

Capítulo 2

As FGRMs no Contexto da Manutenção de Software

A tendência é que os sistemas de software evoluam para atender aos requisitos e alterações do ambiente no qual eles estejam inseridos. Em uma série de estudos, Lehman propôs leis sobre a evolução do software. A Lei da Mudança Contínua (Continuing Change) afirma que um software em uso deve mudar ou se tornará progressivamente menos útil [Lehman, 1980]. A Lei da Complexidade Crescente (Increasing complexity) afirma que as mudanças realizadas em um sistema torna a sua estrutura cada vez mais complexa. Neste contexto, recursos extras devem ser disponibilizados a fim de preservar e simplificar a estrutura do software [Lehman, 1980]. As leis de Lehman têm sido validadas, especialmente aquelas relacionadas ao tamanho e complexidade do software. Em um trabalho sobre o tema Yu & Mishra [Yu & Mishra, 2013] examinaram de forma empírica as Leis de Lehman em relação a evolução da qualidade do software. O estudo demonstrou que a qualidade de um produto de software declinará a menos que uma restruturação seja realizada.

Partindo da premissa de que mudanças em software são inevitáveis, torna-se importante uma disciplina com foco no gerenciamento e controle das alterações. Em geral, dentro do escopo da Engenharia de Software a tarefa fica a cargo da *Manutenção de Software*. Nas próximas seções discutiremos os conceitos básicos da Manutenção de Software que foram utilizados nesta dissertação e sua relação com as FGRMs, que é a sigla para Ferramentas de Gerenciamento de Requisições de Mudança que é o software utilizado para gerenciar uma Requisição de Mudança, o veículo de comunicação de uma falha ou melhoria entre as diferentes partes interessadas de um projeto de software.

2.1 A Manutenção de Software e a Requisição de Mudança

Nesta seção apresentamos terminologias que ajudam no entendimento do papel e finalidade da Manutenção de Software e a relação destes termos com o conceito de Requisição de Mudança. Iniciaremos com uma visão geral do processo de Manutenção de Software.

2.1.1 Visão Geral da Manutenção de Software

De uma maneira geral, podemos definir atividade de manter software como a totalidade das ações necessárias para fornecer suporte a um produto de software. No Padrão IEEE 1219 [ISO/IEEE, 1998] ela é definida como a modificação de um produto de software após a sua entrega com o objetivo de corrigir falhas, melhorar o desempenho ou outros atributos com a finalidade de adaptar o software às modificações ambientais.

Posteriormente a IEEE/EIA 12207 - Padrão para o Processo de Ciclo de Vida do Software [ISO/IEC/IEEE, 2008], retrata a manutenção como um dos principais processos no ciclo de vida do software. Em seu texto a disciplina é definida como a atividade de modificação do código e da documentação associada em decorrência de uma falha ou necessidade de melhoria no software [Society et al., 2014].

De maneira relacionada, *Manutenibilidade* é a propriedade de um sistema ou componente de software em relação ao grau de *facilidade* que ele pode ser corrigido, melhorado ou adaptado [IEEE, 1990]. A ISO/IEC 9126 - 01 [ISO/IEC, 2001] define a Manutenibilidade como um atributo de qualidade do processo de Manutenção.

É possível verificar que *manter* e *evoluir* são aspectos comuns das diferentes definições para Manutenção de Software. Embora exista o entendimento que os processos de manutenção e evolução possuem características distintas [Tripathy & Naik, 2014], não está nos objetivos desta dissertação discutir e apresentar as diferenças. Neste sentido, utilizamos ambos os termos de forma intercambiáveis.

2.1.2 O Processo de Manutenção de Software

O Processo de Manutenção de Software é o conjunto de atividades, métodos, práticas e transformações utilizadas para desenvolver ou manter um software e seus artefatos associados [Paulk et al., 1993]. Existe na literatura alguns modelos para o processo de manutenção de software, especialmente baseados em uma visão “tradicional”. Nesta perspectiva o desenvolvimento e a manutenção de software possuem uma separação

clara. Contudo, no momento do desenvolvimento desta dissertação, existia uma tendência de adoção das práticas propostas pelos agilistas na manutenção de software. Esta inclinação surge da demanda por serviços de manutenção de rápido retorno para o usuário.

