# Investigação da Documentação de Maior Importância para Manutenção de Software

Sérgio Cozzetti Bertoldi de Souza[1], Wesley Christian Gonçalves das Neves[1], Káthia Marçal de Oliveira[1], Nicolas Anquetil[1], Rejane Figueiredo[1]

[1] Mestrado em Gestão do Conhecimento e Tecnologia da Informação – Universidade Católica de Brasília (UCB) – Brasília – DF - Brazil
{sergiocozzetti,wchristiang}@yahoo.com.br,
{kathia,anquetil}@ucb.br

**Resumo.** A engenharia de software vem aprimorando métodos, técnicas e ferramentas em busca de qualidade e produtividade no desenvolvimento de software. Neste contexto, a documentação de software tem grande importância, pois pode influenciar tanto na qualidade quanto na produtividade. A falta ou o excesso de documentação pode colocar em risco o desenvolvimento e a manutenção do software. Encontrar um ponto de equilíbrio na quantidade de documentação gerada é de fundamental importância. Este artigo busca identificar na prática quais são os artefatos de software de maior importância para a atividade de obter o entendimento do sistema na fase de manutenção.

## 1 Introdução

Existe uma grande dificuldade das empresas que desenvolvem software cumprirem suas metas de prazo e orçamento. Segundo o Standish Group [19], somente 16,2% dos projetos de software terminaram dentro do prazo e orçamento estimados. Muitas vezes o prazo de entrega pode significar a sobrevivência ou não de uma organização. Como uma tentativa de minimizar este problema, estudos são realizados na área da engenharia de software. Neste contexto, a abordagem da documentação tem grande importância, no qual o prazo de entrega do projeto pode variar de acordo com a quantidade e qualidade dos artefatos de software a serem entregues.

Ambler [1] sugere duas razões básicas para documentar, uma serve para auxiliar a comunicação durante o projeto e outra para auxiliar o entendimento nas atividades de manutenção. A documentação é necessária quando é preciso estabelecer uma comunicação com uma equipe externa de trabalho. Ela serve como mecanismo de suporte a comunicação quando é combinada com reuniões, teleconferência, correio eletrônico e ferramentas colaborativas [1]. A documentação de software tem um efeito significativo no entendimento do programa [5, 21, 22]. Segundo Wong et al [23], mais de 50% do trabalho de evolução do software é dedicado ao entendimento do programa e a tarefa mais difícil da manutenção de software é o entendimento da estrutura do sistema [20].

Cioch et al [5] relataram a importância da abordagem da documentação no sentido de acelerar a produtividade dos mantenedores, levando em consideração o nível de entendimento das mudanças. Em recente pesquisa, Forward e Lethbridge [8] comprovaram que a documentação pode ser útil mesmo se ela estiver desatualizada e concluíram que ela é uma importante ferramenta de comunicação e deve sempre servir a um propósito específico.

Por outro lado, os métodos ágeis vêm ganhando popularidade por causa da rapidez de entrega. Um exemplo é o Extreme Programing (XP), que tem uma visão muito diferente do que se tem visto sobre documentação de software nestas últimas duas décadas e vem se tornando cada vez mais popular [3]. XP se baseia primariamente na comunicação oral, nos testes e no código fonte para comunicar a intenção e a estrutura [2]. Em outras palavras, se fielmente aplicado, não existe nenhum documento de análise e projeto [3]. Dessa maneira, os métodos ágeis conseguem eliminar a primeira necessidade para documentar: auxiliar a comunicação durante o desenvolvimento.

Entretanto, a falta de documentação pode representar um risco para os projetos de software, principalmente na fase de manutenção (segunda necessidade para documentar). Não se pode garantir que o conhecimento, que está na cabeça dos programadores experientes, permanecerá vivo na equipe quando esta for dissolvida. Segundo Freeman e Munro [9], a falta de documentação ou a documentação incorreta afeta a compreensão do sistema, principalmente na atividade de análise de código. Sem documentação adequada, a única fonte de conhecimento é o código fonte [18, 20, 21]. Rajlich [17] afirma que manter sistema com pouca ou sem documentação demanda muito tempo e custa muito caro.

Baseado nestas premissas, a questão é: qual é a documentação mínima que sirva para o propósito de auxiliar o entendimento do sistema na fase de manutenção? Portanto, o objetivo deste artigo é o de identificar na prática quais são os artefatos de software de maior importância para a atividade de obter o entendimento do sistema na fase de manutenção. O resultado desta pesquisa será útil na definição de artefatos a serem gerados no processo de desenvolvimento de software cujo propósito da documentação seja o de dar suporte às atividades de manutenção.

