UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMATICA
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Filipe Amado Vieira

ESTRATÉGIA DE DOCUMENTAÇÃO NO PROCESSO DE MANUTENÇÃO DE UM SOFTWARE LEGADO

Trabalho de Conclusão de Curso

Recife
2017

Filipe Amado Vieira

ESTRATÉGIA DE DOCUMENTAÇÃO NO PROCESSO DE MANUTENÇÃO DE UM SOFTWARE LEGADO

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Alexandre Marcos Lins de Vasconcelos

Recife
2017

Filipe Amado Vieira

ESTRATÉGIA DE DOCUMENTAÇÃO NO PROCESSO DE MANUTENÇÃO DE UM SOFTWARE LEGADO

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em: _____/_____/_____

_____
Orientador: Alexandre Marcos Lins de Vasconcelos

_____
Examinadora: Simone Cristiane dos Santos

PARECER

AGRADECIMENTOS

Agradeço aos meus pais, Josinete Amado e Bráulio Vieira, pelos grandes sacrifícios realizados que possibilitaram a minha graduação. Serei para sempre grato pelo carinho, incentivo e confiança que depositaram em mim.

Agradeço ao professor Alexandre Vasconcelos pela oportunidade e dedicação, que mesmo à distância me forneceu um grau de atenção que eu nunca esquecerei.

Agradeço ao Centro de Informática, aos professores, colegas e funcionários que contribuíram nessa minha jornada.

Agradeço aos meus colegas de trabalho, que diariamente agregam a minha formação.

Em especial, gostaria de agradecer a minha noiva Laís Sena, sem a qual nada disso estaria acontecendo. A sua força e dedicação foram o meu combustível para seguir em frente. Muito obrigado!

RESUMO

A manutenção é uma das principais fases do ciclo de vida do software. A compreensão do código é a atividade mais custosa nesse processo, pois corresponde a mais da metade do tempo gasto pelos mantenedores. A utilização de documentação técnica é uma forma de mitigar esses custos, contudo, essa documentação é muitas vezes precária ou inexistente, principalmente no contexto de manutenção de um software legado. Neste sentido, este trabalho tem como objetivo fornecer uma estratégia de documentação que facilite os processos de manutenção de um software legado. Para tanto foi realizada uma revisão sistemática da literatura sobre o uso de documentação técnica na fase de manutenção do ciclo de vida do software e algumas propostas de melhoria no processo de documentação de uma instituição pública. As propostas apresentadas sugerem um novo repositório de versionamento, uma forma de priorizar a redocumentação e a utilização de ferramentas para a geração automática de documentação. Espera-se que esse trabalho seja o ponto de partida para alterações significativas dentro da organização abordada e que possa servir como base para novas pesquisas na área de manutenção de software.

Palavras-chave: Documentação. Evolução. Manutenção. Sistemas legados.

ABSTRACT

Maintenance is one of the key phases of the software life cycle. Understanding the code is the costliest activity in this process, accounting for more than half of the time spent by maintainers. The use of technical documentation is a way to mitigate these costs, however, this documentation is often precarious or non-existent, especially in the context of maintaining a legacy software. Thus, this work aims to provide a documentation strategy that facilitates the maintenance processes of a legacy software. For this purpose, a systematic review of the literature about the use of technical documentation in the maintenance phase of the software life cycle and some proposals for improvement in the documentation process of a public institution was carried out. The proposals presented suggest a new versioning repository, a way of prioritizing the redocumentation, and the use of tools for the automatic generation of documentation. It is expected that this work will be the starting point for significant changes within the organization described and that it may serve as the basis for further research in software maintenance.

Keywords: Documentation. Evolution. Maintenance. Legacy System.

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

A manutenção de um software é uma fase essencial do seu ciclo de vida. Mudanças e evoluções ocorrem com a descoberta de falhas, necessidades de melhorias, adaptações a mudanças no ambiente e alterações nos requisitos (BOURQUE; FAIRLEY, 2014). O período de manutenção frequentemente excede em várias vezes o período de desenvolvimento e, por consequência, corresponde à maior parte dos custos de um software (APRIL; ABRAN, 2008).

Segundo a ISO/IEC 14764 de 2006, manutenabilidade, uma característica primária da qualidade, é definida como a capacidade de um produto de software ser modificado. Neste contexto, o desenvolvimento de uma documentação técnica é uma prática importante que impacta positivamente na manutenabilidade, uma vez que mais da metade do esforço de manutenção é devotado ao entendimento do software a ser modificado (BOURQUE; FAIRLEY, 2014). Além disso, a documentação ajuda na análise de impacto das alterações, na identificação da localização no código onde uma mudança precisa ser executada e no entendimento das decisões e soluções tomadas anteriormente.

Muitas organizações mantêm aplicações muito antigas, de alta complexidade e de difícil manutenção, os chamados sistemas legados. Esses sistemas não podem ser simplesmente descartados, pois encapsulam uma grande quantidade de conhecimento e experiência sobre o domínio da aplicação (SOMMERVILLE, 2010). Muitas vezes estão ativos pela necessidade de retorno dos expressivos custos de investimento e por fornecerem serviços críticos fundamentais para a organização.

A falta de documentação associada à saída de pessoal da equipe técnica de desenvolvimento original dificulta a manutenção. Com frequência, os mantenedores são forçados a trabalhar em um software insuficientemente planejado e mal codificado, onde praticamente a única fonte confiável de informação é o código fonte. Isso demanda muito tempo e esforço para entender uma funcionalidade e rastrear o ponto relevante a ser modificado (APRIL et al., 2005).

O conhecimento pessoal adquirido no processo de manutenção é perdido com o tempo. Pessoas tiram férias, ficam doentes, mudam de empresa e até mesmo esquecem a informação. Para grandes equipes é ainda mais complexo manter todos os stakeholders atualizados.

Assim, a criação de documentação para arquivamento e disseminação desse conhecimento é uma solução para esse problema, contudo, tomar decisões sobre a quantidade e profundidade técnica da documentação ainda é um grande desafio. Preocupações sobre seu valor, frequência de uso e utilidade são frequentes, uma vez que ela é vista como uma atividade custosa e de difícil atualização (GAROUSI et al., 2015).

1.2 Objetivos

Neste contexto, o principal objetivo deste trabalho é propor uma estratégia de documentação que facilite os processos de manutenção de um software legado. Mais especificamente, será realizada uma revisão sistemática da literatura sobre o uso de documentação técnica na fase de manutenção do ciclo de vida do software e uma proposta de melhoria no processo de documentação de uma instituição pública.

1.3 Estrutura do Trabalho

Este trabalho apresenta 5 capítulos, incluindo este capítulo introdutório. O capítulo 2 discorre sobre conceitos básicos de documentação, manutenção de software e sistemas legados os quais são necessários para o entendimento das técnicas a serem descritas.

O capítulo 3 apresenta uma revisão sistemática sobre o tema. Nela o protocolo criado para a aplicação da revisão é descrito e são apresentados os principais resultados obtidos. Esses resultados foram utilizados para propor, no capítulo 4, melhorias na estratégia de documentação de um software legado, mantido por uma instituição pública.

No capítulo 5, estão presentes as conclusões do trabalho. Além disso, são mostrados os desafios encontrados e sugestões de trabalhos futuros.

2 CONCEITOS FUNDAMENTAIS

2.1 Manutenção de Software

A ISO/IEC 12207 de 2008 define manutenção como um dos processos primários do ciclo de vida de software. Seu objetivo é a modificação de um sistema de software existente conservando sua integridade. Para isso, faz uso de processos de suporte como documentação, validação, verificação, controle de qualidade, auditoria e tantos outros. Tipicamente, a manutenção inclui modificações no código fonte e na sua documentação devido a problemas ou necessidades de melhoramento.

Apesar do ciclo de vida de manutenção começar após a entrega do software, as atividades de manutenção já ocorrem desde a fase de desenvolvimento (APRIL; ABRAN, 2008). O processo de manutenção começa com o planejamento da manutenção e é executado até a aposentadoria do produto de software. Como o tempo de vida de um software com frequência excede o tempo necessário para produzi-lo, a maior parte dos custos do sistema, 50 a 90%, está relacionada com a sua manutenção.

Manutenção é necessária devido à continua necessidade de mudança dos sistemas. Produtos de softwares são modificados para corrigir falhas, melhorar o design, implementar melhorias, criar interfaces com outros softwares, adaptar o programa para outros hardwares, migrar sistemas legados e atender novas necessidades dos usuários.

As alterações podem ser classificadas como manutenções corretivas, preventivas, adaptativas e perfectivas (ISO/IEC 14764):

- A manutenção corretiva se refere à necessidade de correção de erros, caso o software não atenda aos requisitos estabelecidos.
- A manutenção preventiva se refere a mudanças realizadas com o objetivo de identificar erros em potencial. Ela é bastante utilizada quando os fatores de segurança e risco associados à presença de erros são pontos principais de preocupação.
- A manutenção perfectiva se refere a modificações com foco na melhoria da usabilidade de funcionalidade já existentes, melhorias na documentação

para facilitar a manutenção ou a melhoria de algum outro atributo do software. Ela não agrega novas funcionalidades ao sistema.

- A manutenção adaptativa se refere às alterações necessárias para acomodar mudanças no ambiente, como adaptação a novos requisitos, migração para hardwares mais modernos, alterações de interfaces com outros sistemas. Estudos mostram que esse tipo de manutenção representa mais de 80% das manutenções realizadas.

Os principais problemas do processo de manutenção podem ser divididos em problemas técnicos, gerenciais, de estimação de custos e de geração de métricas (BOURQUE; FAIRLEY, 2014). Dos problemas técnicos, o entendimento limitado do sistema é apontado como um dos principais, pois mais da metade do custo da manutenção é direcionado ao entendimento do software a ser modificado. O processo de documentação pode ser utilizado para mitigar muitos desses custos, pois tem como finalidade auxiliar o entendimento do sistema.

2.2 Documentação Técnica

Documentação de software pode ser descrita como qualquer artefato criado com a intenção de ajudar a comunicar informações sobre o sistema (FORWARD; LETHBRIDGE; 2002). Ela pode se referir a manuais do usuário, comentários de código, documentos de requisitos do sistema, documentos de design, documentos de arquitetura, entre outros tantos artefatos com seus diversos propósitos e níveis de abstração.

A documentação tem um papel importante na área de engenharia de software. Ela explica como o software funciona e como ele foi modelado. Normalmente é gerada para atender diversas necessidades. Estas necessidades podem ser da equipe de desenvolvimento, que utiliza a documentação para gradualmente conceber as soluções de software a serem implementadas; da equipe de manutenção, que precisa ser informada sobre detalhes de implementação e adquirir conhecimento sobre o sistema; ou até mesmo uma necessidade contratual, como quando o desenvolvimento do software é terceirizado.

Os documentos podem ser classificados como textuais ou gráficos (GAROUSI, 2012). Documentos textuais incluem descrições do sistema que podem estar em diferentes formatos como Word, PDF, HTML e arquivos XML. Documentos gráficos incluem planilhas e gráficos, que contam com uma variedade de técnicas de visualização de software para fazer informações complexas mais simples de entender, como digramas UML.

Documentação é criada durante todo o ciclo de vida do software (Figura 1) e serve a diversos propósitos. Durante a fase de análise, os documentos de especificação de requisitos são criados, os quais contêm informações sobre as funcionalidades, limitações e objetivos do sistema. Eles proveem uma forma realista de estimar custos, riscos e cronogramas; fornecem uma base para o desenvolvimento de verificações e validações e ajudam na transferência de produtos de software para novos usuários ou plataformas. Normalmente, são escritos em linguagem natural suplementada por algumas notações mais formais as quais permitem uma descrição mais precisa e concisa de algum requisito ou aspecto da arquitetura em particular (SOMMERVILLE, 2010).

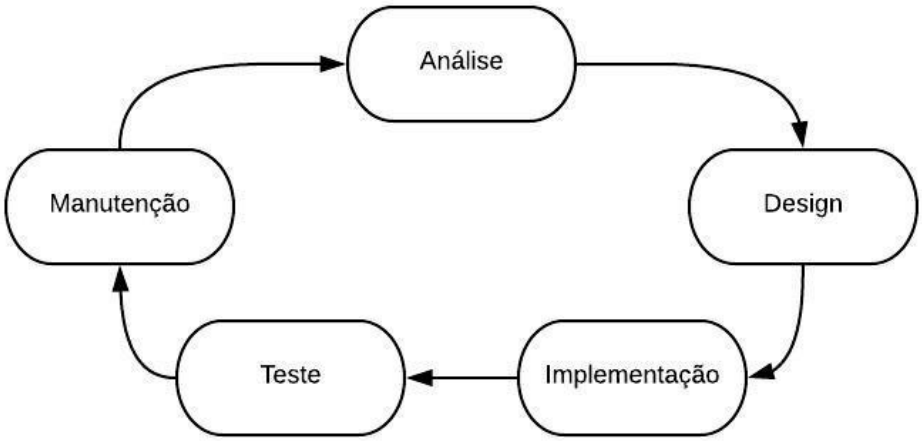


Figura 1: Ciclo de vida do software

Já na fase de design, são criados documentos de modelagem que contêm as abstrações fundamentais do sistema e seus relacionamentos. Cada um dos modelos gerados representa uma visão ou perspectiva diferente sobre o sistema. Eles normalmente são escritos utilizando uma linguagem de notação gráfica, como o Unified Modeling Language (UML), e descrevem as características do software em

detalhes suficientes para que programadores experientes possam desenvolver o sistema com necessidades mínimas de informações adicionais. Os documentos gerados incluem diagramas de processos de negócio, pseudocódigos, diagramas de entidade-relacionamento, dicionário de dados e diagramas UML (BOURQUE; FAIRLEY, 2014; SOMMERVILLE, 2010).