2.1.2.1 Manutenção de Software Tradicional

Nesta seção apresentamos e discutimos alguns modelos para o processo de manutenção, que segundo o nosso entendimento, são os principais disponíveis na literatura. No contexto desta dissertação estes modelos são descritos como tradicionais. Em resumo, um processo de manutenção de software descreve as atividades e suas respectivas entradas e saídas. Alguns modelos são descritos nos padrões IEEE 1219 e ISO/IEC 14764. O processo especificado no padrão IEEE-1219 indica que as atividades de manutenção de software iniciem após a entrega do produto de software. O padrão também discute aspectos de planejamento da manutenção. As atividades que compõem o processo são apresentadas na Figura 2.1.

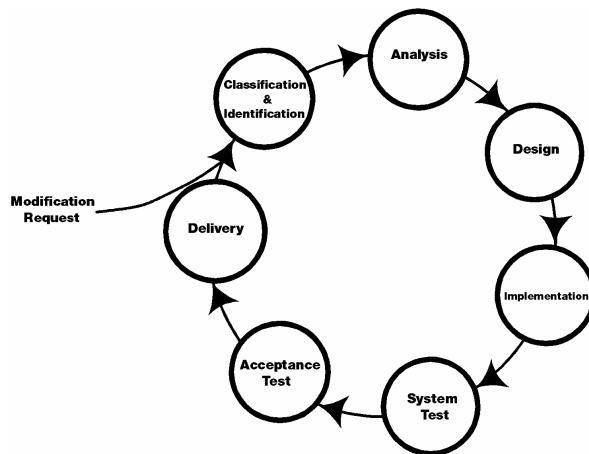


Figura 2.1: IEEE 1219 - Processo de Manutenção de Software

De maneira relacionada, na ISO/IEC 14764 as atividades que compõem o processo são similares aquelas propostas na IEEE- 1219, exceto pelo fato que elas são agregadas de uma forma diferente. O processo descrito na ISO/IEC 14764 são exibidas na Figura 2.2.

As atividades de manutenção propostas na ISO/IEC 14764 são detalhadas nas tarefas descritas a seguir:

- Implementação do Processo
- Análise e Modificação do Problema

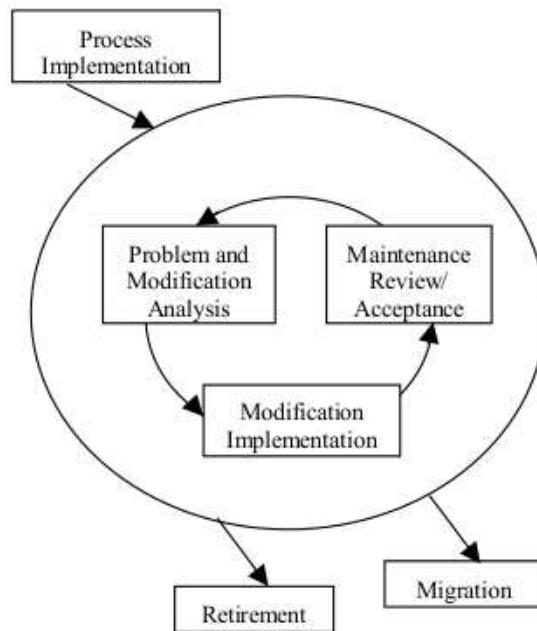


Figura 2.2: ISO/IEC 14764 Processo de Manutenção de Software

- Aceitação e Revisão da Manutenção
- Migração
- Aposentadoria do Software

É possível notar que algumas atividades realizadas durante a manutenção de software são similares a outras presentes no desenvolvimento de software, como por exemplo, análise de desempenho, codificação, teste e documentação. Outra atividade comum à manter e desenvolver software é o gerenciamento dos requisitos. Nas duas situações os profissionais responsáveis por controlar os requisitos devem atualizar a documentação por conta de alterações ocorridas no código fonte. Por outro lado, certas atividades estão vinculadas apenas ao contexto da manutenção de software. O Corpo de Conhecimento em Engenharia de Software [Abran et al., 2004] destaca algumas delas:

