorganizada produzindo uma documentação não satisfatória, ou na maioria das vezes, não produzindo nenhuma documentação. Manter tais sistemas sempre é custoso e problemático. As pessoas que desenvolveram esses sistemas, geralmente não estão mais nas empresas ou realizam outras atividades. É de fundamental importância, portanto, gerar uma documentação suficiente de forma a apoiar manutenções atuais e futuras.

Constatamos essa realidade ao participar de trabalhos de apoio ao desenvolvimento e manutenção em quatro empresas brasileiras. Em cada caso, encontramos sistemas de extrema importância sendo considerados essenciais para a empresa, que passaram por manutenções de melhorias contínuas, por diferentes equipes, resultando em uma documentação que, quando existe, é fraca e desatualizada. Restrições de tempo e custos muitas vezes levam a um desenvolvimento caótico e sem qualidade o que resulta em sistemas mal documentados.

A partir do trabalho nessas empresas, e posteriormente, em discussão com outros profissionais que atuam em diferentes nichos do mercado (governamental e privado) percebemos, a necessidade de definir como redocumentar tais sistemas. A norma de processo de software (ISO/IEC 12207 [1]) e modelos atuais (como SPICE [2], CMM [3]) tratam geralmente de como documentar um sistema ao longo do desenvolvimento ou apenas indicando que as documentações existentes devem ser atualizadas durante o processo de manutenção. No entanto, elas não abordam a questão de como redocumentar tais sistemas para garantir uma documentação suficiente que seja útil para futuras manutenções.

Esse artigo apresenta, nossa experiência na definição e uso de um processo de redocumentação. Na seção seguinte (seção 2), apresentamos uma breve discussão sobre manutenção de sistemas e redocumentação. A partir dessa discussão inicial, apresentamos nossa abordagem de processo de redocumentação (seção 3) e, em seguida, um conjunto de lições aprendidas desse processo obtidas através do seu uso na redocumentação de projetos reais. Na seção 5 apresentamos as conclusões desse trabalho.

## 2. Manutenção e Redocumentação

Todo sistema em uso está sujeito a manutenções. Segundo a primeira lei de evolução de software de Lehman [4] um software em uso ou evolui continuamente ou se torna progressivamente menos útil. A manutenção é reconhecida como um das atividades de engenharia de software mais realizadas, correspondendo a mais de 60% de todo o esforço de uma organização que produz software [5]. Segundo Pfleeger [6], estimativas atuais sugerem que o custo de manutenção pode ter aumentado em até 80% do custo total da vida de um sistema nos anos 2000. De forma geral, manutenção consiste de evoluções, adaptações e/ou correções realizadas num sistema em operação para atender a problemas no software, detectados durante o uso, ou às novas necessidades dos usuários. Fazer manutenção em um sistema que não possui nenhuma documentação não é uma atividade trivial. A equipe pode gastar muito tempo para localizar e corrigir o erro e muitos efeitos colaterais podem surgir por não se saber que programas e dados podem ser afetados com essa alteração.

Diante dessa dificuldade diferentes pesquisas têm sido realizadas na área de engenharia reversa (ver, por exemplo, [7,8]) no sentido de apoiar o entendimento desses sistemas na manutenção. A engenharia reversa busca analisar códigos de programa para criar representações em um nível de abstração recuperando modelos de projeto [5]. Essa análise é

geralmente feita através de ferramentas que interpretam as estruturas submetidas e geram documentos de saída especificados [6]. A redocumentação pode ser vista como uma parte da engenharia reversa, mas de forma mais simplificada. A redocumentação envolve a análise estática do código fonte para produção de uma documentação, sendo examinado o uso de variáveis, chamadas de componentes, caminhos de controle, tamanho do componente, parâmetros de chamada, caminhos de teste e qualquer outra alternativa que ajude no entendimento do que o código realiza e como o realiza [6]. No entanto, ao contrário da engenharia reversa, a redocumentação não busca a recuperação de modelos de projeto de forma automática ou semi-automática.

Segundo Muller *et al.* [9] apesar dos avanços nessas áreas, processos de engenharia reversa precisam ser identificados e melhor apoiados como um passo necessário para torná-la mais aplicável. De forma semelhante, processos de redocumentação precisam ser estabelecidos de forma a identificar por onde começar uma redocumentação, que documentos devem ser gerados que possam ser úteis no desenvolvimento, o que deve ser realizado, como e por quem.

Um processo de software envolve um conjunto e atividades, restrições e recursos que produzem algum tipo de saída desejada [6], ou de forma mais simples, um processo é um conjunto de atividades inter-relacionadas, que transforma entradas em saídas [1]. Todos os esforços na definição e avaliação de processos (ISO, CMM, PSP, ISO9000-3, SPICE entre outros) insistem na importância de se documentar todas as atividades realizadas e manter esta documentação atualizada. A norma ISO/IEC 12207, por exemplo, inclui um processo específico para o propósito de documentação de todas as atividades de outros processos. Um dos principais objetivos da documentação é auxiliar nas manutenções futuras de um sistema. A ISO/IEC 12207 também define um processo específico para manutenção onde é indicado a necessidade de se atualizar a documentação com as modificações realizadas. No entanto, a ISO/IEC 12207 não inclui ou especifica nenhum processo para redocumentação e o processo de documentação definido é muito genérico para ser realmente útil nesse caso.