Este artigo aborda alguns aspectos relacionados a manutenção de software (seção 2) e documentação de software (seção 3). A seção 4 relata a estratégia utilizada para identificar na prática a importância dos artefatos de software na manutenção. As conclusões e trabalhos futuros são abordados na seção 5.

## 2 Manutenção de Software

Tradicionalmente, a manutenção de software caracteriza-se pela realização de modificações em um produto de software que já se encontra em funcionamento [6]. Pressman [16] classifica as atividades de manutenção em quatro tipos: corretiva (para corrigir erros não detectados na fase de teste), adaptativa (para acompanhar a evolução inerente à computação), perfectiva (para continuar satisfazendo as necessidades do usuário) e preventiva (para melhorar a confiabilidade ou manutenibilidade futura).

Nos métodos ágeis, como o XP, a manutenção é encarada, na verdade, como sendo um estado normal de um projeto [2]. Essa prática já é adotada nos modelos evolutivos [16], como no USDP (Unified Software Development Process) [11], no qual o projeto é decomposto em várias iterações. Ao final de cada iteração, agrega-se um novo incremento ao projeto, seja ele, correções e/ou melhorias [4]. A produção de novas funcionalidades e a rotatividade dos membros da equipe ocorrem com o sistema em produção.

A maioria dos problemas associados à manutenção de software pode remeter-se a deficiências na maneira segundo a qual o software foi planejado e desenvolvido. Entre os muitos problemas clássicos que podem estar associados à manutenção, alguns estão diretamente relacionados com a documentação. Por exemplo:

- as mudanças não estão adequadamente documentadas,
- freqüentemente, é difícil ou impossível rastrear o processo através do qual o software foi criado,
- a documentação não existe ou é muito ruim.

Pode-se notar a importância de uma documentação eficiente e atualizada para otimizar o processo de manutenção, independente do tipo de processo adotado.

## 3 Documentação de Software

Documentação de software pode ser definida como um artefato cuja finalidade seja comunicar a informação sobre o sistema de software ao qual ele pertence [7]. Entretanto, é necessário distinguir a diferença entre modelos, documentos, código fonte e documentação. Segundo Ambler [1], do ponto de vista da modelagem ágil, um documento é qualquer artefato externo ao código fonte cujo propósito seja transmitir informação de uma maneira persistente. Modelo é uma abstração que descreve um ou mais aspectos de um problema ou de uma solução potencial para resolver um problema. Alguns modelos podem se tornar documentos, ou incluídos como parte deles, ou simplesmente serem descartados quando cumprirem seu papel. Código fonte é uma seqüência de instruções, incluindo os comentários que descrevem estas instruções, para um sistema de computador. O termo documentação inclui documentos e comentários de código fonte.

A documentação de software tem fundamental importância para a Engenharia de Software. Vários estudos tem sido realizados para minimizar os problemas em torno da documentação de software: documentação desatualizada e de baixa qualidade, processos de documentação dispendiosos e caros, documentação em abundância e sem propósito, dificuldade de acesso, entre outros. Alguns pesquisadores propuseram o uso de hipertexto para facilitar o acesso a documentação. Tilley e Müller [21] propuseram combinar documentação e código fonte de uma maneira fácil e eficiente, utilizando ferramenta de hipertexto. Freeman e Munro [9] propuseram um sistema interativo em hipertexto com gráficos e texto. Rajlich [17] propôs a adoção de uma estratégia de redocumentação incremental e oportunística na qual a compreensão obtida na manutenção é armazenada, por meio de anotações, em um sistema de hipertexto baseado na

Web. Tilley e Huang [22] definiram um modelo de maturidade para documentação de software com foco no produto (DMM – Documentation Maturity Model), baseado no CMM (Capability Maturity Model) do SEI (Software Engeneering Institute), para estabelecer parâmetros de avaliação da qualidade da documentação. Ouchi [13] definiu uma estrutura de dados úteis para subsidiar um sistema de documentação voltado para manutenção de software e ressaltou a importância de métodos padronizados de documentação. Medina [14] descreveu os aspectos importantes para uma boa documentação e organização interna dos programas, como por exemplo, a utilização de comentários, identação dos comandos, padronização de nomes de variáveis e espaçamento para enfatizar a estrutura lógica, no sentido de auxiliar o entendimento do programa. O HCi Journal [10] ressaltou que antes do início de cada projeto, se faz necessário um planejamento da documentação a ser utilizada, pois todo desenvolvimento tem características peculiares. Smith et al [18] e Tilley [20] ressaltaram a importância da engenharia reversa na produção de documentação atualizada. Tilley [20] propôs um método flexível de identificação, documentação, representação e apresentação dos aspectos estruturais da arquitetura do software por meio da engenharia reversa chamada de "documenting-in-the-large". A norma ANSI/ANS 10.3-1995 recomendou a divisão da documentação em quatro categorias: resumo, informação da aplicação, definição das funcionalidades e informação do programa [15]. Lethbridge et al [12] indicou a necessidade de empenhar na produção de um simples e poderoso formato e ferramenta de documentação, ao invés de forçar os engenheiros de software realizar um trabalho caro e ineficaz. Cioch et al [5] propôs uma abordagem de documentação que considere o tipo de informação necessária para cada estágio de aprendizagem (recém-chegado, aprendiz, internos, experientes).