Compreensão do programa: atividades necessárias para obter um conhecimento geral do que um produto de software faz e como as partes funcionam em conjunto;

Transição: uma sequência controlada e coordenada de atividades onde o software é transferido progressivamente do desenvolvedor para o mantenedor;

Aceitação/rejeição de Requisições de Mudança¹: as modificações que ultrapassem determinado limiar de tamanho, esforço ou complexidade podem ser rejeitadas pelos mantenedores e redirecionadas para outro desenvolvedor;

Suporte ao usuário: uma função de suporte para o usuário final que pode resultar na priorização ou avaliação de esforço das Requisições de Mudança;

Análise de impacto: uma técnica para identificar os módulos que possivelmente são afetados por determinada mudança solicitada;

Contratos de Acordo de Nível de Serviço (Service Level Agreements - SLA): acordos contratuais que descrevem os serviços a serem realizados pela equipe de manutenção e os objetivos de qualidade do produto de software.

2.1.2.2 Manutenção de Software na Perspectiva dos Agilistas

Grande parte da literatura em Manutenção de Software trata de técnicas e metodologias tradicionais da Engenharia de Software. Todavia, verifica-se a tendência de que os departamentos dedicados à Manutenção de Software se mostrem interessados nas metodologias propostas pelos agilistas [Heeager & Rose, 2015]. No momento da elaboração desta dissertação boa parte dos textos em Engenharia de Software tratam desenvolvimento e manutenção como atividades com natureza distintas. Todavia, algumas “práticas ágeis” podem ser utilizadas em tarefas de manutenção tais como processo de trabalho iterativo, um maior envolvimento do cliente, a comunicação face a face e os testes frequentes.

A adoção das práticas dos agilistas na manutenção de software pode apresentar algumas dificuldades [Svensson & Host, 2005a]. Entre elas está a adequação das práticas da organização com as necessidades do time de desenvolvimento. Por outro lado, no trabalho de Choudhari & Suman [Choudhari & Suman, 2014] que propõe um modelo de processo para manutenção de software usando práticas da Programação Extrema (Extreme Programming - XP), apresenta como resultado: melhorias no aprendizado e produtividade da equipe por meio do aumento da moral, encorajamento e confiança entre os desenvolvedores.

2.1.3 Papéis na Manutenção de Software

As ações realizadas durante a manutenção de um software são desempenhadas por diferentes pessoas. Neste processo cada integrante da equipe de manutenção pode desempenhar um ou mais papéis. Os nomes e as atividades desenvolvidas por cada um pode variar de um projeto para outro, contudo, é possível determinar uma classificação que agregue um ponto comum entre os diferentes papéis. Nesta dissertação, utilizamos a classificação proposta por Polo e outros [Polo et al., 1999b] cujo objetivo é definir a estrutura da equipe de manutenção através da identificação das tarefas que cada

membro deve executar. O conjunto de papéis é o resultado da aplicação da metodologia MANTEMA [Polo et al., 1999a] em projetos de software bancários espanhóis em que o setor de manutenção foi terceirizada (outsourcing). Os autores reforçam que apesar da taxonomia ter sido criada em um contexto específico, ela pode ser utilizada para aplicação em outras situações.

Para esta dissertação removemos os papéis que segundo o nosso entendimento estão mais vinculados a um contexto de manutenção terceirizada (outsourcing). Além disso, dividimos o papel “time de manutenção” (maintenance team) em *Desenvolvedor* e *Analista de Qualidade* por entendermos que são papéis comuns a muito dos processos de manutenção existentes. Os papéis que compõem a taxonomia utilizadas durante o texto da dissertação estão descritos a seguir:

Usuário Afetado: Indivíduo que utiliza o sistema que correspondente à Requisição de Mudanças (RM) que será relatada. O defeito, a melhoria ou evolução no software, representada pela RM, estão relacionadas com os desejos e necessidades deste papel.