Na fase de implementação, o design do software é concretizado com a programação dos módulos e subsistemas. Os comentários de código são escritos e os documentos gerados em outras fases são alterados e atualizados. Essa fase é fortemente interligada com a fase de testes, na qual são gerados os documentos de planejamento de teste, especificação do design de teste, especificação dos procedimentos de teste, casos de teste e logs de teste. Além disso, os testes por si só já funcionam como uma forma de documentação, pois descrevem o que o código deveria fazer e facilita o seu entendimento (BOURQUE; FAIRLEY, 2014).

Durante a fase de manutenção, os mantenedores devem documentar o problema ou requisição de modificação, a análise dos resultados e as opções de implementação (ISO/IEC 12207). As documentações associadas aos pontos modificados do sistema devem ser atualizadas e caso sejam precárias ou inexistentes, estas podem ser criadas nessa fase. Documentos como especificações, manuais de manutenção, manuais de usuário e guias de instalação são criados ou atualizados quando necessário (ISO/IEC 14764).

Apesar dos benefícios, a adoção do processo de documentação ainda é recebida com bastante resistência. Em contraste com outras disciplinas de engenharia tradicionais, muitos desenvolvedores de software não reconhecem a importância da documentação. Eles correlacionam documentação com milhares de páginas de informações desconexas que ninguém quer escrever e ninguém confia (GAROUSI, 2012). Essa percepção faz com que os desenvolvedores evitem a documentação, não produzam documentos precisos e considerem a documentação somente após o desenvolvimento e não como preparação para tal. Isso contribui para que os artefatos se mantenham incompletos e desatualizados, perpetuando assim essa percepção.

Como dito anteriormente, a atividade de documentar o software é vista como uma atividade muito custosa. Os custos da documentação incluem: a criação, a manutenção e evolução do documento, recuperação das informações e a sua distribuição. Para muitos, esses custos excedem os seus potenciais benefícios, por

isso, se faz necessário identificar a quantidade e profundidade da documentação para se ter um bom custo-benefício.

2.3 Software Legado

Sistemas legados são sistemas de informação antigos desenvolvidos em tecnologias ultrapassadas que continuam em operação por serem críticos para a organização. Eles normalmente são descritos por uma ou mais das seguintes características (KHADKA, 2014):

- Desenvolvido há muitos anos;
- Escritos em linguagem de programação obsoletas;
- Documentação precária ou inexistente;
- Gerenciamento de dados inadequado;
- Estrutura degradada;
- Capacidade de suporte limitada;
- Crescentes custos de manutenção;
- Falta de evolução da arquitetura

Muitos dos sistemas legados de hoje foram construídos há bastante tempo, quando o poder de processamento e a capacidade de armazenamento eram consideravelmente mais limitados. Consequentemente, a eficiência tinha com frequência precedência sobre a legibilidade e manutenabilidade do código, com suas consequências inevitáveis em termos a degradação (KHADKA, 2014). A degradação também está comumente associada com problemas de versionamento, baixa qualidade de documentação, entre outros fatores. Ela tem um impacto profundo nos custos de manutenção.

Alterações nos sistemas legados são inevitáveis. A atualização de novos sistemas e tecnologias em uma organização é constante, o que torna imprescindível que novos sistemas sejam integrados aos antigos. Mesmo sistemas antigos estáveis vão precisar de uma intervenção em algum momento. Porém, esses sistemas resistem significativamente a modificações e evoluções. Os altos custos e riscos envolvidos são restritivos. Uma falha pode acarretar em um impacto significativo ao

negócio (SOMMERVILLE, 2010). Além disso, eles não refletem avanços arquiteturais contemporâneos, como a ênfase no reuso e a construção de componentes. Essas abordagens recentes facilitam a evolução e ajudam a prevenir que esses sistemas se tornem legados com altos custos de manutenção.

3 REVISÃO SISTEMÁTICA

Este capítulo apresenta uma revisão sistemática da literatura cujo protocolo é uma versão adaptada das recomendações propostas por Kitchenham & Charters (2007). Essas recomendações foram utilizadas a fim de tornar os resultados mais confiáveis, auditáveis e de possível reprodução por outros pesquisadores.

A Seção 3.1 apresenta o protocolo criado para a aplicação da revisão, enquanto que a Seção 3.2 descreve os principais resultados extraídos dos estudos selecionados. Em seguida, na Seção 3.3, são tecidos comentários sobre as possíveis ameaças à validade dos resultados e a Seção 3.4 finaliza o capítulo com uma conclusão agregando os principais pontos abordados.

3.1 Metodologia de Pesquisa

3.1.1 Objetivo e Perguntas da Pesquisa

O objetivo desta revisão sistemática foi adquirir conhecimento sobre o estado da arte e as práticas no uso de documentação técnica durante o processo de manutenção e evolução de um software. Dessa forma, buscou-se obter informações que podem auxiliar as tomadas de decisão sobre a escolha de estratégias de documentação na manutenção de um software legado.

Quatro questões de pesquisa (QP) foram definidas para guiar a seleção dos estudos e a extração dos dados:

- QP1: Quais são as estratégias de documentação usadas para preservação do conhecimento adquirido durante o processo de manutenção software?
- QP2: Quais os artefatos de documentação mais úteis e utilizados no processo de manutenção de software?
- QP3: Quais são as características da documentação que impactam na sua utilização?

- QP4: Como definir os pontos do sistema com maior prioridade de documentação?

3.1.2 Estratégias de Busca e Fontes de Dados

A pesquisa foi delimitada a publicações do período de janeiro de 2003 a setembro de 2017. Os estudos foram selecionados a partir de bases de busca automática (Tabela 1) as quais foram escolhidas pela sua relevância no meio acadêmico e disponibilidade dos estudos completos a partir do domínio da Universidade Federal de Pernambuco.

Tabela 1: Bases de busca automáticas utilizadas no processo da revisão sistemática

Bases de Dados	**Endereço Eletrônico**
ACM Digital Library	http://dl.acm.org/
IEEE Xplore	http://ieeexplore.ieee.org/
ScienceDirect	http://www.sciencedirect.com/
Scopus	http://www.info.sciverse.com/scopus/
SpringerLink	http://www.springerlink.com/

A identificação dos artigos foi realizada a partir da definição de *strings* de busca derivadas das questões de pesquisa (Quadro 1). As *strings* foram calibradas pela realização de um estudo piloto, através do qual possíveis adaptações que poderiam aperfeiçoar as *strings* foram identificadas. A busca pelos estudos foi realizada nos meses de outubro e novembro de 2017.

Quadro 1: *Strings* de busca utilizadas na revisão sistemática da literatura.

((software) OR (program) OR (system)) AND (documentation) AND ((maintenance) OR (maintainability) OR (evolution))

3.1.3 Procedimento de Seleção dos Estudos

O processo de seleção dos artigos foi desenvolvido em três etapas: avaliação dos títulos, dos resumos e dos textos completos segundo os critérios de inclusão e exclusão.

Os artigos identificados foram incluídos na revisão desde que satisfizessem todos os seguintes critérios de inclusão (CI):

- CI1: Estudos que respondem pelo menos uma das questões de pesquisa;
- CI2: Estudos primários;
- CI3: Estudos acadêmicos ou da indústria;
- CI4: Estudos que se configuram como artigos completos ou resumos estendidos publicados em periódicos ou conferências.

Foram desconsiderados os estudos que satisfizessem pelo menos um dos seguintes critérios de exclusão (CE):

- CE1: Estudos que não tenham sido escritos em inglês ou português;
- CE2: Estudos que claramente tratam de outros assuntos não relacionados às questões de pesquisa;
- CE3: Estudos cujo foco principal não esteja relacionado com documentação de software;
- CE4: Estudos duplicados;
- CE5: Estudos que não apresentem discussões e resultados pautados em métodos e técnicas de pesquisa científica ou experimentação;
- CE6: Estudos não disponíveis para consulta em versão completa e gratuita pela rede institucional da UFPE;

- CE7: Estudos que não se configuram como artigos completos ou resumos estendidos publicados em periódicos ou conferências, tais como: capítulos de livros, produção artística, patentes, resumos, textos em jornais de notícias ou revistas.

3.1.4 Critérios de Qualidade

Os estudos selecionados foram avaliados a partir da aplicação de 11 critérios de qualidade. Esses critérios foram adaptados de MOREIRA (2015), com o propósito de indicar quais são os trabalhos mais relevantes para a pesquisa. Eles tomaram como base o grau de atendimento às questões de pesquisa e a qualidade do trabalho em relação à estruturação, metodologia, objetivos e resultados:

- CQ1: Descreve abordagens para melhorias do processo de documentação de software?
- CQ2: Aborda o contexto de manutenção de um software legado?
- CQ3: Os documentos mais utilizados na manutenção são descritos?
- CQ4: Aponta problemas e dificuldades na utilização de documentação durante a manutenção de software?
- CQ5: Descreve uma abordagem para priorização na documentação de software?
- CQ6: Descreve claramente o objetivo da pesquisa?
- CQ7: Existe uma descrição adequada do contexto?
- CQ8: A metodologia usada foi descrita de forma objetiva?
- CQ9: Os dados coletados foram exibidos de forma clara e objetiva?
- CQ10: Os resultados condizem com os objetivos?
- CQ11: Os resultados contribuem para o estado da arte e estado da prática?

Os critérios de qualidades foram utilizados após os critérios de inclusão e exclusão com o objetivo de apenas ranquear os artigos de acordo com sua relevância para a pesquisa e, com isso, ter uma referência sobre o peso dos artigos em relação aos resultados. Para cada critério específico dos artigos selecionados foi dada uma nota: 0 (não responde), 0,5 (responde parcialmente) ou 1 (responde totalmente). As

notas em todos os critérios foram somadas e os artigos ranqueados. Esses resultados estão disponíveis no Apêndice B.

3.2 Resultados e Discussão

3.2.1 Resultados do Procedimento de Busca

Após realizar a pesquisa nas bases de dados, foi efetuada a avaliação dos artigos em três etapas, conforme descrito na Seção 3.1.3. A Figura 1 ilustra o processo de seleção dos artigos em cada uma das fases. Foram identificados 18576 artigos, dos quais, 1813 estavam indexados simultaneamente em mais de uma base de dados e, portanto, foram excluídos por serem duplicados. Ao final da análise dos títulos dos artigos, 739 estudos foram considerados potencialmente relevantes. Após realizar a análise dos resumos, 104 foram selecionados para a avaliação do texto completo. No final do processo de seleção, 21 estudos preencheram os critérios preestabelecidos.