Nota-se que os estudos sobre documentação de software visam solucionar problemas de falta de atualização, dificuldade de acesso, falta de qualidade, desorganização, documentação desnecessária e indicam para soluções simples, com o apoio de ferramentas e que sirvam a um propósito.

## 4 Identificação da Documentação de Maior Importância para Manutenção de Software

Para identificar que artefatos de software auxiliam no entendimento do sistema na fase de manutenção foi elaborado um questionário direcionado a profissionais com experiência em manutenção de software. O questionário utilizado foi dividido em duas partes. A primeira com o objetivo de caracterizar o entrevistado contendo nome, e-mail, função, tempo de experiência em manutenção, quantidade de sistemas em que já realizou manutenção e tipo de abordagem conhecida, estruturada e/ou orientação a objetos (Figura 1). Na segunda parte foi formulada a questão "Baseado na sua experiência prática, informe qual a importância dos artefatos de documentação tem na atividade de obter a compreensão do software a ser mantido". Os níveis de importância foram subdivididos em: 1-"Sem importância", 2-"Pouca importância", 3-"Importante", 4-"Muito

importante" e, no caso de o entrevistado não conhecer o artefato, a opção "Não Conheço" (Figura 2).

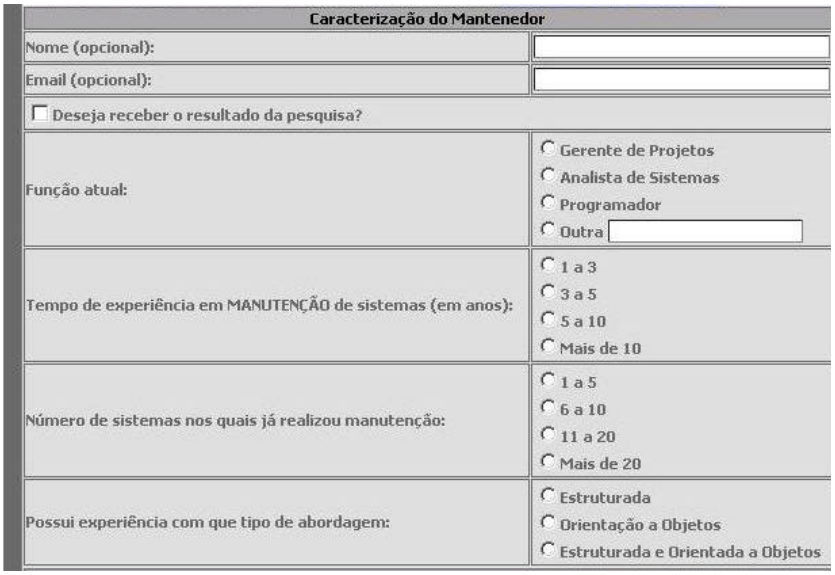


**Figura 1.** Questionário. Parte 1 – Caracterização do entrevistado

Foram apresentados os artefatos mais comuns utilizados nas abordagens da análise estruturada e da orientação a objetos. Eles foram agrupados por fase do ciclo de vida tradicional [16] e tipo de abordagem para facilitar as respostas. A lista de artefatos da análise estruturada teve como base métodos utilizados em algumas empresas nas quais os pesquisadores já trabalharam. Os artefatos da orientação a objeto tiveram como base a UML (Unified Modeling Language) [11].

### 4.1 Análise e Resultados da Pesquisa

O questionário foi disponibilizado na internet e direcionado aos profissionais com experiência prática em manutenção de software por meio da utilização das listas de e-mail dos pesquisadores. Outro meio de obtenção das respostas foi através de questionário em formulário impresso entregue pessoalmente pelos pesquisadores aos entrevistados. Ao todo foram obtidas 67 respostas que possibilitou vislumbrar os resultados a seguir.

A maioria dos profissionais entrevistados foram analistas de sistemas (65%), seguidos por gerentes (25%), programadores (7%) e consultores (3%) conforme ilustrado na Figura 3.