De forma semelhante, os modelos CMM (*Capability Maturity Model)* [3] e SPICE (*Software Process Improvemente Capability Determination)* [2] enfocam processos de desenvolvimento e manutenção mas não dão muita atenção a necessidade de redocumentação. O amplamente conhecido processo unificado (*Rational Unified Process*) [10] considera a atividade de manutenção como um novo ciclo de desenvolvimento, o que implicitamente supõe uma documentação existente "apenas" ser atualizada. Finalmente, o CM3 (*Corrective Maintenance Matutity Model)* [11], um modelo de definição de processos para manutenção corretiva, além de bastante particular para esse tipo de manutenção, também não considera a redocumentação.

Dessa forma, partindo do pressuposto que a redocumentação é uma necessidade e que os processos atuais ou consideram a documentação como uma parte do desenvolvimento ou como manutenção de documentações existentes, partimos para a definição de um processo de redocumentação que garanta gerar uma documentação suficiente para apoiar a atividade de manutenção.

## 3. Definição de um Processo de Redocumentação

Nesta seção, apresentamos o processo de redocumentação considerando suas atividades, sub-atividades, artefatos gerados, e responsáveis pela sua realização. Classificamos esse

Processo de Redocumentação como um processo de apoio a ser utilizado dentro dos processos de manutenção e operação definidos na ISO12207 (Figura 1) pelo fato de considerarmos essencial para a realização de manutenções.

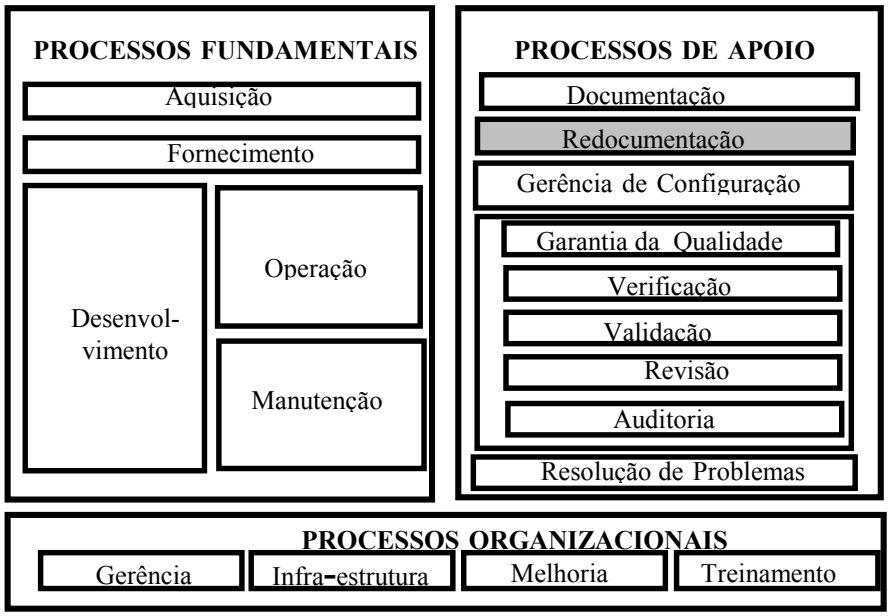


**Figura 1 – Processo de Redocumentação na ISO/IEC 12207**

Para a definição do processo de redocumentação, consideramos três características básicas:

i) Ser baseado na engenharia reversa – ou seja, o processo de redocumentação deve ser baseado em uma abordagem "bottom-up" (a partir do código), não devendo ser realizadas novamente todas as fases de desenvolvimento (levantamento de requisitos, análise, projeto, etc.)

ii) Gerar a documentação mínima necessária – para diminuir o custo de redocumentação e maximizar as chances de a documentação gerada ser sempre mantida atualizada depois, nós seguimos a recomendação definida em Pressman [5] de limitar a redocumentação ao mínimo necessário;

iii) Buscar automatização quando possível – ou seja, tentar definir artefatos no processo que possam ser gerados automaticamente ou semi-automaticamente desde que mantenham uma informação de valor para manutenção.

O Processo de Redocumentação está dividido em três grandes fases (figura 2) e estas divididas em atividades:

- **Fase de Preparação**: que consiste em levantar informações sobre o sistema a ser redocumentado. A partir dos dados coletados nessa fase, será feito o planejamento do que será redocumentado. Essa fase é composta de duas atividades: Inventário do sistema e Auditoria do Sistema.
- **Fase de Planejamento**: esta fase consiste de planejar o que será redocumentado, quem deverá realizar a documentação e a definição de um cronograma para as atividades a serem realizadas. Essa fase constitui de apenas uma atividade, o Planejamento da Redocumentação.