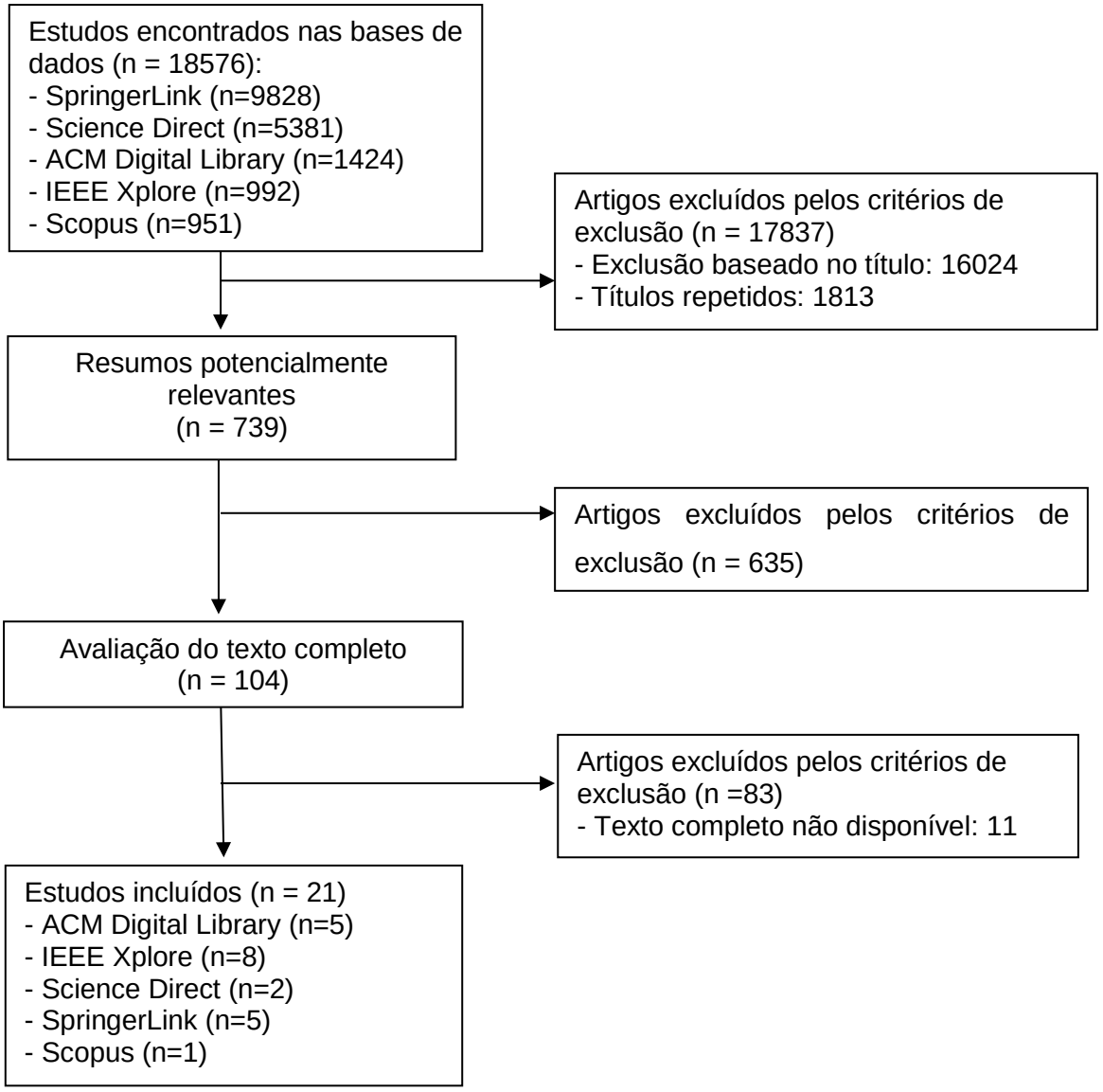


Figura 2: Fluxograma do processo de seleção dos estudos

Observou-se que uma considerável quantidade de estudos retornou da pesquisa nas bases de dados para avaliação inicial e que somente alguns deles foram relevantes para responder as questões de pesquisa. Isso se explica pela baixa precisão de certas bases de dados em relação a string de busca escolhida, em especial a Springerlink e o ScienceDirect. Elas retornaram uma quatindade muito grande de artigos, na sua grande maioria irrelevantes à pesquisa. Uma string de busca mais restritiva não pode ser utilizada, pois outras bases de dados importantes, como IEEE xplore e ACM, acabam não retornando quase nenhum artigo.

Os artigos selecionados, listados no Apêndice A, foram organizados em três categorias: por base de dados, por ano de publicação e por tipo da publicação. A

Tabela 2 apresenta os artigos selecionados por base de dados, onde a maioria dos estudos foi extraída do IEEE Xplore (38,09%).

Tabela 2: Proporção de artigos selecionados por base de dados.

Base de Dados	Número de artigos n (%)	ID
IEEE Xplore	8 (38,09)	E01, E02, E06, E08, E12, E13, E15, E16
ACM Digital Library	5 (23,81)	E05, E07, E10, E11, E14
Springerlink	5 (23,81)	E03, E09, E19, E20, E21
Science Direct	2 (9,52)	E17, E18
Scopus	1 (4,76)	E4
Total	21 (100)	--

ID: Identificação do estudo.

Os dados relacionados ao ano de publicação dos artigos selecionados encontram-se na Tabela 3. Observa-se homogeneidade no número de publicações a cada ano, com um ligeiro aumento nos anos de 2006, 2008 e 2015, mas sem uma diferença significativa dos demais anos.

Tabela 3: Proporção de artigos selecionados por ano de publicação.

Ano de publicação	Número de artigos n (%)	ID
2003	1 (4,76)	E01
2004	1 (4,76)	E02
2005	1 (4,76)	E03
2006	2 (9,52)	E04, E05, E06
2007	1 (4,76)	E07
2008	3 (14,29)	E08, E09, E10
2010	1 (4,76)	E11
2012	1 (4,76)	E12
2013	2 (9,52)	E13, E14
2014	1 (4,76)	E15
2015	3 (19,04)	E16, E17, E18, E19
2016	2 (9,52)	E20, E21
Total	21 (100)	--

ID: Identificação do estudo.

A Tabela 4 apresenta os artigos selecionados por tipo da publicação. Os artigos escolhidos foram do tipo Periódico e Conferência e sua proporção foi de praticamente metade/metade.

Tabela 4: Proporção de artigos selecionados por tipo de publicação.

Tipo de publicação	**Número de artigos n (%)**	**ID**
Periódico	12 (52,38)	E01, E03, E04, E05, E06, E08, E14, E17, E18, E19, E20, E21
Conferência	9 (47,62)	E02, E07, E09, E10, E11, E12, E13, E15, E16
Total	21 (100)	--

ID: Identificação do estudo.

3.2.2 Análise e Discussão do Conteúdo dos Artigos Selecionados

3.2.2.1 Artefatos de Documentação Mais Utilizados

Vários artigos identificados na pesquisa nas bases de dados apontam o código fonte e os comentários de código como as principais e mais utilizadas fontes de informação no processo de manutenção de software. Elas são consideradas as mais confiáveis, atualizadas e disponíveis formas de documentação (E04, E07, E13, E16, E18, E21).

Com relação aos diagramas UML, os diagramas de casos de uso e os de sequência destacam-se como os mais utilizados (E04, E05, E08, E09, E13, E16, E17), porém, alguns estudos, especialmente o trabalho de Arisholm e colaboradores (2006), mostram que diagramas UML parecem ter um impacto relativo na manutenção, dependendo principalmente da complexidade dos artefatos a serem modificados e das tarefas executadas (E06, E10, E11, E14, E17). Tarefas simples são prejudicadas por um volume excessivo de documentação, pois geram uma sobrecarga de informação.

Os mantenedores do sistema são distraídos pelos modelos de análise e gastam mais tempo no entendimento deles do que no entendimento do problema. Por outro lado, tarefas mais complexas se beneficiam do uso de documentação UML, principalmente em relação à corretude das alterações e ao entendimento de partes críticas do sistema.

Além do fator complexidade, a experiência do mantenedor com o UML tem um impacto significativo. A informação adicional fornecida pelos modelos é inútil para mantenedores sem treinamento adequado de como acessá-la. Portanto, um certo grau de experiência é necessário para se beneficiar do uso dos diagramas de casos de uso e sequência (E08, E17).

3.2.2.2 Características Relevantes da Documentação

Legibilidade, tamanho e simplicidade dos documentos foram os fatores mais citados nos artigos como influenciadores do uso de documentação no processo de manutenção, conforme pode ser visualizado na Tabela 5. Isso implica que artefatos menores e mais objetivos são preferíveis a documentos mais densos e completos. Os mantenedores não parecem se beneficiar das informações adicionais e, segundo Fernández-Sáez e colaboradores (2015), “mantenedores nem sempre utilizam a documentação disponível e preferem trabalhar diretamente com o código fonte; mesmo se a documentação estiver disponível, ela não é utilizada”.

Outras características relevantes são o treinamento no uso da documentação, a confiança nas informações e acessibilidade aos artefatos. A falta de treinamento no uso da documentação faz com quem a extração dos dados necessários dos artefatos seja custosa e, por isso, os mantenedores tendem a evita-los. Isso perpetua a falta de confiança nas informações, pois os documentos ficam desatualizados e não condizentes com a realidade do sistema. As barreiras de acesso ao documento também levam eles a ficarem esquecidos e desatualizados, muitas vezes o ninguém tem conhecimento que o artefato sequer existe.

Tabela 5: Frequência de citação das características da documentação que impactam na sua utilização.

Características	Frequência de Citação n (%)	ID
Simplicidade	7 (14,58)	E01, E05, E07, E08, E14, E16, E21
Legibilidade	6 (12,5)	E07, E08, E11, E15, E18, E20
Tamanho	5 (10,42)	E06, E16, E18, E20, E21
Treinamento	4 (8,34)	E03, E06, E08, E17
Acessibilidade	3 (6,25)	E07, E11, E16
Atualização	3 (6,25)	E06, E08, E18
Confiança	3 (6,25)	E11, E16, E20
Precisão	3 (6,25)	E08, E15, E18
Completude	2 (4,17)	E03, E07
Consistência	2 (4,17)	E03, E15
Rastreabilidade	2 (4,17)	E08, E19
Tempo	2 (4,17)	E06, E16
Corretude	1 (2,08)	E03
Detalhamento	1 (2,08)	E16
Estruturação	1 (2,08)	E15
Ferramentas	1 (2,08)	E16
Padronização	1 (2,08)	E16
Relevância	1 (2,08)	E18
Total	48 (100)	--

ID: Identificação do estudo.

3.2.2.3 Priorização de documentação

Ao escolher que partes do sistema devem ser documentadas, o foco deve ser nas partes mais críticas. Documentar os pontos simples não tem um custo-benefício

muito alto, pois os ganhos com entendimento e corretude das alterações são minimizados pelos custos agregados de manter o documento. Outro ponto para se levar em consideração é a frequência com que a documentação será consultada e se existem recursos disponíveis para mantê-la atualizada (E06, E08, E10, E11, E21).

3.2.2.4 Estratégia de Documentação de Software Legado

Uma estratégia de documentação durante a manutenção pode ser traçada baseada nas principais recomendações extraídas dos estudos selecionados:

- Fazer o uso de uma redocumentação “situacional” e “oportunista” documentando o sistema de forma incremental, dando prioridade ao que terá um valor prático. Os critérios determinantes devem ser não só se a documentação gerada será usada, mas também se ela terá ou não condições de ser mantida (E01, E02, E11, E12, E18);
- Focar na qualidade dos comentários de código conforme discutido na Seção 3.2.2.1, os comentários de código são, junto com o código fonte, as principais fontes de informação na manutenção. Eles se destacam como uma forma prática de documentação, pois os desenvolvedores podem documentar ao mesmo tempo que estão programando, sem a necessidade de acessar ferramentas externas. Se beneficiam também de estarem atrelados aos pontos do sistema em que serão úteis. A escrita de bons comentários pode ser auxiliada pela criação de guias de uso (E04, E07, E13, E16, E18, E21);
- Priorizar a documentação das partes complexas do sistema. Para tarefas simples as documentações são muitas vezes supérfluas, mas para tarefas complexas ela tem um melhor custo-benefício (E06, E08, E10, E11, E21);
- Uma vez definidas partes complexas prioritárias do sistema, os diagramas UML de casos de uso e sequência podem ser vistos como uma opção de foco inicial de sua documentação, pois de acordo com Dzidek e colaboradores (2008), eles possuem o melhor custo-benefício como introdução do UML na organização. Ressalta-se que para acessar integralmente os benefícios é necessário certo investimento no treinamento do uso dos modelos (E04, E05, E08, E09, E13, E16, E17);

- A geração automática de documentação é uma solução para tentar responder alguns dos pontos de atrito no uso da documentação, como a atualização, disponibilidade e precisão dos artefatos. Porém é preciso ter cuidado para que os documentos gerados não tendam a ser grandes e complexos (E11), pois estudos apontam para vantagens na criação de documentos mais curtos e simples para a manutenção. Para garantir a legibilidade recomenda-se automatizar apenas o necessário e, quando possível, ter os artefatos revisados por humanos (E06, E11, E16, E18, E20, E21).

3.3 Ameaças a Validade

Com relação à revisão sistemática da literatura descrita no Capítulo 3, apesar do esforço em seguir um protocolo bem fundamentado, algumas limitações foram observadas e podem representar uma ameaça à validade do estudo. Uma delas consiste em somente um avaliador ser destinado a realizar a seleção dos estudos. A subjetividade na interpretação do avaliador pode ter causado a exclusão de estudos relevantes, já que esta problemática não foi corrigida por um revisor. Além disso, de acordo com a Figura 2, onze artigos não puderam ser analisados completamente, pois não se encontravam disponíveis na rede da UFPE e não foi logrado êxito nas tentativas de obter os artigos diretamente com os autores. Possivelmente estudos de interesse não foram incluídos na análise por ocasião desta limitação.

3.4 Considerações Finais

Neste capítulo foi apresentada uma revisão sistemática da literatura com o objetivo de coletar informações relacionadas às práticas de documentação no processo de manutenção de software. Vinte e um estudos foram selecionados, analisados e tiveram informações extraídas a fim de responder as quatro questões da pesquisa.

O código fonte e os comentários de código foram identificados como a principal fonte de informação. Foi evidenciada ainda a necessidade de construção de artefatos mais leves e que foquem apenas em pontos mais complexos do sistema. As principais características da documentação foram levantadas - legibilidade, tamanho, atualização das informações, complexidades dos artefatos - e devem ser levadas em consideração sempre que surgir a necessidade de documentação.