A Figura 4 mostra um considerável nível de experiência dos entrevistados, sendo que a maioria deles (51%) possuía mais de 5 anos de vivência na manutenção de software.

| • Artefatos | | Conheço | | | | Não Conheço |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | |
| **Levantamento** | | | | | | |
| Estruturada | Lista de Requisitos | ○ | ○ | ○ | ○ | ○ |
| | Diagrama de Contexto | ○ | ○ | ○ | ○ | ○ |
| | Descrição dos Requisitos | ○ | ○ | ○ | ○ | ○ |
| Orientação a Objetos | Documento de Visão | ○ | ○ | ○ | ○ | ○ |
| | Diagrama de Caso de Uso | ○ | ○ | ○ | ○ | ○ |
| Estruturada e OO | Modelo Conceitual de Dados | ○ | ○ | ○ | ○ | ○ |
| | Glossário | ○ | ○ | ○ | ○ | ○ |
| **Análise** | | | | | | |
| Estruturada | Funções Derivadas dos Requisitos | ○ | ○ | ○ | ○ | ○ |
| | Diagrama Hierárquico de Funções | ○ | ○ | ○ | ○ | ○ |
| | Diagrama Fluxo de Dados | ○ | ○ | ○ | ○ | ○ |
| Orientação a Objetos | Caso de Uso Especificado | ○ | ○ | ○ | ○ | ○ |
| | Diagrama de Classes | ○ | ○ | ○ | ○ | ○ |
| | Diagrama de Atividades | ○ | ○ | ○ | ○ | ○ |
| | Diagrama de Seqüência | ○ | ○ | ○ | ○ | ○ |
| | Diagrama de Estados | ○ | ○ | ○ | ○ | ○ |
| Estruturada e OO | Protótipo Não Funcional | ○ | ○ | ○ | ○ | ○ |
| | Modelo Lógico de Dados (MER) | ○ | ○ | ○ | ○ | ○ |
| | Dicionário de Dados | ○ | ○ | ○ | ○ | ○ |
| **Projeto** | | | | | | |
| Estruturada | Modelo de Arquitetura | ○ | ○ | ○ | ○ | ○ |
| | Diagrama Geral de Transações | ○ | ○ | ○ | ○ | ○ |
| | Especificação de Componentes | ○ | ○ | ○ | ○ | ○ |
| Orientação a Objetos | Diagrama de Colaboração | ○ | ○ | ○ | ○ | ○ |
| | Diagrama de Componentes | ○ | ○ | ○ | ○ | ○ |
| | Diagrama de Pacotes | ○ | ○ | ○ | ○ | ○ |
| Estruturada e OO | Modelo Físico de Dados | ○ | ○ | ○ | ○ | ○ |
| | Protótipo Funcional | ○ | ○ | ○ | ○ | ○ |
| **Codificação** | | | | | | |
| Estruturada e OO | Comentários do Código Fonte | ○ | ○ | ○ | ○ | ○ |
| | Código Fonte | ○ | ○ | ○ | ○ | ○ |
| **Teste** | | | | | | |
| Estruturada e OO | Plano de Teste Unitário | ○ | ○ | ○ | ○ | ○ |
| | Plano de Teste Sistema | ○ | ○ | ○ | ○ | ○ |
| | Plano de Teste Homologação | ○ | ○ | ○ | ○ | ○ |
| **Implantação** | | | | | | |
| Estruturada e OO | Plano de Migração de Dados | ○ | ○ | ○ | ○ | ○ |
| | Plano de Implantação | ○ | ○ | ○ | ○ | ○ |
| | Manual do Usuário | ○ | ○ | ○ | ○ | ○ |

**Figura 2.** Questionário. Parte 2 – Identificação dos artefatos importantes

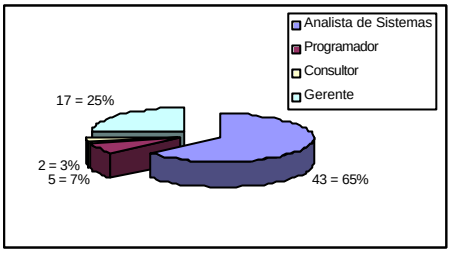


**Figura 3.** Função atual dos entrevistados

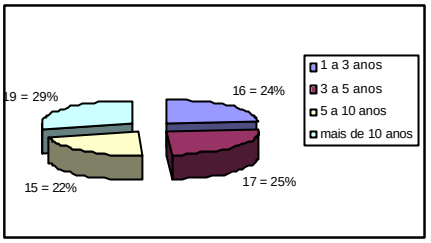


**Figura 4.** Tempo de experiência em manutenção de software

O elevado nível de conhecimento em manutenção de software dos entrevistados foi confirmado pela participação em mais de seis sistemas (Figura 5).