Reportador: Responsável por registrar a RM que pode ser qualquer pessoa envolvida no processo de Manutenção de Software. Neste sentido, as atividades relacionadas com o papel de Reportador podem estar vinculados com outras contidas nesta classificação. A Figura 2.3 apresenta esta situação através de um Diagrama de Caso de Uso, onde o *Reportador* pode ser um usuário do sistema ou um membro da equipe de manutenção.

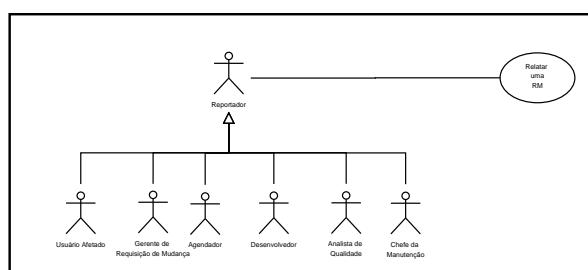


Figura 2.3: Diagrama de caso de uso do papel Reportador

Gerente de Requisição de Mudança (Maintenance-request manager): Responsável por decidir se uma RM será aceita ou rejeitada. Além disso, ele define qual tipo de manutenção deverá ser aplicada. Posteriormente cabe a este profissional encaminhar a RM para o Agente de Triagem.

Agente de Triagem (Scheduler) : Deve planejar a fila de RMs e atribuí-las para o desenvolver mais apto. A decisão pode considerar a carga de trabalho existente para cada desenvolvedor.

Desenvolvedor: Responsável por realizar as ações que irão solucionar a RM.

Analista de Qualidade: Tem por responsabilidade avaliar se uma RM solucionada por um Desenvolvedor afim de verificar se ela foi resolvida de forma correta e dentro dos padrões de qualidade do projeto.

Chefe da Manutenção (Head of Maintenance): Este papel é responsável por definir os padrões e procedimentos que compõem o processo de manutenção que será utilizado.

Apesar da classificação de papéis derivar de um contexto específico (setor bancário e empresas com a área de manutenção terceirizada), ela é capaz de acoplar com outros tipos modelos de processo como aquele proposto por Ihara e outros [Ihara et al., 2009a]. Naquele estudo foi criada uma representação de um processo de modificação de falhas (bugs) tomando como base as diversas situações que ela possui em uma FGRM no contexto de projetos de código aberto. O processo resultante é facilmente acoplável com a classificação utilizada em nosso estudo.

Cabe ressaltar que está fora do escopo deste estudo elaborar uma taxonomia dos papéis envolvidos na Manutenção de Software em função de supormos que isto corresponde a um esforço bem extenso. Nossa ação é identificar se existem papéis e quais são eles, sem com isso, envolver em uma consolidação definitiva.

2.1.4 Requisição de Mudança

2.1.4.1 Conceitos Básicos

Uma Requisição de Mudança (RM) é o veículo para registrar a informação sobre o defeito, evolução ou melhoria de um sistema [Tripathy & Naik, 2014]. De maneira geral, uma RM pode ser especializada como o *Pedido de Correção* de uma falha ou o *Pedido de Melhoria* que pode estar relacionado com o aprimoramento de funcionalidades ou com a melhoria da qualidade do sistema. Esta visão é apresentada na Figura 2.4. Alguns autores utilizam os termos *relato de defeito* ou *relato de melhoria* como sinônimos para a RM. Todavia, no escopo desta dissertação, o relato é um atributo da RM que representa o texto que descreve uma falha ou pedido de melhoria (vide Figura 2.5).