Como sugestão para trabalhos futuros, indica-se a realização de mais estudos na área de documentação na manutenção, pois o número de estudos ainda é muito baixo. Além disso, um foco maior no uso da documentação nos diferentes tipos de manutenção (corretiva, adaptativa, perfectiva e preventiva) pode trazer informações mais específicas e relevantes.

4 PROPOSTAS DE MELHORIAS

Este capítulo apresenta um estudo de caso em uma instituição pública que mantém um sistema legado. A Seção 4.1 consiste em uma descrição do contexto da manutenção desse sistema, abordando as peculiaridades da instituição e as tecnologias utilizadas. Nesta seção também se encontra o resultado de um questionário aplicado a doze desenvolvedores responsáveis pela manutenção do sistema. A Seção 4.2 apresenta algumas propostas de melhoria baseadas no resultado da revisão sistemática do capítulo anterior e nas respostas do questionário para avaliação da prática de documentação.

4.1 Avalição do Estado Atual

A instituição pública em análise possui um sistema legado utilizado diariamente por mais de dois mil servidores. Ele é o principal sistema da instituição e vem sendo desenvolvido e mantido por mais de 10 anos.

O setor responsável pela sua manutenção é composto por vinte e oito servidores, contudo, destes nenhum é dedicado exclusivamente a esta tarefa. A equipe, assim como o próprio sistema, é segmentada seguindo as divisões de negócio da área fim e não seguindo uma distribuição de áreas da TI, como divisões de manutenção, testes e desenvolvimento. Por este motivo, existem muitas replicações de uma mesma tarefa. Alterações em uma funcionalidade semelhante precisam ser replicadas em cada um dos subsistemas.

A confiança do usuário final é baixa devido às diversas inconsistências no sistema. Os recursos necessários para saná-las não estão disponíveis. Ao longo dos anos, as gestões normalmente deram uma ênfase maior à criação de novas funcionalidades do que na melhoria das já existentes, com isso agrega-se cada vez mais complexidades ao sistema.

4.1.1 Tecnologias Utilizadas

O sistema legado em questão é dependente de algumas tecnologias. Elas são a causa de grande frustração dos desenvolvedores, tanto pela falta de treinamento nas ferramentas e linguagens necessárias, quanto pela dificuldade inerente das tecnologias com relação à manutenção dos artefatos criados.

O WEBINSIDE 5.0 (WI), surgido em 2002 sob o nome de WebIntegrator, é a principal ferramenta de desenvolvimento do front-end. Ela originalmente forneceu algumas facilidades, como a criação rápida de páginas web com a utilização de componentes internos da ferramenta e a possibilidade de desenvolvimento e manutenção remotos. Por outro lado, a manutenabilidade do sistema é bastante prejudicada pela sua utilização. Os recursos de depuração do código são praticamente inexistentes, a interface da ferramenta prejudica sua legibilidade e incentiva práticas ruins de programação. Na Figura 3 é mostrada a interface do WI para a programação de um código pseudo-html com um conjunto de marcações nativas a ferramenta.

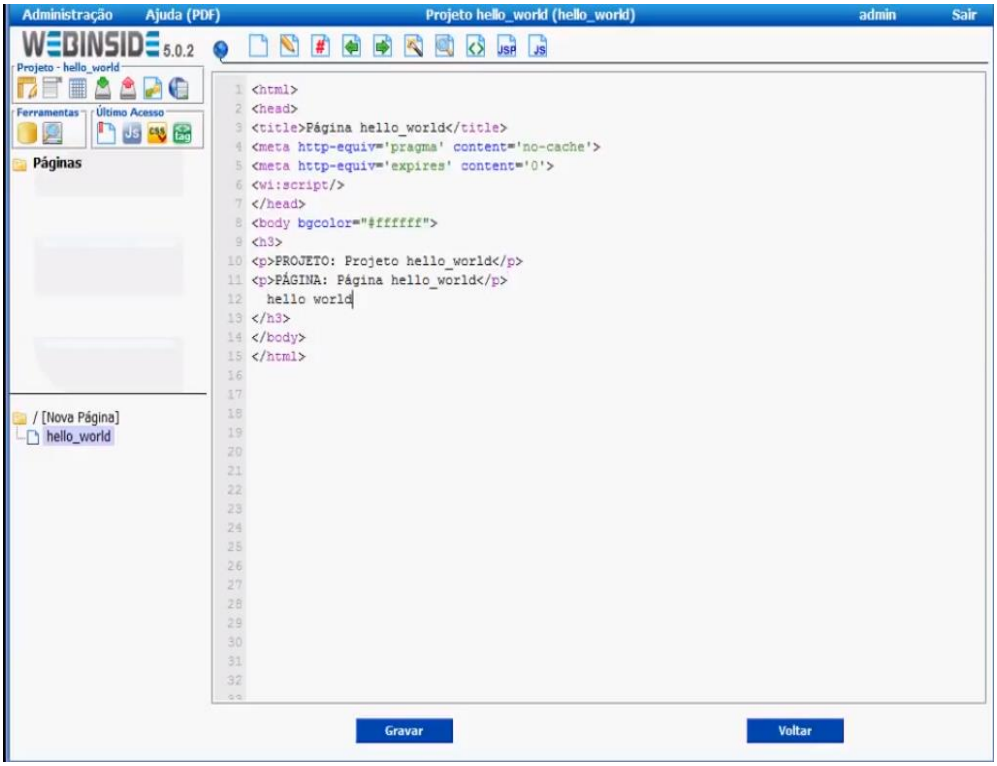


Figura 3: Interface do WebInsider.

O Intersystem Caché é um gerenciador de banco de dados multiparadigma e funciona como a principal base de dados do sistema. Nele os dados podem ser modelados e armazenados como tabelas, objetos ou arrays multidimensionais. Além disso, ele também possui uma camada de desenvolvimento de aplicação, como exposto na Figura 4, para definição de lógica de negócio. Nela é desenvolvido o back-end do sistema utilizando a linguagem Object Script Caché. A precariedade do suporte externo disponível e a falta de treinamento na linguagem acarretam em uma baixa qualidade do código gerado.

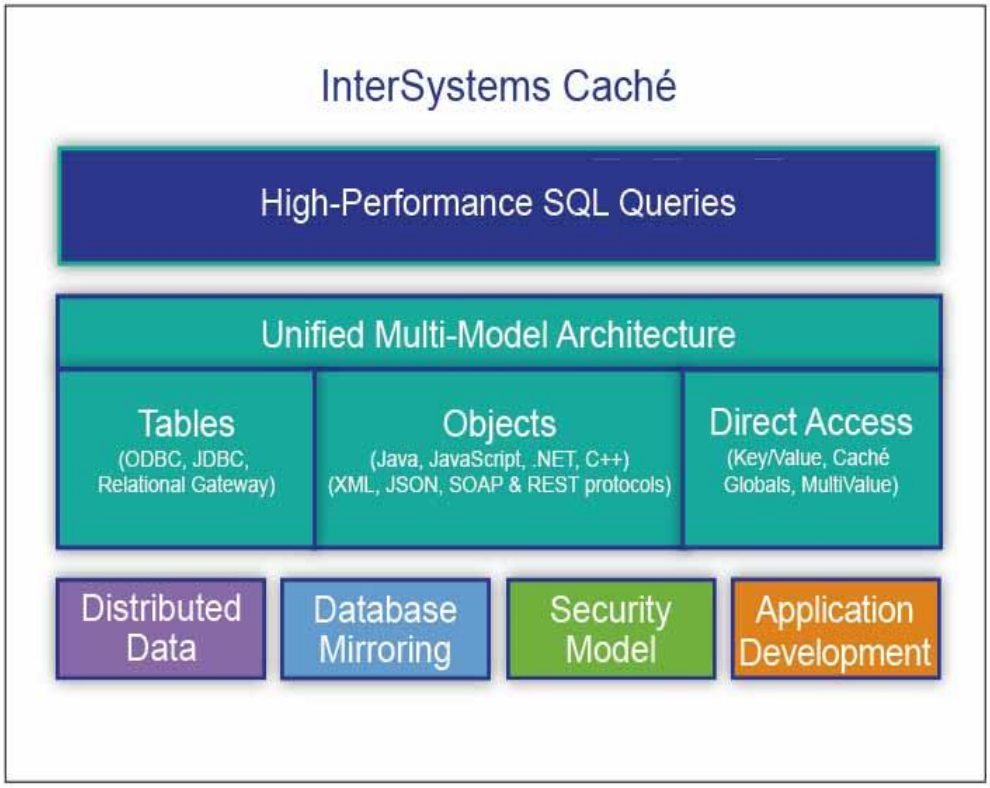


Figura 4: Tecnologias Intersystem Caché.

Atualmente não existe um controle de versão eficiente para este sistema legado. Um repositório SVN é utilizado apenas para armazenamento do código, mas esse não faz parte do ciclo de implantação. Isso se deve à dificuldade de integração do repositório com as tecnologias citadas anteriormente.

Recentemente uma iniciativa de migração dessas tecnologias foi iniciada. Três projetos pilotos estão atualmente em desenvolvimento utilizando uma nova arquitetura. O Angular 4 foi selecionado como a tecnologia a ser utilizada no front-end e serviços REST programados em Java foram selecionados para o back-end. Esses

projetos foram escolhidos por implementarem funcionalidades novas, com necessidades de integrações menores com o sistema legado. No entanto, muitos dos desenvolvedores ainda não estão familiarizados com essas tecnologias, o que gera preocupações sobre a manutenção futura desses módulos.

4.1.2 Questionário de Avaliação

Um questionário foi feito para avaliar a prática de documentação no setor. Ele foi construído tendo como base questionários utilizados em alguns dos artigos selecionados na revisão sistemática (E4, E18). Doze desenvolvedores envolvidos com a manutenção diária do sistema responderam a este questionário durante o mês de novembro de 2017. O questionário completo é encontrado no apêndice C.

A primeira questão aborda o tempo de serviço como parte do quadro da instituição. Os resultados são exibidos na Figura 5.

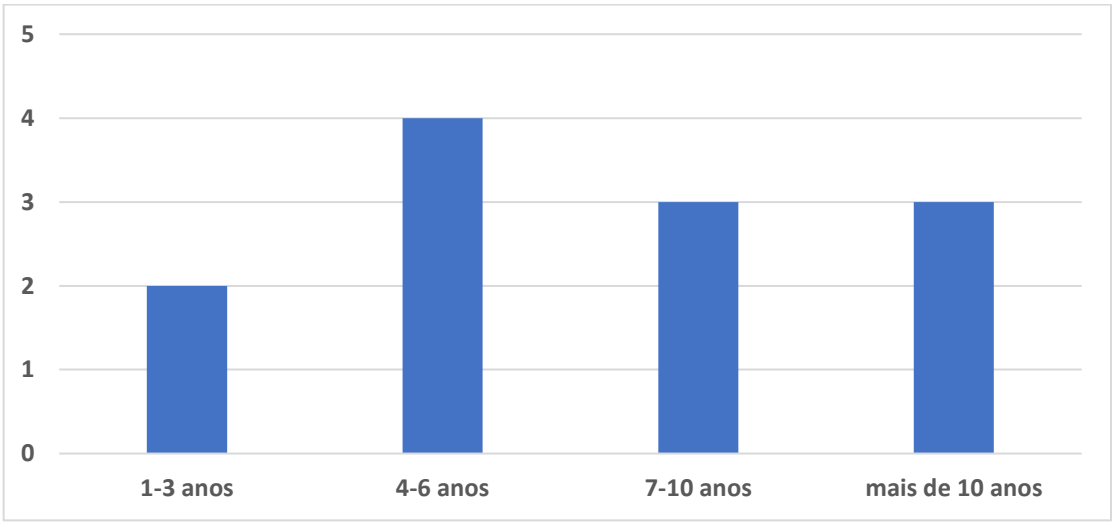


Figura 5: Quantidade de anos trabalhados na instituição pelos servidores.

A grande maioria dos mantenedores não fez parte da concepção original do sistema, metade dos entrevistados tem menos de 6 anos na instituição. Isso se deve à grande rotatividade, com a entrada e saída constante de servidores (desacelerada recentemente devido à situação política e financeira do país).

A segunda questão indaga sobre as fontes de informação mais utilizadas durante a manutenção e o resultado está apresentado na Figura 6. Seguindo o resultado da revisão sistemática, o código fonte se destaca como a principal fonte de informação durante a manutenção. Ficou evidente também a baixa utilização das documentações formais, com os desenvolvedores dando preferência a anotações pessoais.