A questão referente ao conhecimento das abordagens (análise estruturada e orientação a objetos) mostrou que a maioria dos entrevistados possuía experiência em ambas a abordagens (62%), conforme Figura 6.

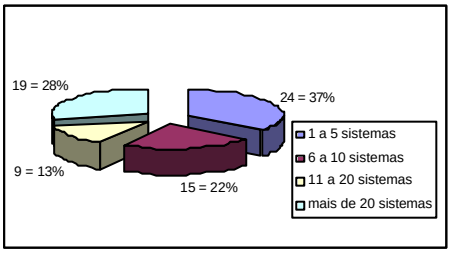


**Figura 5.** Número de sistemas mantidos

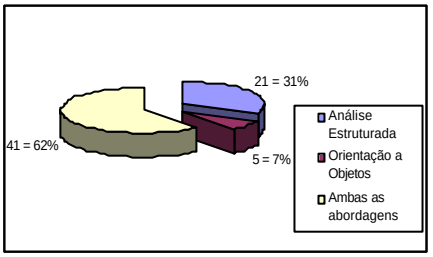


**Figura 6.** Experiência nas abordagens

O principal objetivo da pesquisa é identificar a importância dos diversos artefatos de software para manutenção de sistemas. A Tabela 1 mostra a totalização dos resultados obtidos na aplicação do questionário e informa o número de respostas e o percentual equivalente para cada artefato conhecido pelos participantes da pesquisa.

Com base na Tabela 1 foi possível verificar a importância dos artefatos em cada abordagem (análise estruturada e orientada a objetos) e em cada fase do processo de desenvolvimento de sistemas.

Várias análises puderam ser feitas a partir dos resultados obtidos. Uma delas foi a de verificar que os resultados comprovam a importância da documentação de software para a manutenção, visto que considerando a escala de 4 pontos, mais de 50% das respostas, em cada um dos artefatos, foram nas opções 3 (importante) e 4 (muito importante).

No entanto, o objetivo era identificar os artefatos por abordagem de maior importância para a manutenção de software, ou seja, os que tiveram maior número de respostas igual a4 (muito importante). As Tabelas 2 e 3 mostram os artefatos em que o número de respostas 4 foi maior que 50%.

**Tabela 1.** Totalização dos resultados do questionário

| Artefatos | | Conhecem | | | | Não Conhecem |
|---|---|---|---|---|---|---|
| | | Sem Importância | Pouco Importante | Importante | Muito Importante | |
| **Levantamento** | | | | | | |
| Estruturada | Lista de Requisitos | 6 = 9,6% | 8 = 12,9% | 16 = 25,8% | 32 = 51,6% | 5 = 7,4% |
| | Diagrama de Contexto | 4 = 6,6% | 21 = 35% | 22 = 36,6% | 13 = 21,6% | 7 = 10,4% |
| | Descrição dos Requisitos | 3 = 4,7% | 7 = 11,1% | 15 = 23,8% | 38 = 60,3% | 4 = 5,9% |
| Orientada Objetos | Documento de Visão | 2 = 4% | 12 = 24,4% | 21 = 42,8% | 14 = 28,5% | 18 = 26,8% |
| | Diagrama de Caso de Uso | 0 = 0% | 6 = 10,7% | 20 = 35,7% | 30 = 53,5% | 11 = 16,4% |
| Estrut. e OO | Modelo Conceitual de Dados | 4 = 6,4% | 8 = 12,9% | 22 = 35,4% | 28 = 45,1% | 5 = 7,4% |
| | Glossário | 5 = 7,9% | 21 = 33,3% | 24 = 38% | 13 = 20,6% | 4 = 5,9% |
| **Análise** | | | | | | |
| Estruturada | Funções Derivadas dos Requisitos | 4 = 8,5% | 15 = 31,9% | 18 = 38,2% | 10 = 21,2% | 20 = 29,8% |
| | Diagrama Hierárquico de Funções | 5 = 8,6% | 13 = 22,4% | 27 = 46,5% | 13 = 22,4% | 9 = 13,4% |
| | Diagrama Fluxo de Dados | 5 = 8% | 10 = 16,1% | 28 = 45,1% | 19 = 30,6% | 5 = 7,4% |
| Orientada Objetos | Caso de Uso Especificado | 1 = 1,9% | 4 = 7,6% | 20 = 38,4% | 27 = 51,9% | 15 = 22,3% |
| | Diagrama de Classes | 0 = 0% | 3 = 5,5% | 17 = 31,4% | 34 = 62,9% | 13 = 19,4% |
| | Diagrama de Atividades | 3 = 5,5% | 7 = 12,9% | 28 = 51,8% | 16 = 29,6% | 13 = 19,4% |
| | Diagrama de Seqüência | 0 = 0% | 6 = 10,9% | 31 = 56,3% | 18 = 32,7% | 12 = 17,9% |
| | Diagrama de Estados | 6 = 11,7% | 15 = 29,4% | 24 = 47% | 6 = 11,7% | 16 = 23,8% |
| Estrut. e OO | Protótipo Não Funcional | 8 = 13,7% | 13 = 22,4% | 25 = 43,1% | 12 = 20,6% | 9 = 13,4% |
| | Modelo Lógico de Dados (MER) | 0 = 0% | 3 = 4,6% | 15 = 23,4% | 46 = 71,8% | 3 = 4,4% |