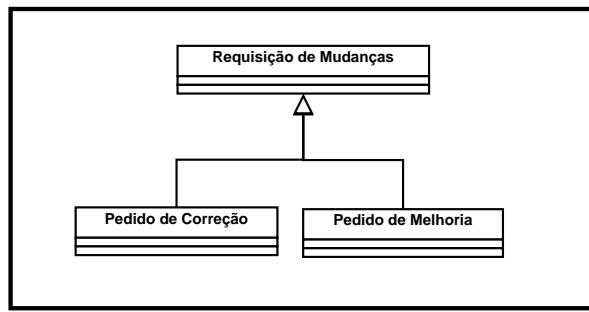


Figura 2.4: Modelo conceitual de uma Requisição de Mudanças

As principais características que compõem uma RM podem ser visualizadas no modelo exibido na Figura 2.5. Trata-se de uma adaptação do que foi proposto no trabalho de Singh & Chaturvedi [Singh & Chaturvedi, 2011] que descreve um processo genérico de como uma RM é relatada.

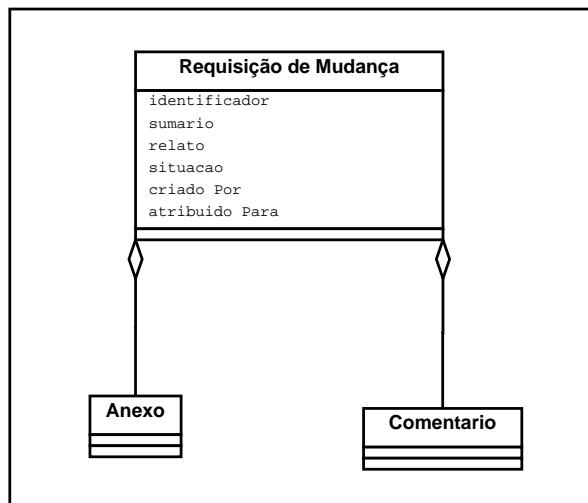


Figura 2.5: Informações que compõem uma RM

Os principais conceitos envolvidos no modelo estão detalhados a seguir:

Identificador Sequência de caracteres, geralmente numérica, que permite distinguir de maneira única uma RM.

Sumário Um título ou resumo da RM.

Relato Descrição detalhada da RM incluindo “o que”, “onde”, “por quê”, “como” e “quando” a situação relatada ocorreu. A mensagem que aparece durante a operação do sistema pode ser incluída, bem como a entrada inserida e/ou a saída esperada.

Situação A situação atual de uma RM. Representa os diversos estados que uma RM possui em seu ciclo de vida. Nesta dissertação discutimos brevemente o ciclo de vida de uma RM na Subseção 2.1.4.2.

Criado Por Nome da pessoa ou um identificador previamente registrado no sistema de quem criou a RM.

Atribuído Para A RM pode ser atribuída a uma pessoa específica caso ela seja capaz de resolvê-la, caso contrário, a RM será atribuída para alguém que possui o papel de definir o desenvolvedor mais apto para solucioná-la. Neste estudo, o membro de uma equipe de manutenção com esta função é denominado *Agente de Triagem*.

Anexo Refere-se a informação em formato diferente de texto e que pode ser incluída na RM. Por exemplo, casos de teste, capturas de tela, e cadeia de registros de ativação (stack trace).

Comentário Registra o histórico de discussões realizadas durante o processo de solução da RM².

Conforme pode ser observado na Figura 2.5 os atributos *Comentário* e *Anexo* foram modelados como uma agregação, dando aos mesmos um caráter multivalorado. Esta escolha foi intencional para salientar que uma RM pode conter diversos anexos ou comentários. No caso dos comentários esta característica é ainda mais relevante tendo em vista que eles são realizados durante o processo de solução de uma RM. A partir do conjunto de comentários é possível coletar informações relevantes para a manutenção do software e que podem ser utilizadas para solucionar futuras RMs.

Os atributos que compõem uma RM pode variar dependendo de fatores como a ferramenta utilizada para o gerenciamento, o projeto e a equipe de manutenção. Esses campos fornecem uma variedade de metadados descritivos tais como *importância*, *prioridade*, *gravidade*, *componente*, e *produto* [Zhang et al., 2016]. Em alguns casos a RM pode conter um campo de modo à relacioná-la com outra já existente na base de dados. Este tipo de vínculo é importante para minimizar problemas da gestão das RMs como por exemplo as duplicadas. Alguns dos problemas relacionados com a gestão das RMs estão descritos na Seção 2.1.4.3. A Figura 2.6 exibe um exemplo representativo de uma RM contendo os elementos básicos descritos no modelo proposto na Figura 2.5.