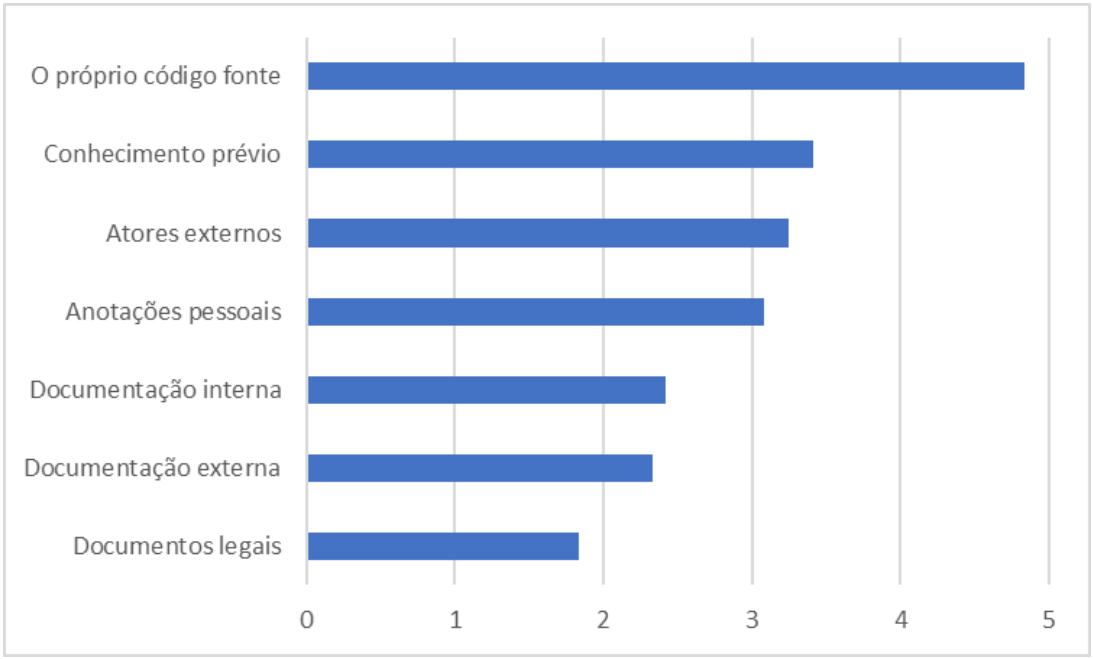


Figura 6: Principais fontes de informação utilizadas durante a manutenção.

As próximas três perguntas são, respectivamente, sobre a frequência de uso, a frequência de atualização e o grau de experiência na criação de certos tipos de documentação. As Figuras 7, 8 e 9 descrevem os resultados. Novamente de acordo com os resultados da revisão sistemática, os comentários de código são os mais utilizados e atualizados dos documentos citados. Outro resultado interessante foi a baixa influência dos comentários do repositório SVN. Isto se explica pela não inclusão do versionamento no fluxo de implantação.

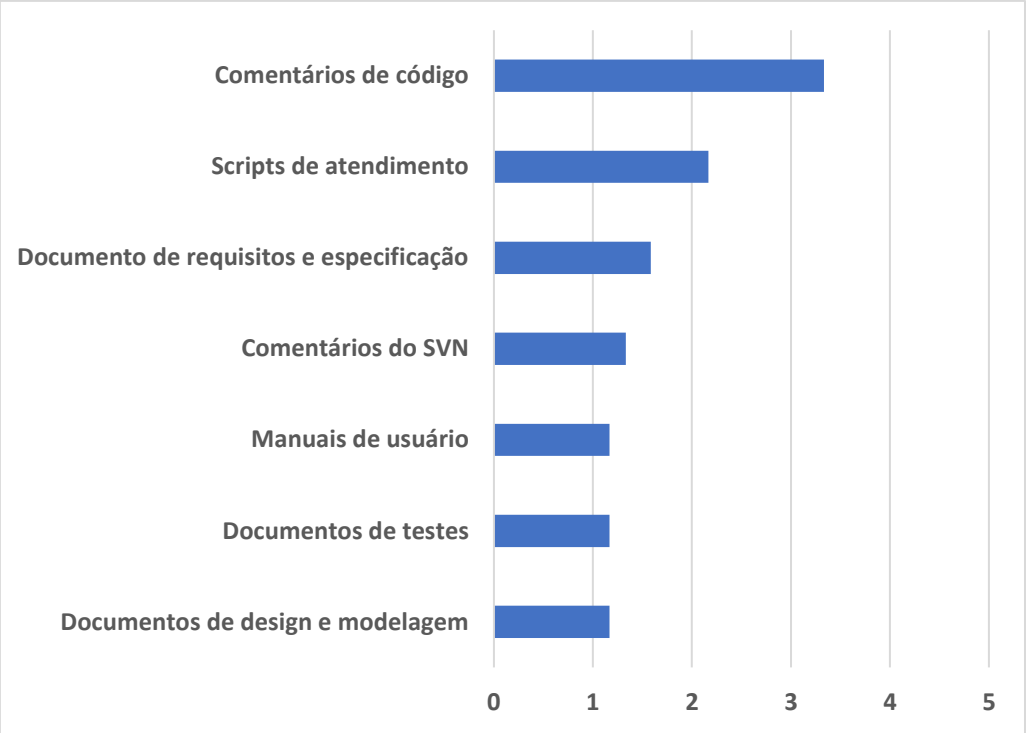


Figura 7: Frequência de utilização por tipo de documento.

A grande maioria dos documentos recebeu uma baixíssima pontuação, tanto em relação à frequência de uso, quanto em relação à sua atualização durante a manutenção, como demonstram as Figuras 7 e 8. Alguns desses documentos, como o de requisitos e o manual do usuário, são gerados por um outro setor responsável pela análise de negócio. A sua confecção não tem envolvimento direto da TI e não está bem definido quem seria responsável por mantê-los.

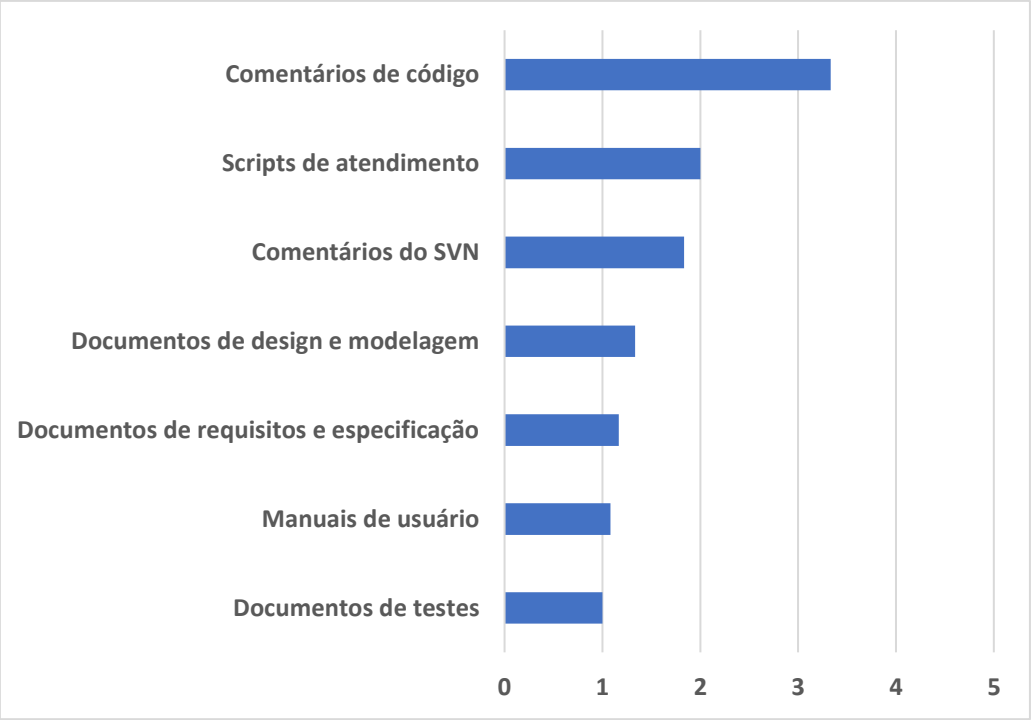


Figura 8: Frequência de atualização por tipo de documento.

Quando os servidores foram solicitados a pontuar o seu grau de experiência com a criação de determinados artefatos de documentação, a escrita de comentários de código teve a maior pontuação. Os documentos de modelagem e design receberam uma pontuação maior em relação a sua criação do que a obtida na sua utilização. Isto pode indicar que alguns desses documentos são criados, mas não são utilizados na manutenção. Estes resultados estão descritos na Figura 9.

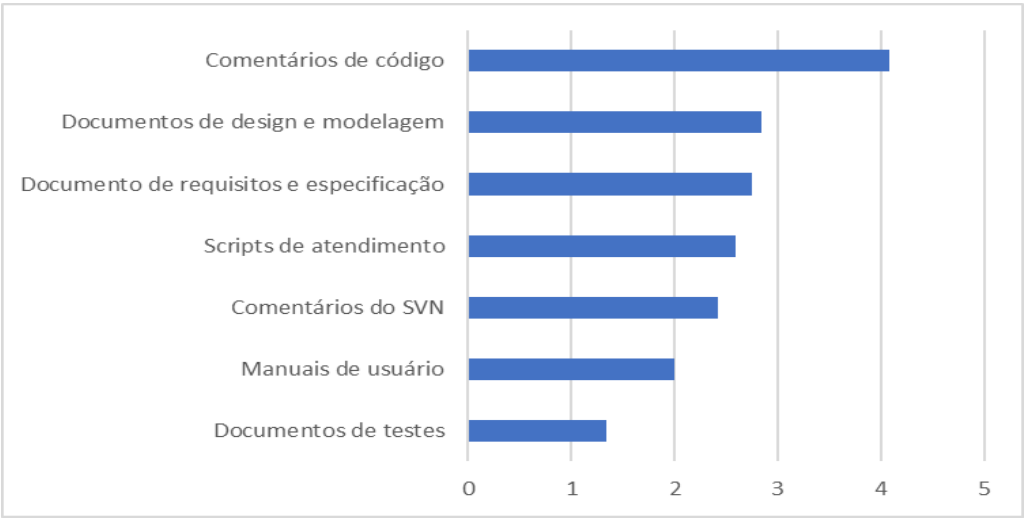


Figura 9: Experiência na criação por tipos de documentos**.**

A próxima questão aborda a utilidade da documentação existente e apresentou um resultado inusitado. Dois dos desenvolvedores avaliaram a documentação existente como muito útil (excelente), enquanto o restante respondeu com uma avaliação média ou péssima. Uma investigação mais aprofundada é necessária para identificar possíveis causas desta disparidade. Os resultados aparecem na Figura 10.

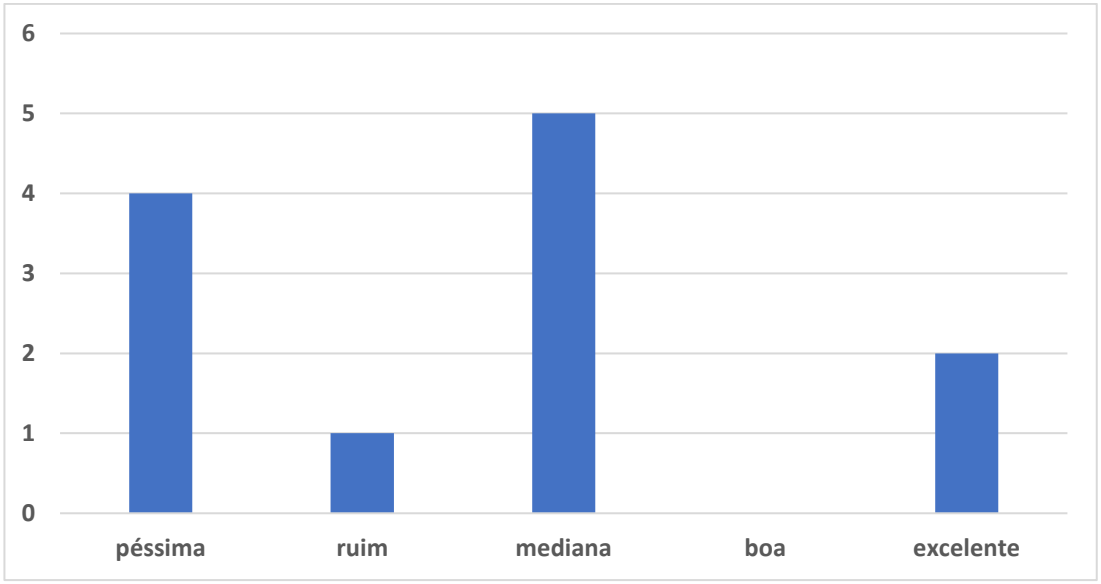


Figura 10: Avaliação da utilidade da documentação existente na instituição pública.

As duas últimas questões lidam com a qualificação da documentação segundo alguns fatores coletados dos artigos da revisão sistemática. A Figura 11 mostra o resultado da avaliação da documentação existente, enquanto a Figura 12 descreve os fatores entendidos pelos entrevistados como mais relevantes em uma documentação. Os resultados mostram que a atualização do conteúdo dos documentos é o fator mais relevante, ao mesmo tempo a avaliação da documentação existente nesse quesito é baixíssima.