| | | | | | | |
|---|---|---|---|---|---|---|
| | Dicionário de Dados | 1 = 1,5% | 11 = 17,4% | 21 = 33,3% | 30 = 47,6% | 4 = 5,9% |

**Tabela 1 (cont.).** Totalização dos resultados do questionário

| Artefatos | | Conhecem | | | | Não Conhecem |
|---|---|---|---|---|---|---|
| | | Sem Importância | Pouco Importante | Importante | Muito Importante | |
| **Projeto** | | | | | | |
| Estruturada | Modelo de Arquitetura | 5 = 8,7% | 15 = 26,3% | 20 = 35% | 17 = 29,8% | 10 = 14,9% |
| | Diagrama Geral de Transações | 5 = 10,2% | 13 = 26,5% | 21 = 42,8% | 10 = 20,4% | 18 = 26,8% |
| | Especificação de Componentes | 4 = 6,7% | 10 = 16,9% | 28 = 47,4% | 17 = 28,8% | 8 = 11,9% |
| Orient. Objetos | Diagrama de Colaboração | 6 = 12,5% | 17 = 35,4% | 22 = 45,8% | 3 = 6,2% | 19 = 28,3% |
| | Diagrama de Componentes | 5 = 9,6% | 12 = 23% | 25 = 48% | 10 = 19,2% | 15 = 22,3% |
| | Diagrama de Pacotes | 5 = 9,8% | 17 = 33,3% | 26 = 50,9% | 3 = 5,8% | 16 = 23,8% |
| Estrut. e OO | Modelo Físico de Dados | 0 = 0% | 2 = 3,1% | 24 = 37,5% | 38 = 59,3% | 3 = 4,4% |
| | Protótipo Funcional | 7 = 11,4% | 13 = 21,3% | 22 = 36% | 19 = 31,1% | 6 = 8,9% |
| **Codificação** | | | | | | |
| Estrut. e OO | Comentários do Código Fonte | 0 = 0% | 5 = 7,5% | 10 = 15,1% | 51 = 77,2% | 1 = 2,9% |
| | Código Fonte | 0 = 0% | 0 = 0% | 4 = 6,1% | 61 = 93,8% | 2 = 7,4% |
| **Teste** | | | | | | |
| Estrut. e OO | Plano de Teste Unitário | 5 = 8% | 13 = 20,9% | 23 = 37% | 21 = 33,8% | 5 = 7,4% |
| | Plano de Teste Sistema | 4 = 6,3% | 11 = 17,4% | 25 = 39,6% | 23 = 36,5% | 4 = 5,9% |
| | Plano de Teste Homologação | 6 = 9,8% | 9 = 14,7% | 19 = 31,1% | 27 = 44,2% | 6 = 8,9% |
| **Implementação** | | | | | | |
| Estrut. e OO | Plano de Migração de Dados | 7 = 11,6% | 11 = 18,3% | 25 = 41,6% | 17 = 28,3% | 7 = 10,4% |
| | Plano de Implantação | 5 = 7,8% | 10 = 15,6% | 27 = 42,1% | 22 = 34,3% | 3 = 4,4% |
| | Manual do Usuário | 6 = 9,5% | 11 = 17,4% | 22 = 34,9% | 24 = 38% | 4 = 5,9% |

Analisando a Tabela 2, podemos observar que os artefatos de maior importância envolvem:

- requisitos, referentes aos artefatos lista de requisitos e sua descrição na análise estruturada e diagrama de caso de uso e sua especificação na análise orientada a objetos;
- modelos de dados, referentes aos artefatos modelo lógico e físico de dados na análise estruturada. Os mesmos artefatos associados ao diagrama de classe que é a base para a definição desses modelos na análise orientada a objetos, e
- código, referente ao código fonte e comentários no mesmo nas duas abordagens.