- **Fase de Redocumentação**: nesta fase a redocumentação propriamente dita deve ser realizada sendo composta de quatro atividades: Definição de Visão de Alto Nível, Gerar Referências Cruzadas, Definir Subsistemas, Gerar Documentação de Baixo Nível.

Cada uma das atividades gera uma documentação específica na qual podem ser usadas não apenas descrições textuais como diagramas propostos por diferentes abordagens, recomendando-se, no entanto, utilizar aquela de maior domínio na organização.

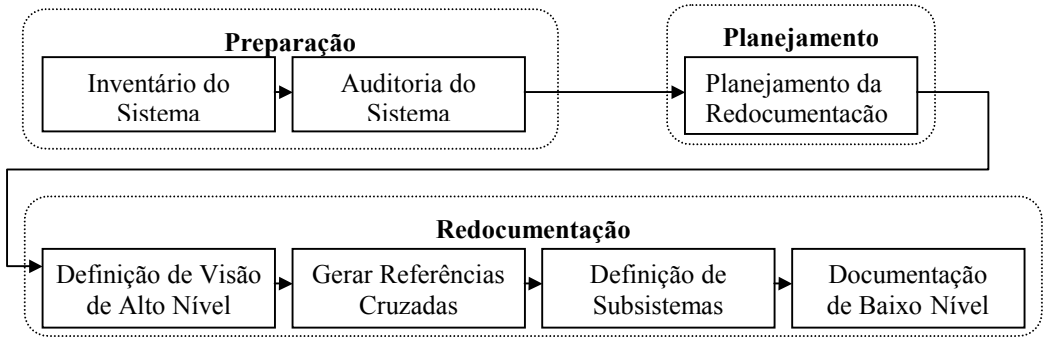


**Figura 2 – Processo de Redocumentação**

## 3.1 Preparação

A fase de Preparação consiste em entender o sistema sendo analisado para permitir realizar o planejamento da redocumentação existente. Essa fase está dividida em duas atividades: Inventário do Sistema e Auditoria do Sistema

O **Inventário do Sistema** tem como objetivo estabelecer qual é a situação da organização em relação ao sistema definindo: O que se sabe sobre o sistema? Onde encontrar essas informações? Onde procurar por novas informações? Essa atividade deve produzir um Relatório de Inventário do Sistema.

Para realizar o inventário deve-se, inicialmente, identificar informações quantitativas sobre o sistema, ou seja, levantar um inventário dos componentes do sistema, identificando a quantidade de: subsistemas ou módulos dentro do mesmo sistema (se existirem)[1]; funções, ou funcionalidades, do sistema; classes, se possível distinguindo entre classes do negócio (geradas para atender a própria aplicação) e classes de apoio (utilizadas por várias aplicações e que não fazem parte do negócio); métodos por classes (públicos e privados); tabelas (do negócio e auxiliares); arquivos[2] (do negócio e auxiliares). A partir daí, deve-se para cada subsistema, ou arquivo de negócio no caso de não haver subsistemas identificar as classes que os compõe e para cada classe identificar seus métodos.

Uma vez levantados os dados do sistema, devem ser identificadas e documentadas as possíveis fontes de informação no que se refere à documentação e pessoas que participaram do desenvolvimento e uso do sistema. No que se refere à documentação, verificar se os seguintes itens existem, seus nomes e onde encontrá-los: documentação do sistema

[1] Em algumas organizações, o sistema pode ser apenas composto de um conjunto de arquivos não havendo uma definição clara dos seus subsistemas.
[2] Arquivo de negócio – arquivo com código que atendam funções do sistema
Arquivo auxiliar – arquivo com funções auxiliares de acesso a dados, formatação, etc.

(diagramas, modelos, descrição dos requisitos ou das funcionalidades implementadas, documentação de um banco de dados, etc.), documentação do usuário (Descrição do funcionamento do sistema e explicação das funcionalidades disponíveis); e, documentação de manutenção do sistema (relatórios de bugs, relatórios de modificação do sistema)

No que se refere às pessoas, selecionar (nome e contato) um grupo de usuários com experiência de uso do sistema (nesse caso indicar quantos anos de experiência) que possam estar disponíveis para fornecer alguma informação, caso necessário, durante a redocumentação do sistema; identificar (nome e contato) usuários que participaram da definição dos requisitos e de funcionários que participaram do desenvolvimento do sistema e com experiência de manutenção do sistema.

A **Auditoria do Sistema** tem como objetivo estabelecer qual é a situação do sistema e das fontes de documentação identificadas. Ou seja, devem ser avaliados o nível de desorganização do sistema e o grau de confiança que pode se ter em cada fonte de informação do sistema. Essa é uma atividade longa e custosa e deve ser feita por uma equipe que não esteja trabalhando no dia a dia da manutenção pelo fato dessa atividade acabar sendo deixada para depois tornando o processo ainda mais longo.