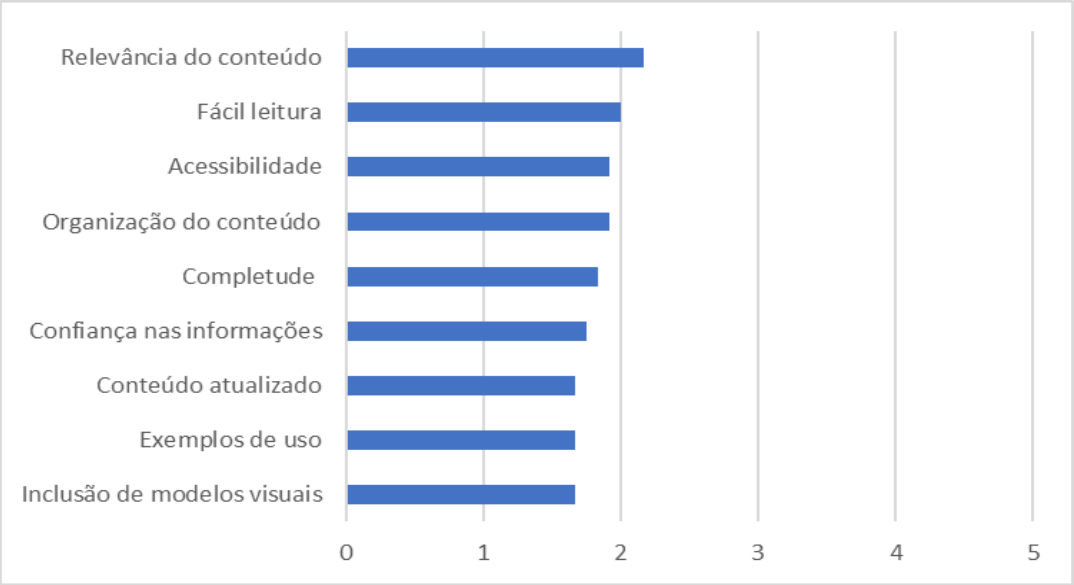


Figura 11: Qualificação da documentação existente.

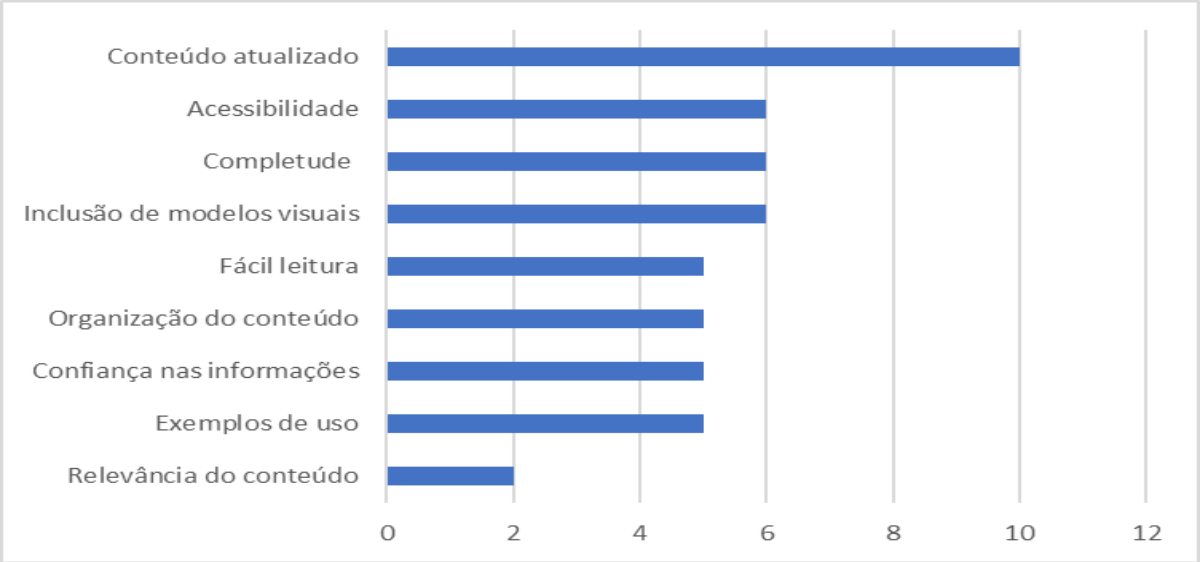


Figura 12: Fatores mais importantes para aumentar a qualidade da documentação.

4.2 Propostas

Nesta seção são descritas propostas de melhoria para o processo de documentação de software de uma instituição pública. O escopo dessas propostas

está limitado a uma equipe de desenvolvimento responsável pela manutenção de um software legado. As recomendações foram propostas levando-se em consideração os resultados da revisão sistemática descrita no Capítulo 3 e do questionário para avaliação da prática de documentação descrito, na seção 4.1 deste capítulo.

4.2.1 Utilização do Git no fluxo de desenvolvimento e implantação

O papel do código fonte como principal fonte de informação na manutenção é indiscutível, porém hoje os mantenedores não têm informações básicas sobre ele. O histórico guardado no SVN, atual repositório do código fonte, não é utilizado com frequência e não é possível rastrear as alterações a uma requisição de modificação específica.

A proposta é a utilização do Git, uma ferramenta de versionamento, no fluxo de desenvolvimento e implantação do software legado. Os custos de implementação dessa proposta eram muito altos e impeditivos, visto que é necessária uma reestruturação geral do setor. Muitos desses custos, como o treinamento do pessoal, foram diluídos com os projetos de evolução tecnológica que já fizeram uso do Git. A inclusão de desenvolvedores nesses projetos foi gradual e por isso foi possível testar o novo fluxo de forma controlada.

O código fonte do software legado já está disponível através do Git, apesar do seu armazenamento ainda estar sendo feito de forma indireta. O objetivo foi fornecer uma forma fácil e padronizada de montagem do ambiente de produção em uma máquina local, problema recorrente do setor e de extremo impacto dada a importância do código fonte como principal meio de obtenção de informações durante a manutenção.

Apesar dos esforços no sentido de migração para novas tecnologias, o sistema ainda está altamente acoplado às tecnologias WebInsider e Caché, descritas na Seção 4.1.1, com as quais o sistema legado foi desenvolvido. Por esse motivo, a proposta inclui a integração do Git com o fluxo de desenvolvimento e implantação nessas tecnologias. A versão mais recente do Caché possui novos recursos que facilitam a integração, apesar de alguns problemas encontrados com a sincronização com os servidores. Já a ferramenta de desenvolvimento WebInsider não possibilita

essa integração de forma fácil, necessitando do desenvolvimento internamente de uma solução. Alguns protótipos para isso já estão sendo desenvolvidos e testados.

4.2.2 Identificação e redocumentação de pontos prioritários

Os pontos principais do sistema são conhecidos, mas como não existem recursos suficientes para abordá-los de uma só vez, uma forma de priorizá-los é necessária. A proposta é a extração de informações de algumas ferramentas utilizadas no setor a fim de identificar os pontos do sistema com necessidades frequentes de manutenção.

Atualmente, coletar essas informações é muito complexo. A introdução do Git na proposta anterior ajudará bastante nesse sentido. Algumas outras ferramentas também podem ser utilizadas. O GLPI é uma ferramenta utilizada no setor para gerenciamento dos incidentes que são categorizados de acordo com os subsistemas envolvidos. As informações estatísticas extraídas dele podem ser confrontadas com os dados extraídos das ferramentas de versionamento (Git e SVN) para se ter uma ideia real das complexidades do sistema e suas necessidades de modificações mais frequentes.

Documentar as partes do sistema mais alteradas, considerando também sua complexidade, garante a utilidade dos artefatos gerados. De acordo com o resultado da revisão sistemática, diagramas UML de casos de uso e sequência serão os artefatos gerados inicialmente, enquanto se faz um estudo na prática das reais necessidades de documentação.

Um fator importante encontrado com a aplicação do questionário foi a baixa experiência dos mantenedores com os diagramas. Por isso, faz parte dessa proposta o treinamento no uso de UML.

4.2.3 Geração Automática de documentos (back-end)

Um dos grandes desafios do setor são os recursos limitados disponíveis para geração e manutenção dos artefatos de documentação. Os altos custos de manutenção do sistema consomem todo o tempo dos mantenedores. O desvio de recursos para documentação é difícil de justificar devido às prioridades e prazos exigidos das demandas existentes. Nesse contexto, a geração automática de documentação é uma solução viável para atender a demanda por uma documentação atualizada, pois não acarreta em um custo expressivo inicial na geração dos artefatos.

A ferramenta Caché Class Explorer tem como objetivo a geração de modelagens UML a partir do código fonte do sistema legado no Intersytem Caché, tecnologia utilizada para o desenvolvimento do back-end da aplicação descrita na Seção 4.1.1. Apesar de não ser uma ferramenta oficial da Intersytem, essa ferramenta é recomendada em seu site (INTERSYSTEM, 2017). Ela pode ser utilizada para geração de artefatos sobre demanda para se ter uma ideia inicial do sistema e assim facilitar outras estratégias de documentação. A Figura 13 apresenta um exemplo de geração de um diagrama de Classes a partir de um código desenvolvido no Caché.

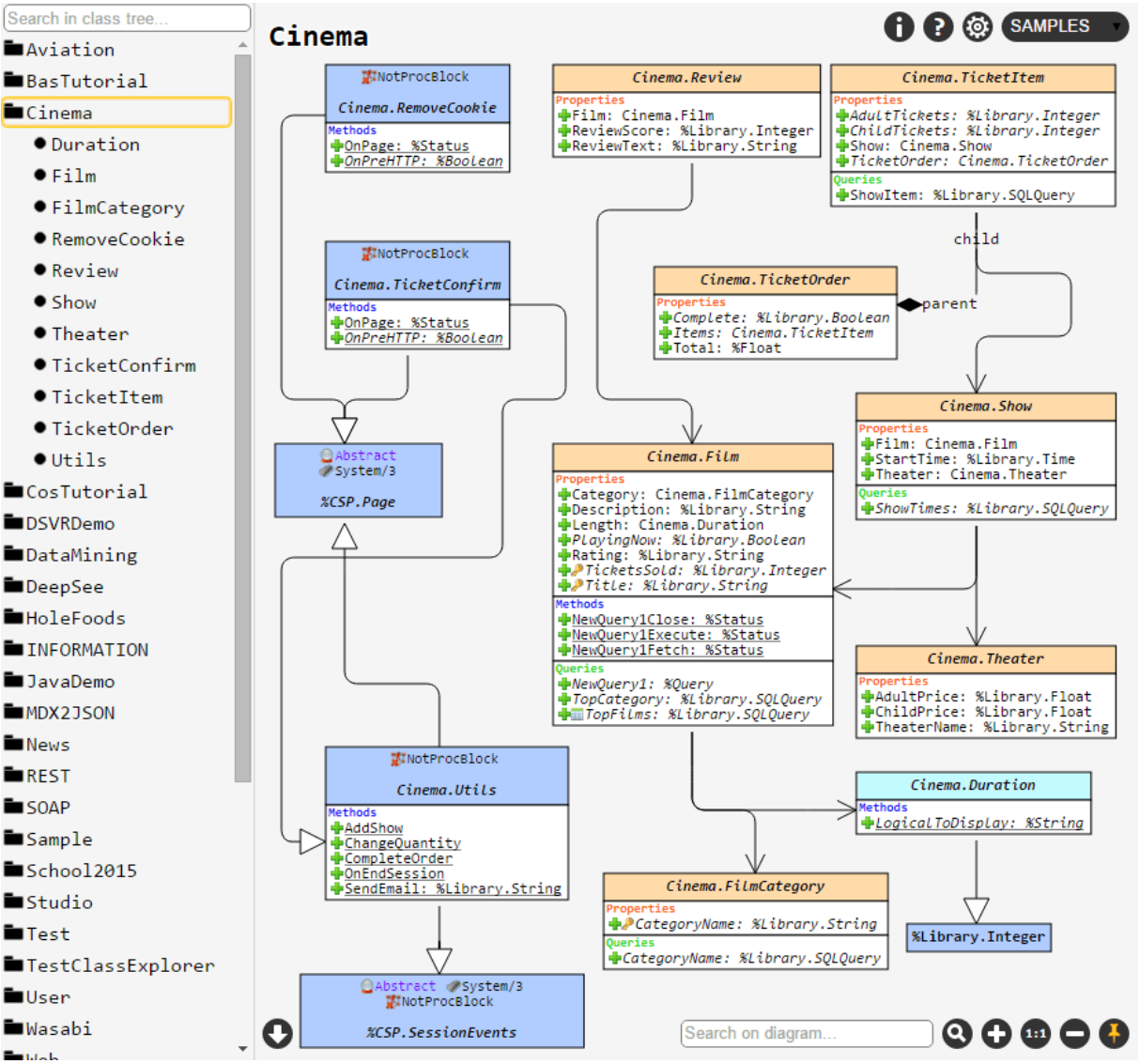


Figura 13: Caché Class Explorer.

4.2.4 Geração Automática de documentos (front-end)

Existe uma preocupação grande que os módulos novos, desenvolvidos em uma nova arquitetura, não sigam o mesmo caminho de outras tentativas fracassadas de evolução tecnológica. A maior preocupação é que se perca a capacidade de manutenção atual, visto que a maioria dos desenvolvedores tem maior familiaridade com a arquitetura antiga.