Pode-se observar, ainda, que código fonte e seus comentários são artefatos mais importantes, o que é bastante aceitável se tratando de manutenção. No entanto, é curioso observar que o modelo lógico de dados foi considerado mais importante que o modelo físico que, normalmente, representa os dados implementados. Isso leva a crer na preocupação dos profissionais da área de manutenção com o modelo conceitual do sistema, buscando identificar a maior semântica que os modelos podem expressar independente de como foram implementados.

**Tabela 2.** Importância dos artefatos da análise estruturada

| Artefatos | Sem Importância | Pouco Importante | Importante | Muito Importante |
|---|---|---|---|---|
| Código Fonte | 0,0% | 0,0% | 6,2% | 93,8% |
| Comentários do Código Fonte | 0,0% | 7,6% | 15,2% | 77,3% |
| Modelo Lógico de Dados | 0,0% | 4,7% | 23,4% | 71,9% |
| Descrição dos Requisitos | 4,8% | 11,1% | 23,8% | 60,3% |
| Modelo Físico de Dados | 0,0% | 3,1% | 37,5% | 59,4% |
| Lista de Requisitos | 9,7% | 12,9% | 25,8% | 51,6% |

**Tabela 3.** Importância dos artefatos da orientação a objetos

| Artefatos | Sem Importância | Pouco Importante | Importante | Muito Importante |
|---|---|---|---|---|
| Código Fonte | 0,0% | 0,0% | 6,2% | 93,8% |
| Comentários do Código Fonte | 0,0% | 7,6% | 15,2% | 77,3% |
| Modelo Lógico de Dados | 0,0% | 4,7% | 23,4% | 71,9% |
| Diagrama de Classes | 0,0% | 5,6% | 31,5% | 63,0% |
| Modelo Físico de Dados | 0,0% | 3,1% | 37,5% | 59,4% |
| Diagrama de Caso de Uso | 0,0% | 10,7% | 35,7% | 53,6% |
| Caso de Uso Especificado | 1,9% | 7,7% | 38,5% | 51,9% |

## 5 Conclusão

O propósito deste artigo foi identificar na prática quais são os artefatos de software de maior importância para a atividade de obter o entendimento do sistema na fase de manutenção. Pode-se observar que o artefato mais importante é o código fonte, que juntamente com os comentários nele existente reforçam a importância da adoção de padrões de codificação e de organização interna do código conforme relatado por Medina [14]. Outros artefatos que se destacaram foram os modelos de dados (físico e lógico e diagrama de classes), o que ressalta a importância desses modelos estarem atualizados e disponíveis no momento da manutenção. Para a abordagem estruturada, foi dada uma importância muito grande a descrição dos requisitos. Na orientação a objetos, a importância foi observada na especificação dos casos de uso.

O questionário utilizado nesta pesquisa levou em consideração as abordagens estruturada e orientada a objetos, levando em consideração artefatos de software mais comuns nestas abordagens. No entanto, o questionário pode ser adaptado e aplicado para outras abordagens ou métodos específicos alterando-se apenas a listas de artefatos.

Os dados desta pesquisa podem ser úteis para várias análises em torno dos artefatos de documentação de software no sentido de buscar o equilíbrio na produção de artefatos. A partir destes resultados pode ser possível a identificação de quais artefatos podem ser considerados em um método cujo propósito da documentação seja o de gerar artefatos úteis para dar suporte a manutenção de software.

Esta pesquisa foi baseada na opinião de profissionais com experiência em manutenção de sistemas. O próximo passo da pesquisa será o de comprovar a importância de tais artefatos dentro de um ambiente de manutenção de software. Para tanto devem ser usadas métricas para verificar o nível de utilização de cada artefato no momento da obtenção do conhecimento do sistema.

## Agradecimentos

Agradecemos a todos os participantes pela importante contribuição dada a esta pesquisa. Nosso especial agradecimento ao nosso amigo André Luiz Lopes Azevedo por ter desenvolvido a página do questionário na Internet, sem a qual não seria possível obter um número satisfatório de respostas.

O presente trabalho foi realizado com o apoio do CNPq, uma entidade do Governo Brasileiro voltada ao desenvolvimento Científico e Tecnológico.