A auditoria deve ser feita no que se refere a qualidade interna (percebida por desenvolvedores e analistas) e qualidade externa (percebida pelos usuários finais).

No que se refere a qualidade interna, a auditoria pode ser feita (i) a partir de dados já existentes ou (ii) de forma mais investigativa. O primeiro caso é possível, somente, quando a organização tem um controle sobre o número de modificações solicitado decorrentes de erros no sistema e um registro sobre o impacto dessas modificações. Para isso procura-se identificar algumas medidas sobre essas manutenções, tais como: número de erros conhecidos; número médio de novos erros por ano/trimestre/mês; número médio de novos erros introduzidos pela correção de um erro; número de erros conhecidos ainda não corrigidos; etc. No segundo caso, procuramos fazer uma investigação detalhada sobre as fontes de informações identificadas (geralmente banco de dados, códigos e documentos) com dois objetivos básicos: achar contradições entre as diferentes fontes de informação; e verificar padrões e qualidades gerais (por exemplo: será que o sistema respeita as exigências de uma boa modularização?).

No que se refere a qualidade externa, sugerimos que o sistema seja avaliado considerando as características de qualidade em uso da ISO9126-1 [12]: *atratividade*, que refere-se ao quanto o sistema é atrativo ao uso pelo usuário; *efetividade*, que se refere a facilidade que o usuário tem de alcançar o objetivo desejado quando executando uma funcionalidade; *segurança*, que se refere à segurança das informações registradas e das operações realizadas, e; *satisfação* do sistema no que se refere ao resultado desejado quando utilizado. Essas características podem ser avaliadas através de formulários com uso de escalas como definido na versão anterior da ISO9126. Além disso, sugerimos que o usuário identifique quais as melhores e piores funcionalidades e as funcionalidades que ele nunca usa. Essas questões finais dá uma idéia do que o usuário pensa sobre o sistema.

### 3.2 Planejamento

Esta fase consiste apenas da atividade de Planejamento da Redocumentação cujo objetivo é identificar como será feita a redocumentação identificando o que pode ser feito de forma automatizada ou semi-automatizada, e o que deve ser feito manualmente. Essa atividade gera como produto o Plano de Redocumentação.

Para elaborar esse plano deve ser considerados estado de desorganização do sistema e as fontes de informação já identificadas. Além disso, deve-se decidir sobre um dos seguintes caminhos de redocumentação (conforme definido em [5]):

i) Documentação enquanto mantém ("document when touched") - nesse caso pode não ser necessário redocumentar completamente a aplicação. Cada parte do sistema deve ser completamente redocumentada quando ela estiver sofrendo alguma manutenção. Ao longo do tempo, a documentação relevante e útil do sistema vai evoluindo;
ii) Documentação completa – necessária quando o sistema é considerado crítico para o negócio é, portanto de fundamental importância tê-lo totalmente redocumentado. Mesmo nesse caso, no entanto, deve-se reduzir a redocumentação a um mínimo essencial.

Em ambos os casos o responsável pela redocumentação deve avaliar algumas questões para identificar o que deve ser redocumentado. A Tabela 1 mostra possíveis questões, identificando o que deve ser redocumentado e nesse caso se essa documentação pode ser feita

**Tabela 1 – Questões úteis no planejamento da redocumentação**

| Questão a analisar | Redocumentação | Forma |
|---|---|---|
| As funcionalidades do sistema estão documentadas? | Documentar cada funcionalidade considerando:<br>- objetivo<br>-parâmetros de entrada<br>-parâmetros de saída | Manual |
| O fluxo de negócio (controle de seqüência/ chamada das funcionalidades) está definido e documentado? Se sim, existe o mapeamento com o que está implementado no sistema? | Definir o fluxo do negócio geral associando a funcionalidades que respondem a cada aspecto | Manual |
| No caso de programação orientada a objetos: | | |
| – As classes, atributos e associações são conhecidas, entendidas e documentadas? | Documentar classes, atributos e métodos. | Manual |
| – Existe alguma hierarquia de classes definida? | Definir relações entre classes | Automática |
| – Existe identificação das informações de interface, controle e dados, e o mapeamento entre os mesmos? | Definir relações entre classes | Automática |
| No caso de programação estruturada: | | |
| – Existe definição do diagrama de módulos (programa e suas sub-rotinas) para todas as funcionalidades? | Definir Diagrama de Estrutura de Módulos | Automática ou semi-automática |
| – Existe uma especificação de alto nível sobre os processos definidos? | Especificar os módulos identificados | Manual |
| Existe modelo de dados definido? | Definir Modelo de Dados | Automática |
| Se existir, possui dicionário de dados? | Descrever Dicionário de Dados | Manual |
| As regras de integridade relacional são definidas no próprio banco? | Não sendo, documentar cada regra de integridade e definir onde estão sendo implementadas. | Manual |
| As regras do negócio são explicitamente definidas? | Documentar e associar onde estão sendo implementadas | Manual |
| Existe algum documento identificando que funcionalidades lêem, criam, excluem ou atualizam que dados? | Gerar Tabela de cruzamento funções e dados | Automática |
| Existem caminhos de teste definidos? | Documentar teste do sistema | Manual |

de forma automática, semi-automática ou manual. É importante, enfatizar que as questões a serem investigadas não se limitam a essas, e que a depender do tipo da aplicação outras informações pode ser importante de ser redocumentada.