Diante disso, foi proposta a padronização e utilização de ferramentas para documentação desses novos módulos. Inicialmente, está sendo estudado o uso da

ferramenta COMPODOC para geração automática da documentação do front-end Angular.

Essa ferramenta possui diversas funcionalidades que aumentam a manutenabilidade do sistema. A Figura 14 mostra a interface de visualização dos componentes, módulos e serviços e suas interações criada pela ferramenta para auxiliar o entendimento do código, a tarefa mais custosa da manutenção.

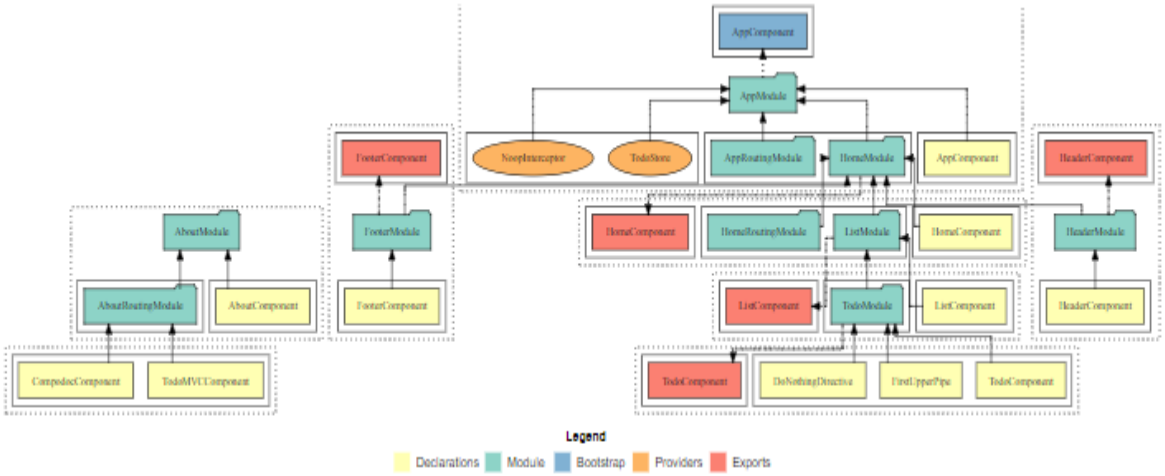
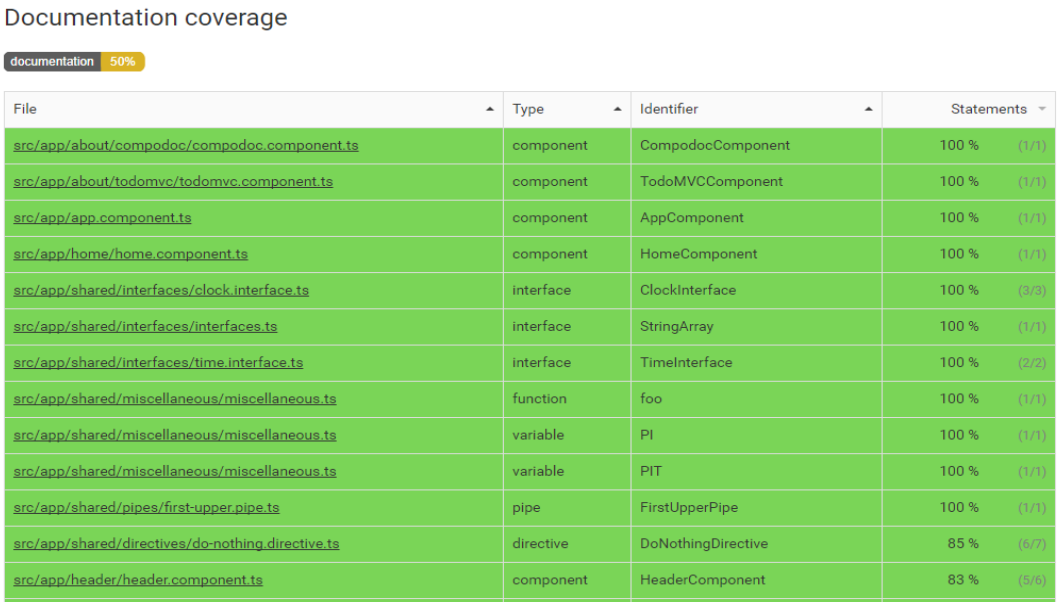


Figura 14: Interface de visualização de componente do COMPODOC.

Outras funcionalidades importantes fornecidas pela ferramenta são as métricas de cobertura. A Figura 15 mostra a interface de visualização do grau da cobertura da documentação dos componentes. Com ela é possível se ter um link direto para os pontos do sistema que necessitam de atenção e é possível avaliar de forma rápida a qualidade da documentação dos componentes sendo criados. Uma interface semelhante está disponível com relação a cobertura dos testes unitários. Essa ferramenta já começou a ser utilizada em um dos projetos de evolução e uma vez testada e avaliada, deverá fazer parte do fluxo principal de desenvolvimento e manutenção do front-end.



Documentation coverage

documentation 50%

File	Type	Identifier	Statements
src/app/about/compodoc/compodoc.component.ts	component	CompodocComponent	100 % (1/1)
src/app/about/todomvc/todomvc.component.ts	component	TodoMVCComponent	100 % (1/1)
src/app/app.component.ts	component	AppComponent	100 % (1/1)
src/app/home/home.component.ts	component	HomeComponent	100 % (1/1)
src/app/shared/interfaces/clock.interface.ts	interface	ClockInterface	100 % (3/3)
src/app/shared/interfaces/interfaces.ts	interface	StringArray	100 % (1/1)
src/app/shared/interfaces/time.interface.ts	interface	TimeInterface	100 % (2/2)
src/app/shared/miscellaneous/miscellaneous.ts	function	foo	100 % (1/1)
src/app/shared/miscellaneous/miscellaneous.ts	variable	PI	100 % (1/1)
src/app/shared/miscellaneous/miscellaneous.ts	variable	PIT	100 % (1/1)
src/app/shared/pipes/first-upper.pipe.ts	pipe	FirstUpperPipe	100 % (1/1)
src/app/shared/directives/do-nothing.directive.ts	directive	DoNothingDirective	85 % (6/7)
src/app/header/header.component.ts	component	HeaderComponent	83 % (5/6)

Figura 15: Cobertura de documentação do código angular.

4.3 Considerações Finais

Neste capítulo foram apresentadas algumas propostas de melhoria para a documentação de um software legado de uma instituição pública. Para tanto, foram utilizados um questionário com oito questões aplicado a doze desenvolvedores responsáveis pela manutenção desse sistema e os resultados da revisão sistemática da literatura descritos na Seção 3.2.

Diante dos resultados, as propostas sugerem a utilização do Git (um novo repositório de versionamento), uma forma de priorização de pontos para redocumentação e a utilização de duas ferramentas para a geração de documentação automática. Espera-se que essas sugestões sejam o ponto de partida de alterações significativas dentro da organização abordada e que possa ser utilizado como base para avaliações e propostas de melhorias futuras.

5 CONCLUSÃO

Neste trabalho foi apresentada uma revisão sistemática da literatura com o objetivo de identificar as práticas de documentação durante o processo de manutenção de software. Com base no resultado desta revisão, foi possível construir um questionário para avaliação do estado atual da prática de documentação de uma instituição pública e, posteriormente, propor melhorias no seu processo de manutenção.

Ao término desse estudo, percebeu-se que diferente da ideia inicial, o uso e utilidade da documentação no processo de manutenção é relativo. Descobriu-se a necessidade de construção de artefatos mais leves e que foquem apenas em pontos mais complexos do sistema. As principais características da documentação foram levantadas - legibilidade, tamanho, atualização das informações, complexidades dos artefatos - e devem ser levadas em consideração sempre que surgir a necessidade de documentação.

Sugere-se a realização de mais estudos na área de documentação para manutenção, pois o número de estudos ainda é muito insipiente. Muitas vezes os estudos focam na comparação entre desenvolvimento e manutenção. Um foco maior no uso da documentação nos diferentes tipos de manutenção (corretiva, adaptativa, perfectiva e preventiva) pode trazer informações mais específicas e relevantes.

Espera-se que esse trabalho seja o ponto de partida de alterações significativas dentro da organização abordada. As propostas apresentadas sugerem um novo repositório de versionamento, uma forma de priorizar a redocumentação e a utilização de ferramentas para a geração automática de documentação. O presente trabalho pode ser usado como base para novas avaliações e propostas de melhorias, assim como para novas pesquisas na área de manutenção de software.

REFERÊNCIAS

APRIL, A.; ABRAN, A. Software Maintenance Management: Evaluation and Continuous Improvement. 1ª ed. Wiley-IEEE Computer Society Press, 2008.

APRIL, A.; HAYES, J.H.; ABRAN, A.; DUMKE, R. Software Maintenance Maturity Model (SMmm): The Software Maintenance Process Model. **Journal of Software Maintenance & Evolution: Research & Practice.**v.17, n.3, p.197-223, 2005.

ARISHOLM, E.; BRIAND, LC.; HOVE, S. E.; LABICHE, Y. The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation. **IEEE Transactions on Software Engineering**. v.32, n.6, p. 365-381, 2006.

BAVOTA, G.; CANFORA, G.; PENTA, M. D.; OLIVETO, R.; PANICHELLA, S. An Empirical Investigation on Documentation Usage Patterns in Maintenance Tasks. In: 29º IEEE INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE, n. 13976882., 2013. Eindhoven: IEEE, 2013. p. 210-219.

BAYER, J.; MUTHIG, D. A View-Based Approach for Improving Software Documentation Practices. In:13º ANNUAL IEEE INTERNATIONAL SYMPOSIUM AND WORKSHOP ON ENGINEERING OF COMPUTER BASED SYSTEMS, n. 9017024., 2006. Potsdam: IEEE, 2006. p.268-278.

BLOM, M.; NORDBY, E. J.; BRUNSTROM, A. An Experimental Evaluation of Documentation Methods and Reusability. In: 2008 INTERNATIONAL CONFERENCE ON SOFTWARE REUSE, n. 5030., 2008. Berlin: Lecture Notes in Computer Science, 2008. p. 372-375.

BOURQUE, P.; FAIRLEY, R.E. Guide to the Software Engineering Body of Knowledge - SWEBOK, v3.0. The Institute of Electrical and Electronics Engineers, Piscataway, 2014.

DAS, S.; LUTTERS, W.G.; SEAMAN, C. B. Understanding Documentation Value in Software Maintenance. In: 2007 SYMPOSIUM ON COMPUTER HUMAN INTERACTION FOR THE MANAGEMENT OF INFORMATION TECHNOLOGY, n.2., 2007. Cambridge: 2007.

DOBING, B.; PARSONS, J. How UML is used? **Communications of the ACM**. v.49, n.5, p.109-113, 2006.

DZIDEK, W. J.; ARISHOLM, E.; BRIAND, L.C. A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance. **IEEE Transactions on Software Engineering**. v.34, n.3, p.407-432, 2008.

FERNÁNDEZ-SÁEZ, A. M.; CAIVANO, D.; GENERO, M.; CHAUDRON, M. R. V. On the use of UML documentation in software maintenance: Results from a survey in industry. In: 18º INTERNATIONAL CONFERENCE ON MODEL DRIVEN ENGINEERING LANGUAGES AND SYSTEMS, 2015. Ottawa: IEEE,.2015.

FERNÁNDEZ-SÁEZ, A.M.; GENERO, M.; CAIVANO, D.; CHAUDRON, M.R.V. Does the level of detail of UML diagrams affect the maintainability of source code?: a family of experiments. **Empirical Software Engineering**. v. 21, n.2016, p.212-259, 2016.

FORWARD, A.; LETHEBRIDGE, T. C. The relevance of software documentation, tools and technologies: a survey. In: PROCEEDINGS OF THE 2002 ACM SYMPOSIUM ON DOCUMENT ENGINEERING. 2002. New York, 2002, p. 26–33.

GAROUSI, G.; YUSIFOGLU, V.; RUHE, G.; ZHI, J.; MOUSSAVI, M.; SMITH, B. Usage and usefulness of technical software documentation: An industrial case study. **Information and Software Technology**. v.57, n.2015, p. 664–682, 2015.

GAROUSI, G. Hybrid Methodology for Analyzing Software Documentation Quality and Usage, 2012, 155f., University of Calgary, Calgary 2012.

GAROUSI, G.; GAROUSI-YUSIFOGLU, V.; RUHE, G.; ZHI, J.; MOUSSAVI, M.; SMITH, B. Usage and usefulness of technical software documentation: An industrial case study. **Information and Software Technology**. v.57, n. 2015, p.664-682, 2015.

GEET, J.V.; EBRAERT, P.; DEMEYER, S. Redocumentation of a Legacy Banking System: An Experience Report. In: INTERNATIONAL WORKSHOP ON PRINCIPLES OF SOFTWARE EVOLUTION. 2010. New York. 2010, p.33-41.