## Referências


1. AMBLER, Scott W. Agile Documentation. 2001-2004, The Official Agile Modeling (AM) Site, 2001, Disponível em : <http://www.agilemodeling.com/essays/ agileDocumentation.htm>, Acesso em: 02 abr. 2001.
2. BECK, Kent. Extreme Programming Explained: Embrace Change. Boston: Addison-Wesley, 2000.
3. BRIAND, Lionel C.. Software Documentation: How Much is Enough?, Proceedings of the 7th European Conference on Software Maintenance and Reengineering (CSMR), IEEE, 2003, p.13.
4. CARNEIRO, José Morched. Um processo ágil de desenvolvimento de software para web: Proagilweb. 2003. Dissertação (Mestrado em Gestão do Conhecimento e Tecnologia da Informação) - Universidade Católica de Brasília, Curso de Pós-Graduação em Gestão do Conhecimento e Tecnologia da Informação, Brasília-DF, Brasil.
5. CIOCH, Frank A.; PALAZZOLO, Michael; LOHRER Scott. A Documentation Suite for Maintenance Programmers, Proceedings of the International Conference on Software Maintenance (ICSM '96), IEEE, 1996, p.286-295.
6. DIAS, Márcio Greyck Batista. Organizando o conhecimento utilizado na manutenção de software. 2003. Dissertação (Mestrado em Gestão do Conhecimento e Tecnologia da Informação) - Universidade Católica de Brasília, Curso de Pós-Graduação em Gestão do Conhecimento e Tecnologia da Informação, Brasília-DF, Brasil.
7. FORWARD, Andrew. Software Documentation – Building and Maintaining Artefacts of Comunication, 2002. Teste de Mestrado, Universidade de Ottawa, Ottawa, Toronto, Canadá.
8. FORWARD, Andrew; LETHBRIDGE, Timothy C.. The Relevance of Sofware Documentation, Tools and Techologies: A Survey, Proceedings of the 2002 ACM symposium on Document engineering, ACM Press, 2002, p.26-33.
9. FREEMAN, Robert M.; MUNRO, Malcolm. Redocumentation for the Maintenance of Software, Proceedings of the 30th annual Southeast regional conference, p.413-416, ACM, 1992.
10. HCI JOURNAL. What to put in software maintenance documentation. HCi Consulting, 2001, Disponível em : <http://www.hci.com.au>, Acesso em: 06 nov. 2003.
11. JACOBSON, Ivar; BOOCH, Grady; RUMBAUGH, James. The Unified Software Development Process. Addison-Wesley, 2000.
12. LETHBRIDGE, Timothy C.; SINGER Janice; FORWARD, Andrew. How Software Engineers Use Documentation: The State of the Pratice, IEEE Software, Novembro, 2003, IEEE, p.35-39.
13. OUCHI, Miheko L.. Software Maintenance Documentation, Proceedings of the 4th annual international conference on Systems documentation (SIGDOC'85), Nova Iorque, EUA, ACM Press, 1985, p.18-23.
14. MEDINA, Enrique A.. Some Aspects of Software Documentation, Proceedings of the 3rd annual international conference on Systems documentation (SIGDOC'84), Cidade do México, 1984, p.57-59.
15. PHONA, Vir. A Standard for Sofware Documentation, IEEE Computer, Outubro, 1997, p.97-98.

16. PRESSMAN, Roger S. Software Engineering: a practitioner's approach. Nova Iorque: McGraw-Hill, 5ª Edição, 2001.
17. RAJLICH, Václav. Incremental Redocumentation Using the Web, IEEE Software, Setembro/Outubro 2000, p.102-106.
18. SMITH, Dennis; THOMAS, Bill; TILLEY, Scott R.. Documentation for Software Engineers: What is Needed to Aid System Understanding?, Proceedings of the 19th annual international conference on Computer documentation (SIGDOC'01), Santa Fé, Novo México, EUA, Outubro, 2001, p.235-236.
19. THE STANDISH GROUP. The Standish Group Report - CHAOS. The Standish Group, 1995, Disponível em: <http://www.scs.carleton.ca/~beau/PM/Standish-Report.html>, Acesso em: 07 abr. 2004.
20. TILLEY, Scott R.. Documenting-in-the-large vs. Documenting-in-the-small, Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: distributed computing - Volume 2 (CASCON'93), Toronto, Ontário, Canadá, 1993, p.1083-1090.
21. TILLEY, Scott R.; MÜLLER, Hausi A.. INFO: A Simple Document Annotation Facility, Proceedings of the 9th annual international conference on Systems documentation (SIGDOC'91), ACM Press, 1991, p.30-36.
22. TILLEY, Scott R; HUANG, Shihong. Towards a Documentation Maturity Model, Proceedings of the 21st annual international conference on Documentation (SIGDOC'03), San Francisco, Califórnia, EUA, Outubro, 2003, p.93-99.
23. WONG Kenny, TILLEY, Scott R.; MÜLLER, Hausi A.; STOREY Margaret-Anne D.. Structural Redocumentation: A Case Study, IEEE Software, Janeiro, 1995, p.46-54.