Para cada um dos aspectos anteriores identificados deve-se decidir: o que será documentado manualmente no sentido do que deve ser descrito em cada caso, que ferramentas serão adquiridas ou investigadas para atender algum item de redocumentação, que ferramenta será implementada para atender algum item de redocumentação, e; o pessoal que será responsável por cada item de redocumentação.

### 3.3 Redocumentação

Essa fase consiste da redocumentação propriamente dita englobando quatro atividades: Definição de visão de Alto Nível, Gerar Referências Cruzadas, Definir Subsistemas e Gerar Documentação de Baixo Nível.

A **Definição da Visão de Alto Nível** consiste em descrever a funcionalidade geral do sistema. Além disso, devem ser identificados a interação com outros sistemas, dispositivos de hardware, bancos de dados e operações manuais. O objetivo dessa documentação é prover uma descrição de alto nível do sistema no que se refere as suas grandes funcionalidades. Nesse momento essas funcionalidades são identificadas e descritas em linhas gerais. É importante ressaltar que o interesse não é detalhar a funcionalidade e sim ter uma explicação geral do que é essa funcionalidade no sistema. Deve-se, portanto, inicialmente, identificar as grandes funcionalidades do sistema. Para isso, pode-se, por exemplo, usar o manual do usuário do sistema que (mesmo desatualizado) geralmente está presente em vários sistemas. Para cada funcionalidade identificar perfil do usuário que executa, se requer algum dispositivo de hardware específico, se interage com outros sistemas e se depende da execução de outras funcionalidades do sistema.

Essa atividade pode ser documentada textualmente. No entanto, o uso de diagramas é recomendável por facilitar a melhor visualização das funcionalidades. É reconhecido que, a representação gráfica é sempre bastante útil na documentação de sistemas. Não existe na literatura um diagrama específico que atenda a esse objetivo. No entanto, muitas abordagens existentes podem ser adaptadas de forma a atender esse propósito. Recomenda-se que seja utilizada uma abordagem já familiar na organização por facilitar não só a elaboração do diagrama como seu entendimento. Uma boa alternativa é utilizar Diagramas de casos de uso proposta pela UML (Unified Modeling Language) de forma adaptada. Cada caso de uso vai corresponder a uma funcionalidade do sistema que pode corresponder a várias funções (programas) do sistema. Como na modelagem de caso de uso tradicional pode-se utilizar atores para representar tudo que interage com o sistema sejam pessoas, hardware ou outro sistema. Cada caso de uso deve ser descrito enfatizando seu objetivo e o que atende dentre o do sistema.

A atividade de **Gerar Referências Cruzadas** consiste em gerar (semi)automaticamente referências cruzadas entre dados e funções (programas), dados e dados ou entre funções e funções.

A referência cruzada entre dados e funções é importante para permitir visualizar que dados são criados, atualizados e consultados por que funções. Isso é importante pois auxilia no momento da manutenção a identificar quais as potenciais funções podem ser afetadas por terem sofrido manutenções. Dessa forma para cada função (ou programa do sistema) deve se

identificar quais tabelas tem registros criados (**C**reate), consultados (**R**ead), atualizados (**U**pdate) ou excluídos (**D**elete).

A referência cruzada entre dados e dados é basicamente a geração do modelo de dados correspondente. Atualmente, existem algumas ferramentas no mercado que já fazem essa engenharia reversa. Na verdade, a maioria dos sistemas gerenciadores de banco de dados fornece utilitários com esse fim. No entanto, no caso da integridade referencial estiver feita diretamente no código, deve-se elaborar um programa para varrer o código para definir as relações entre as tabelas.

A referência cruzada entre funções e funções identifica que funções (ou programas) compõem uma funcionalidade principal e quais são utilizados em vários programas. Dessa forma, deve-se buscar listar todos os programas e relacionar quem chama quem.

A atividade de **Definir Subsistemas** consiste de identificar os subsistemas considerando a Visão de Alto Nível e as referências cruzadas identificadas. Essa atividade consiste das seguintes tarefas: (i) relacionar cada função identificada no inventário e utilizada na referência cruzadas com as grandes funcionalidades do sistema identificadas na visão de alto nível; (ii) identificar automaticamente agrupamento de funções que interagem muito entre elas definindo possíveis subsistemas; (iii) analisar os possíveis subsistemas identificados considerando as funções que compõe cada funcionalidade da visão de alto nível; (iv) e definir subsistemas, descrevendo seu objetivo, escopo e identificando as funções (programas) que o compõe. Uma idéia interessante é documentar quais os subsistemas definidos no diagrama utilizado para documentar a visão de alto nível. No caso de Diagrama de Caso de Uso, por exemplo, deve-se circular todos os casos de uso de um mesmo subsistema e colocar o nome internamente ao círculo.