GRAVINO, C.; SCANNIELLO, G.; TORTORA, G. Source-code comprehension tasks supported by UML design models: Results from a controlled experiment and a

differentiated replication. **Journal of Visual Languages & Computing**. v.28, n.2015, p.23-38, 2015.

INTERSYSTEM. Disponível em: https://community.intersystems.com/post/cach%C3%A9-class-explorer-%E2%80%94-exploring-cach%C3%A9-uml-notation. Acessado em: 18 de novembro de 2017.

ISO/IEC 14764 (2006). ISO/IEC 14764 - Software Engineering — Software Life Cycle Processes — Maintenance. International Organization for Standardization/ International Electrotechnical Commission. Piscataway, EUA, 2006.

ISO/IEC 12207 (2008). ISO/IEC 12207:2008 - Systems and software engineering– Software life cycle processes. International Organization for Standardization/ International Electrotechnical Comission. Geneve: v. 8, 2008.

KAJKO-MATTSSON, M. A Survey of Documentation Practice within Corrective Maintenance. **Empirical Software Engineering.** v.10, n. 1, p.31-55, 2005.

KHADKA, R; BATLAJERY, B.V.; SAEIDI, A. M.; JANSEN, S.; HAGE, J. How do professionals perceive legacy systems and software modernization? In: PROCEEDINGS OF THE 36TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 2014. Hyderabad: ACM New York, 2014, p.36-47.

KITCHENHAM, B.; CHARTERS, S. Guidelines for performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE 2007-001, Departament of Computer Science Keele University, Keele, 2007.

LETHBRIDGE, T. C.; SINGER, J.; FORWARD, A. How software engineers use documentation: the state of the practice. **IEEE Computer Society**. v.20, n.6, p. 35-39, 2003

MADER P.; EGYED, A. Do developers benefit from requirements traceability when evolving and maintaining a software system? **Empirical Software Engineering**. v.20, n.2015, p.413-441, 2015.

MOREIRA, R. T. *Um Perfil de Capacidade para a Melhoria do Processo em Micro e Pequenas Organizações Orientadas à Manutenção e Evolução de Produtos de Software.* 2015. 282f. Tese de Doutorado - UFPE, Brasil, 2015.

NUGROHO, A.; CHAUDRON, M.R.V. A Survey into the Rigor of UML Use and its Perceived Impact on Quality and Productivity. In: 2º ACM-IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT. 2008 Kaiserslautern: ACM New York. 2008, p.90-99.

PLOSCH, R.; DAUTOVIC, A.; SAFT, M. The Value of Software Documentation Quality. In: 14º INTERNATIONAL CONFERENCE ON QUALITY SOFTWARE. 2014. Dallas: IEEE. 2014. p. 333-342.

ROSTKOWYCZ, A. J.; RAJLICH, V.; MARCUS, A. A case study on the long-term effects of software redocumentation. In: 20º IEEE INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE. 2004. Chicago: IEEE. 2004, p. 92-101.

SCANNIELLO, G., GRAVINO, C., GENERO, M., CRUZ-LEMUS, J. A.; TORTORA, G. On the impact of UML analysis models on source-code comprehensibility and modifiability. **ACM Transactions on Software Engineering and Methodology**. v. 23, n.2, p.1-26, 2013.

SOMMERVILLE, I. Software Engineering. 9ªed. Addison-Wesley, 2010.

SOUZA, S.C.B.; ANQUETIL, N.; OLIVEIRA, K.M. Which documentation for software maintenance? **Journal of the Brazilian Computer Society**. v.12, n.3, p.31-44, 2006.

TU, Y.C.; TEMPERO, E.; THOMBORSON, C. An experiment on the impact of transparency on the effectiveness of requirements documents. **Empirical Software Engineering**. v.21, n.2015, p.1035-1066, 2016.

APÊNDICE A- DESCRIÇÃO DAS INFORMAÇÕES GERAIS DOS ARTIGOS PARA ANÁLISE EM PROFUNDIDADE

ID	Título	BD	Autoria	Ano	Fonte	Tipo
E01	How software engineers use documentation: the state of the practice	IEEE Xplore	Lethbridge, T. C.; Singer, J.; Forward, A.	2003	IEEE Computer Society	Periódico
E02	A case study on the long-term effects of software redocumentation	IEEE Xplore	Rostkowycz, A. J.; Rajlich, V.; Marcus, A.	2004	20º IEEE International Conference on Software Maintenance	Conferência
E03	A Survey of Documentation Practice within Corrective Maintenance	SpringerLink	Kajko-Mattsson, M.	2005	Empirical Software Engineering	Periódico
E04	Which documentation for software maintenance?	Scopus	Souza, S.C.B.; Anquetil, N.; Oliveira, K.M.	2006	Journal of the Brazilian Computer Society	Periódico
E05	How UML is used?	ACM Digital Library	Dobing, B.; Parsons, J.	2006	Communications of the ACM	Periódico
E06	The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation	IEEE Xplore	Arisholm, E.; Briand, LC.; Hove, S. E.; Labiche, Y.	2006	IEEE Transactions on Software Engineering	Periódico
E07	Understanding Documentation Value in Software Maintenance	ACM Digital Library	Das, S.; Lutters, W.G.; Seaman, C. B.	2007	2007 Symposium on Computer Human Interaction for the Management of Information Technology	Conferência
E08	A Realistic Empirical Evaluation of the Costs and Benefits of UML in Software Maintenance	IEEE Xplore	Dzidek, W. J.; Arisholm, E.; Briand, L. C.	2008	IEEE Transactions on Software Engineering	Periódico
E09	An Experimental Evaluation of Documentation Methods and Reusability	SpringerLink	Blom, M.; Nordby, E. J.; Brunstrom, A.	2008	2008 International Conference on Software Reuse	Conferência
E10	A Survey into the Rigor of UML Use and its Perceived Impact on Quality and Productivity	ACM Digital Library	Nugroho, A.; Chaudron, M.R.V.	2008	2º ACM-IEEE International Symposium on Empirical Software Engineering and Measurement	Conferência
E11	Redocumentation of a Legacy Banking System: An Experience Report	ACM Digital Library	Geet, J.V.; Ebraert, P.; Demeyer, S.	2010	International Workshop on Principles of Software Evolution	Conferência
E12	A View-Based Approach for Improving Software Documentation Practices	IEEE Xplore	Bayer, J.; Muthig, D.	2012	13º Annual IEEE International Symposium and Workshop on	Conferência

Continua

					Engineering of Computer Based Systems	
E13	An Empirical Investigation on Documentation Usage Patterns in Maintenance Tasks	IEEE Xplore	Bavota, G.; Canfora, G.; Penta, M. D.; Oliveto, R.; Panichella, S.	2013	29º IEEE International Conference on Software Maintenance	Conferência
E14	On the impact of UML analysis models on source-code comprehensibility and modifiability	ACM Digital Library	Scanniello, G., Gravino, C., Genero, M., Cruz-Lemus, J. A.; Tortora, G.	2013	ACM Transactions on Software Engineering and Methodology	Periódico
E15	The Value of Software Documentation Quality	IEEE Xplore	Plosch, R.; Dautovic, A.; Saft, M.	2014	14º International Conference on Quality Software	Conferência
E16	On the use of UML documentation in software maintenance: Results from a survey in industry	IEEE Xplore	Fernández-Sáez, A. M.; Caivano, D.; Genero, M.; Chaudron, M. R. V.	2015	18º International Conference on Model Driven Engineering Languages and Systems	Conferência
E17	Source-code comprehension tasks supported by UML design models: Results from a controlled experiment and a differentiated replication	ScienceDirect	Gravino, C.; Scanniello, G.; Tortora, G.	2015	Journal of Visual Languages & Computing	Periódico
E18	Usage and usefulness of technical software documentation: An industrial case study	ScienceDirect	Garousi, G.; Garousi-Yusifoglu, V.; Ruhe, G.; Zhi, J.; Moussavi, M.; Smith, B.	2015	Information and Software Technology	Periódico
E19	Do developers benefit from requirements traceability when evolving and maintaining a software system?	SpringerLink	Mader P.; Egyed, A.	2015	Empirical Software Engineering	Periódico
E20	An experiment on the impact of transparency on the effectiveness of requirements documents	SpringerLink	Tu, Y.C.; Tempero, E.; Thomborson, C.	2016	Empirical Software Engineering	Periódico
E21	Does the level of detail of UML diagrams affect the maintainability of source code?: a family of experiments	SpringerLink	Fernández-Sáez, A.M.; Genero, M.; Caivano, D.; Chaudron, M.R.V.	2016	Empirical Software Engineering	Periódico

Conclusão

ID: Identificação do estudo.

APÊNDICE B – QUALIFICAÇÃO DOS ARTIGOS

ID	CQ1	CQ2	CQ3	CQ4	CQ5	CQ6	CQ7	CQ8	CQ9	CQ10	CQ11	TOTAL
E04	1	0,5	1	1	1	1	1	1	1	1	1	10,5
E08	1	0	1	1	1	1	1	1	1	1	1	10
E16	1	0	1	1	1	1	1	1	1	1	1	10
E18	1	0	1	1	1	1	1	1	1	1	1	10
E21	1	0	1	1	1	1	1	1	1	1	1	10
E02	1	1	0	0,5	1	1	1	0,5	1	1	1	9
E03	1	0	1	1	0	1	1	1	1	1	1	9
E06	1	0	1	0,5	0,5	1	1	1	1	1	1	9
E11	1	1	0	1	1	1	1	0,5	0,5	1	1	9
E17	1	0	1	1	0	1	1	1	1	1	1	9
E01	1	0	1	1	0	1	0,5	1	1	1	1	8,5
E07	1	0	1	1	1	1	1	0	0	1	1	8
E09	0,5	0	1	1	0	1	1	1	0,5	1	1	8
E10	1	0	0	1	0	1	1	1	1	1	1	8
E12	0,5	0	0	1	0,5	1	1	1	1	1	1	8
E14	0	0	1	1	0	1	1	1	1	1	1	8
E20	1	0	0	0	1	1	1	1	1	1	1	8
E13	0	0	1	0	0,5	1	1	1	1	1	1	7,5
E19	0,5	0	0	1	0	1	1	1	1	1	1	7,5
E05	0	0	1	0,5	0,5	0,5	0,5	1	1	1	1	7
E15	0	0	0	1	0	1	1	1	1	1	1	7

ID: Identificação do estudo.

APÊNDICE C - QUESTIONÁRIO PARA AVALIAÇÃO DA PRÁTICA DE DOCUMENTAÇÃO.

1 - A quantos anos você trabalha na instituição?

- 1-3 anos
- 4-6 anos
- 7-10 anos
- Mais de 10 anos

2 - Durante a manutenção do sistema, com que frequência (sendo 1 considera baixa e 5 considerada alta) você consulta cada uma das seguintes fontes de informação para completar uma tarefa?

- O próprio código fonte
- Documentação interna (ex: scripts de atendimento, documentos de projeto, manuais de usuário)
- Documentação externa (ex: documentação do caché)
- Conhecimento prévio (não necessita consultar informação)
- Atores externos (ex. outras diretorias, usuários)
- Anotações pessoais
- Documentos legais (ex: resoluções, portarias, leis)

3 - Com base a sua resposta da pergunta anterior, especifique a frequência (sendo 1 considera baixa e 5 considerada alta) que você consulta os tipos de documentação técnica abaixo:

- Documento de requisitos e especificação
- Documentos de design e modelagem (ex: diagramas UML)
- Documentos de testes
- Comentários de código
- Comentários do SVN
- Manuais de usuário
- Scripts de atendimento

4 - Durante a manutenção do sistema. Com que frequência (sendo 1 considera baixa e 5 considerada alta) você atualiza cada um destes tipos de documentação técnica?

- Documento de requisitos e especificação
- Documentos de design e modelagem (ex: diagramas UML)
- Documentos de testes
- Comentários de código
- Comentários do SVN
- Manuais de usuário
- Scripts de atendimento

5 - Quanta experiência você tem com a criação dos seguintes tipos de documentação técnica?

- Documento de requisitos e especificação
- Documentos de design e modelagem (ex: diagramas UML)
- Documentos de testes
- Comentários de código
- Comentários do SVN
- Manuais de usuário
- Scripts de atendimento

6 - Na sua opinião, quão útil é documentação existente (sendo 1 considera pouco e 5 considerada muito)?

7 - Qualifique a documentação técnica existente baseado nos seguintes fatores:

- Organização do conteúdo
- Acessibilidade
- Inclusão de modelos visuais
- Relevância do conteúdo
- Fácil leitura
- Exemplos de uso
- Completude
- Conteúdo atualizado
- Confiança nas informações

8 - Independente da pergunta anterior, na sua percepção, quais são os fatores mais importantes para aumentar a qualidade da documentação (por favor, escolha apenas os 3 maiores):

- Organização do conteúdo
- Acessibilidade
- Inclusão de modelos visuais
- Relevância do conteúdo
- Fácil leitura
- Exemplos de uso
- Completude
- Conteúdo atualizado
- Confiança nas informações