Em síntese, apesar das diferentes nomenclaturas existentes na literatura (demanda, bug, defeito, bilhete, tíquete, requisição de modificação, relato de problema)

²O conceito de solução bem como de outros relacionados ao ciclo de vida de uma RM estão descritos com maiores detalhes na Seção 2.1.4.2



Figura 2.6: Um exemplo de uma RM do Projeto Eclipse

uma Requisição de Mudança representa uma descrição, independente da estrutura, que visa gerar a manutenção ou evolução do software. A manutenção ou evolução estão relacionados com o reparo de uma falha ou com um desejo ou necessidade do usuário do software. Nesta dissertação procuramos ficar aderentes ao termo “Requisição de Mudança” e sua sigla *RM*.

2.1.4.2 Ciclo de Vida de uma Requisição de Mudança

Uma RM descreve os desejos e necessidades dos usuários de como um sistema deve operar. Quanto uma RM é relatada dois fatores devem ser levados em conta [Tripathy & Naik, 2014]:

- *Corretude da RM*: uma RM deve ser descrita de forma não ambígua tal que seja fácil revisá-la afim de determinar sua corretude. O “formulário”, que são os campos que devem ser preenchidos na RM, são a chave para efetiva interação entre a organização que desenvolver o software e os seus usuários. O formulário, neste sentido, documenta informações essenciais sobre mudanças no software, hardware e documentação.
- *Comunicação clara das RMs entre as partes interessadas*³: as RMs necessitam ser claramente comunicadas entre as parte interessadas, inclusive entre a equipe

³Na Seção 2.1.3 discutiremos em maior detalhe as diferentes partes interessadas no contexto da manutenção de software.

de manutenção. O resultado por avaliar de maneira distinta uma RM pode ser contra-produtivo: *(i)* a equipe que realiza mudanças no sistema e a equipe que executa testes podem ter visões contraditórias sobre a qualidade do software; *(ii)* O sistema alterado pode não atender às necessidades e desejos dos usuários finais.

No caminho entre sua criação e solução uma RM possui diferentes estágios. O ciclo de vida de uma RM pode ser ilustrado através do diagrama de estados da Figura 2.7. No diagrama uma RM inicia como *Submetida (Submit)* e vai sendo modificada até alcançar o estado *Fechada (Closed)* onde é considerada como solucionada. Neste caminho o conjunto de fatores que resultou na necessidade de relatar uma RM pode não mais existir. Neste caso, ela é alterada para o estado *Rejeitada (Decline)*.

Existe um aspecto importante do ciclo de vida de uma RM que não consta na discussão apresentada por Tripathy & Naik. Em teoria uma RM poderia ter novos estados como “Reaberta”, quando um usuário ou outro membro da equipe de manutenção entende que ela não foi solucionada, ou “Relacionada” quando uma nova RM é na realidade uma sucessora ou possui algum tipo de relação com outra RM anteriormente registrada.

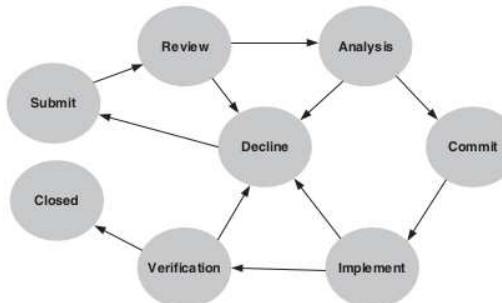


Figura 2.7: Diagrama de estados de uma RM. Extraído de [Tripathy & Naik, 2014]

A seguir apresentamos as características dos estados do ciclo de vida de uma RM como base na discussão realizada por Tripathy & Naik [Tripathy & Naik, 2014]. Para cada estado pode haver mais de um papel responsável pelas ações executadas. Também pode ocorrer que uma mesma pessoa desempenhe diferentes papéis neste processo. Uma discussão sobre os papéis utilizados no escopo desta dissertação pode ser encontrada na Seção 2.1.3.