Finalmente, a atividade **Gerar Documentação de Baixo Nível** consiste em gerar documentação para todos os itens identificados na atividade de Planejamento da Redocumentação, ou seja, deve ser documentado todo o item (classes, tabelas, modelos, etc.). Essa documentação consiste basicamente da análise manual de códigos para interpretação e elaboração da documentação. No entanto, alguma documentação automática pode ainda ser gerada conforme a identificação da necessidade pelo analista responsável. Se a empresa possui funcionários que já conhecem o sistema deve destinar essa documentação a esses profissionais. O objetivo aqui é descrever cada função do sistema. Todas funções independentes do tamanho e da complexidade devem ser descritas considerando objetivo, uma descrição geral e parâmetros de entrada e saída. Para cada parâmetro deve ser identificado o formato e uma descrição geral sobre o mesmo. As funções podem corresponder a programas e sub-rotinas ou métodos na programação orientada a objetos.

No caso dos dados não tiverem sido documentados deve-se gerar dicionário de dados para todas as tabelas e seus campos. Outro, aspecto importante é definir as regras de negócio considerando todas as relações dos dados (ou modelo de dados) identificados.

## 4. Uso do Processo de Redocumentação: Lições Aprendidas

Durante os últimos meses temos acompanhado o uso do processo de redocumentação definido acima em quatro instituições: duas privadas (I1 e I2) e duas governamentais (I3 e I4). O acompanhamento de uso nessas empresas mostra algumas dificuldades e benefícios desse processo.

Uma característica comum das quatro instituições é que nenhuma delas designou pessoas para trabalhar exclusivamente com a redocumentação, mas incluíram a atividade de redocumentação como uma atividade a mais da equipe de manutenção. A diferença básica é que as instituições I1 e I2 assumiram o processo de redocumentação como atividades que deveriam ser feitas quando possível, e as instituições I3 e I4 se beneficiaram de ter pessoas com horas de trabalho exclusivamente voltadas para a redocumentação. Vale a pena ressaltar que nesses dois últimos casos, os profissionais estavam realizando a redocumentação como parte de uma pós-graduação em universidade.

Inicialmente, motivados pela possibilidade do resultado decorrente do processo (ter seus sistemas legados redocumentados), as equipes de I1 e I2 trabalharam objetivamente na primeira atividade de Preparação: o Inventário do Sistema. As duas empresas adotaram a solução de implementar um pequeno programa que varresse todo o código e apoiasse na realização da atividade de inventario (listas de classes, funções, métodos etc.). Após concluída essa atividade, se iniciaria a atividade de Auditoria do Sistema. O parecer inicial das duas empresas no momento de iniciar a atividade, é que toda a equipe conhecia os problemas do sistema e podia garantir que o sistema precisava realmente ser redocumentado, pois a qualidade das especificações e da documentação do modelo de dados e código fonte de programa era insuficiente. No entanto, quando requisitados para indicar os problemas de qualidade em cada um dos itens a reação foi a mesma: consideravam o trabalho como desnecessário já que iria se concluir o que todos já sabiam. Apesar, da insistência da equipe de controle de qualidade as pessoas acabaram não realizando essa atividade (embora I2 tenha iniciado). Na empresa I1, a equipe de manutenção foi designada para melhorias urgentes no sistema e o processo de redocumentação foi deixado para um segundo momento. Na empresa I2, foi feito um breve planejamento do que deveria ser redocumentado (fase seguinte à de Auditoria do Sistema) e a equipe de manutenção resolveu aproveitar a contratação eminente de novos funcionários para designar que esses produzissem as referências cruzadas e a documentação de cada código de programa a medida que estivessem aprendendo sobre o sistema. Essa atividade ainda está em andamento.

As instituições I3 e I4 tinham algumas características similares: além de ter uma pessoa com horas designadas de trabalho para realizar as atividades de redocumentação, escolheram como projeto piloto de redocumentação sistemas pequenos que pudessem ser viáveis de ser redocumentados em até três meses (considerando poucas horas de trabalho para pessoal responsável – em torno de cinco por semana).

A instituição I3 concluiu o processo de redocumentação em 64 horas divididas em aproximadamente dois meses de trabalho. O código fonte do programa analisado é na linguagem COBOL e JCL totalizando 42.431 linhas de código. Para realizar a atividade de Inventário do sistema foi utilizada uma ferramenta disponível em mainframes que permite a contagem de programas e tabelas. A Tabela 2 mostra algumas informações do inventário do sistema (para se ter uma idéia da dimensão do sistema) e o tempo de realização das atividades realizadas.