Submetida (Submit). Este é o estado inicial de uma RM recentemente criada. Geralmente são os usuários do sistema a fonte primária das RM nesta situação. Com base no nível de prioridade da RM, ela é movida de *Submetida* para *Em Revisão*.

Normalmente cabe ao *Gerente de Requisição de Mudança* a responsabilidade da manipulação inicial das RMs. Neste instante ele se torna o “dono” da RM.

Em Revisão (Review). Normalmente, cabe ao *Gerente de Requisição de Mudanças* manipular as RMs no estado *Em Revisão* através das seguintes atividades:

- Verificar se a RM submetida recentemente é idêntica a outra já existente. Se a RM é identificada como duplicada o estado da mesma é alterado para *Rejeitada*. Neste caso, uma breve explicação e algum tipo de ligação para a original são inseridos nos comentários da RM.
- Aceitar o nível de prioridade atribuído para a RM ou alterá-lo.
- Determinar o nível de severidade da RM: normal ou crítico.

No caso das atividades descritas anteriormente não possam ser realizadas, a RM é movida para o estado em *Em análise*.

Em Análise (Analysis). Neste estágio uma análise de impacto é conduzida para entender o que foi solicitado na RM e estimar o tempo necessário para implementá-la. Caso não seja possível ou desejável atender a RM ela é alterada para a situação *Rejeitada*. Caso contrário a RM é movida para estado *Compromissada (Commit)*. No estado *Em Análise* o “dono” da RM é denominado *Agente de Triagem*.

Compromissada (Commit). A RM no estado *Compromissada* não foi atendida mas se encontra no planejamento do projeto de modo a estarem disponíveis em uma próxima versão do produto. Por se um estado relacionado ao gerenciamento da manutenção, as RMs nesta situação estão à cargo do *Gerente de Requisição de Mudança*. Algumas RMs podem ser incluídas em futuras versões do sistema após acordo com as demais partes interessadas.

Em Implementação (Implement). No estágio de *Em Implementação* diferentes cenários podem ocorrer:

- A RM pode ser rejeitada caso sua implementação não seja factível.
- Caso a RM seja possível de implementar, os desenvolvedores realizam a codificação e os testes. Após o desenvolvimento ser finalizado a RM é movida de *Em Implementação* para *Em verificação*.

Em Verificação (Verification). No estado de verificação as atividades são controladas pela equipe de testes. A verificação de uma RM pode ser realizada pelo seguintes métodos: demonstração, análise, inspeção ou teste. No primeiro caso, o software é executado com um conjunto de testes. A inspeção significa revisar o código em busca de defeitos. No caso da análise, o processo consiste em demonstrar que o sistema está em operação.

Fechada (Closed). Após a verificação de que a RM foi atendida, ela é movida de *Em verificação* para *Fechada*. Esta ação é realizada pelo *Analista de Qualidade* que é o “proprietário” da RM durante o estado de *Em Verificação*. Nesta dissertação, quando referenciamos ao último estágio do ciclo de vida da RM utilizaremos o termo “Solução da RM” para representar a situação onde a falha relatada ou a melhoria solicitada é entendida, por algum membro das partes interessadas, como atendida.

Rejeitada (Decline). Uma RM pode ser rejeitada caso ela deixar de produzir relevante impacto no sistema, não seja possível tecnicamente realizar o que foi solicitado na RM e a equipe de qualidade conclui que as mudanças no software para atender à RM não podem ser satisfatoriamente verificadas.

O modelo de ciclo de vida discutido por Tripathy & Naik possui foco em organizações que possuem uma área exclusivamente dedicada à manutenção de software. Em outros contextos, como por exemplo em projetos de código aberto, o processo de modificação dos estados de uma RM pode ser diferente. No trabalho de Ihara e outros [Ihara et al., 2009b] foi conduzido um estudo de caso nos projetos Firefox e Apache e um dos resultados foi um diagrama de estados que representa o processo de modificação de uma RM. Este diagrama é apresentado na Figura 2.8.