A instituição I4 ainda está com o processo em andamento estando na atividade de Geração de Referências Cruzadas. Até o momento gastou-se 24 horas em aproximadamente um mês e meio de trabalho. O código fonte do programa (batch) analisado é em Cobol totalizando 29.562 linhas de código. Nenhuma atividade foi realizada com apoio de ferramentas. A Tabela 3 mostra algumas informações do inventário do sistema e o tempo de realização das atividades concluídas.

**Tabela 2 – Dados sobre Execução do Processo na Instituição 3**

| Item | Quantidade |
|---|---|
| Subsistemas | 1 |
| Funções do sistema | 49 |
| Classes de negócio | - |
| Classes de apoio | - |
| Métodos por classes - públicos | - |
| Métodos por classes – privados | - |
| Tabelas do negócio | 16 |
| Tabelas auxiliares | 9 |
| Arquivos do negócio | 71 |
| Arquivos auxiliares | 32 |

(a) Parte do Inventário

| Atividade | Qtde(horas) |
|---|---|
| Inventário do sistema | Automático |
| Auditoria do sistema | 4 |
| Planejamento da Redocumetnação | 8 |
| Definição da Visão de alto nível | 12 |
| Gerar Referências cruzadas | 20 |
| Definir subsistemas | - |
| Documentação de baixo nível | 20 |

(b) Horas realizadas por atividade

**Tabela 3 – Dados sobre Execução do Processo na Instituição 4**

| Item | Quantidade |
|---|---|
| Subsistemas | 1 |
| Funções (programas) do sistema | 8 |
| Classes de negócio | - |
| Classes de apoio | - |
| Métodos por classes - públicos | - |
| Métodos por classes – privados | - |
| Tabelas do negócio | 2 |
| Tabelas auxiliares | 1 |
| Arquivos do negócio | 4 |
| Arquivos auxiliares | 5 |

(a) Parte do Inventário

| Atividade | Qtde(horas) |
|---|---|
| Inventário do sistema | 6 |
| Auditoria do sistema | 8 |
| Planejamento da Redocumetnação | 1 |
| Definição da Visão de alto nível | 5 |
| Gerar Referências cruzadas | Em andamento |
| Definir subsistemas | - |
| Documentação de baixo nível | Não realizada |

(b) Horas realizadas por atividade

A partir do acompanhamento de uso desses processos podemos obter algumas lições aprendidas sobre o processo de redocumentação definido nesse artigo:

i) *É importante designar horas de trabalho específicas para a redocumentação* – embora reconheçamos que normalmente as empresas não têm como alocar funcionários exclusivamente para essa atividade, o compartilhamento de pessoas realizando manutenção e redocumentação implica em não realizar a redocumentação adequadamente, visto a urgência (sempre considerada) para as manutenções;

ii) *É necessário o acompanhamento direto da gerência* – a atividade de redocumentação é subestimada e desvalorizada. Os mantenedores de software ainda preferem alterar diretamente o código a entender o sistema através de uma documentação (essa é uma realidade percebida até mesmo no desenvolvimento em que muitos desenvolvedores preferem partir para codificação que gerar documentação). Acreditamos que o processo foi interrompido na instituição I1 e subestimado na instituição I2 pelo fato da própria equipe priorizar as atividades de codificação em detrimento de redocumentação.

iii) *Planejar é importante* – Como tudo em engenharia de software, planejar é sempre importante antes de se realizar uma atividade. Para iniciar a redocumentação

propriamente dita, consideramos necessário analisar o estado do sistema em termos quantitativos (Inventário do sistema) e qualitativos (Auditoria do Sistema) e só então fazer um Planejamento da Redocumentação. Essas três primeiras atividades têm em comum de não resultar diretamente em produtos de documentação. Em conseqüência elas são consideradas menos produtivas e tendem a ser percebidas como inúteis. No entanto, essas atividades são essenciais para se ter uma boa visão do trabalho que a redocumentação representará e da eficiência do processo de redocumentação.

iv) *Formalize opiniões sobre a qualidade do sistema* - Os analistas tendem a considerar que já conhece o suficiente o estado do sistema e não querem "perder tempo" fazendo uma avaliação formal deste estado. A tendência é de não realizar a atividade de Auditoria do Sistema, baseando a atividade de planejamento sobre a qualidade percebida informalmente e sobre considerações gerenciais (utilidade e evolução prevista do sistema). Consideramos esta abordagem errada no sentido que a experiência na instituição I2 confirmou que a percepção intuitiva da qualidade de um sistema, com partes mal conhecidas, tende a ser diferente da qualidade real. O início da auditoria ressaltou aspectos deficientes mas indicou aspectos que não eram tão ruins quanto esperado.

v) *A atividade de Definir Subsistemas nem sempre é necessária* - a definição de subsistemas em uma empresa normalmente seguem regras de negócio estando claramente implementados. No entanto, não excluímos essa atividade pelo fato de existir muitos trabalhos em engenharia reversa para investigação de subsistemas em sistemas legados [13], [14] o que nos leva a crer na sua possível necessidade.

vi) *Sempre que possível deve-se usar ferramentas* – é importante o apoio automatizado principalmente no que se refere as atividades de Inventário do Sistema e de Gerar Referências Cruzadas. O trabalho é mais produtivo e menos suscetível a erros. O uso de ferramentas ou mesmo geração de pequenos programas que apóiem essas atividades pode ser bastante proveitoso. As instituições I1 e I2 não tinham ferramentas prontas e geraram pequenos programas para realizar o inventário.

vii) *Os conceitos de engenharia de software devem ser esclarecidos* - mesmo que os profissionais que estejam responsáveis pela redocumentação tenham experiência no desenvolvimento de software é necessário uma explicação sobre processo de software e mais particularmente sobre o processo de redocumentação a ser utilizado. Percebemos, algumas vezes que os profissionais não têm padronizado definições sobre subsistemas, módulos, funcionalidades e programas; além de algumas vezes não conhecer bem os artefatos gerados no processo.

viii) *Mostre os benefícios sempre que possível* – o processo de redocumentação pode ser longo a depender do tamanho do sistema. É preciso manter a equipe com visão clara dos benefícios a serem gerados: de que forma cada artefato será utilizado, o destaque para o que não se conhecia e se aprendeu sobre o sistema, os resultados já gerados e a comparação com situações do passado onde não se tinha documentação.

## 5. Conclusão

A importância de uma documentação boa e atualizada para realizar manutenção de software é reconhecida pela maioria dos engenheiros de software. Mas também é reconhecido que documentações fracas ou não existentes são uma realidade em muitos sistemas em uso.

Redocumentar esses sistemas é necessário para garantir que futuras manutenções sejam realizadas de forma melhor e com resultados de maior qualidade.

Nesse artigo, apresentamos um processo para realizar a redocumentação de software. Esse processo foi projetado considerando características de engenharia reversa e buscando uma redocumentação que seja mínima o suficiente para ser possível ser gerada e continuamente mantido pela organização.

Apresentamos ainda, lições aprendidas a partir de nossa experiência de acompanhamento de uso desse processo em diferentes instituições.

**Referências Bibliográficas**


[1] NBR ISO/IEC 12207, 1998, Tecnologia de Informação – Processos de Ciclo de Vida de Software, Associação Brasileira de Normas Técnicas, Rio De Janeiro, Brasil.

[2] EMAM, K. E., DROUIN, J. N., MELO, W., 1998, SPICE – The Theory and Practice of Software Process Improvement and Capability Determination, California, IEEE Computer Society.

[3] PAULK, M. C., WEBER, C. V., CURTIS, B., CHRISSIS, M. B. (eds), 1995, The Capability Maturity Model: Guidelines for Improving the Software Process, Addison-Wesley.

[4] LEHMAN, M,M., 1980, On Understanding Laws, Evolution and Conversation in the Large-Program Life Cycle, The Journal of System and Software. 1:213-21.

[5] PRESSMAN, R. S., 2001, Software Engineering: A Practitioner's Approach, 5th edition, McGraw-Hill.

[6] PFLEEGER, S., 2001, Software Engineering – Theory and Practice, New- Jersey, Prentice-Hall Inc., 2a Edição.

[7] von MAYRHAUSER, A., e VANS, A.M., 1994, Dynamic Code Cognition Behaviors For Large Scale Code, Proceedings of 3rd Workshop on Program Comprehension, pp. 74—81.

[8] PENTEADO, R.A.D., 1996,Um método para Engenharia Reversa Orientada a Objetos, Tese de Doutorado, USP, São Carlos.

[9] MULLER, H, JAHNKE, J.H., SMITH, D. et al, 2000, Reverse Engineering: A roadmap, In Antony Finkelstein (ed) – The future of Software Engineering – 22nd International Conference on Software Engineering, pp 47-60.

[10] JACOBSON, I, BOOCH, G e RUMBAUGH, J, 1998, The Unified Software Process, Addison-Wesley.

[11] MIRA KAJKO-MATTSSON, 2001, Motivating the Corrective Maintenance Maturity Model, Proceeding of The International Confernce on Engineering Complex Computer Systems.

[12] ISO/IEC 9126/NBR 13596, 1996, Tecnologia de Informação – Avaliação de Produto de Software – Características de Qualidade e Diretrizes para o seu uso, ABNT – Associação Brasileira de Normas Técnicas.

[13] GIRARD, J-F. KOSCHKE, R., 1997, Finding Components in a Hierarchy of Modules: a Step towards Architectural Understanding, International Conference on Software Maintenance, ICSM'97, pp. 58—65.

[14] ANQUETIL, N., LETHBRIDGE, T.C. 1999, Recovering Software Architecture from the Names of Source Files, Journal of Software Maintenance: Research and Practice, v. 11, pp. 1-21.