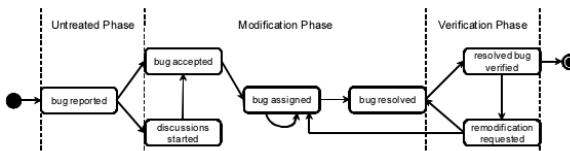


Figura 2.8: Um processo de modificação de uma RM utilizando uma FGRM. Extraído de [Ihara et al., 2009b]

Os autores ponderam que apesar da maneira que uma RM é modificada pode alterar em diferentes projetos, o diagrama é capaz de representar de maneira geral o processo de transição de uma RM. O processo é composto de três diferentes fases: *não tratada (untreated)*, *modificação (modification)* e *verificação (verification)*.

A fase *não tratada* foca em um subprocesso onde as RMs são relatadas, todavia, não foram aceitas ou atribuídas à algum membro da equipe. A fase de *modificação* é um subprocesso onde as RMs são efetivamente modificadas. Nesta fase uma RM é aceita e posteriormente atribuída a algum desenvolvedor. Caso o pedido da RM não possa ser atendido ela é *rejeitada*. A fase de *verificação* é o subprocesso onde membros com a responsabilidade de garantia da qualidade verificam se as RMs modificadas foram efetivamente solucionadas. Caso uma RM modificada por um desenvolvedor não seja verificada, ela poderia não ser reconhecida como fechada (*closed*).

É possível relacionar o modelo descrito por Tripathy & Naik [Tripathy & Naik, 2014] e aquele proposto por Ihara e outros [Ihara et al., 2009b]. Em ambos é possível identificar fases onde uma RM é relatada, analisada e verificada. Além disso, os modelos discriminam situações onde o pedido descrito na RM não é capaz de ser realizado. Nesta dissertação utilizamos de forma geral o modelo proposto por Ihara e outros. Nos casos em que houver necessidade de um maior detalhamento a discussão tomará como base o modelo de Tripathy & Naik.

2.1.4.3 Problemas e Desafios do Gerenciamento das RMs

A gestão das RMs é um desafio em projetos de software de diferentes tamanhos. A literatura discute e apresenta os problemas sobre o gerenciamento das RMs. A seguir discutimos os que do nosso ponto de vista são os mais relevantes.

Localização do Problema: A tarefa de encontrar a origem de uma falha de software é complexa e consome muito tempo. O estudo de Lúcia e outros afirma que na faixa de 84 a 93% de problemas em software afetam entre 1 e 2 arquivos de código-fonte [Thung et al., 2012a]. Apesar da pequena quantidade de arquivos afetados, identificar em quais deles o problema reside (*buggy files*) é uma tarefa árdua [Thung et al., 2014c].

Neste contexto, pesquisadores vêm propondo abordagens baseadas em Recuperação da Informação para localizar o arquivo que contém uma falha com base no texto do relato de uma RM. Nesse tipo de abordagem existe a tentativa de encontrar um elo entre o texto do relato e um conjunto de arquivos que podem estar relacionados com a solução do problema [Wong et al., 2014].

Dificuldade na Visualização das Informações das RMs: Uma tomada de decisão deve ser subsidiada por informações corretas. Este fato não é diferente na manutenção e desenvolvimento de software. Pouco se sabe sobre o comportamento

evolutivo, o tempo de vida, distribuição e estabilidade dos problemas reportados nas FGRM [Hora et al., 2012]. Este problema é reforçado pela forma como as FGRMs armazenam os dados das RMs. Em geral, esses exibem informações sobre as RMs de forma textual, o que não é apenas complicado para navegar, mas também dificulta a compreensão das complexas peças de informação que giram em torno dos problemas de software [Dal Sasso & Lanza, 2014]. Por esta razão estudos estão sendo realizados de modo a propor novas formas de visualização da informação contida em uma RM [Takama & Kurosawa, 2013, Hora et al., 2012].

Baixa Qualidade do Relato: Durante o processo de solução de uma RM a reprodução manual das falhas reportadas é demorada e tediosa. Os mantenedores tentam reproduzir problemas usando a informação contida nas RMs que muitas das vezes está incompleta [White et al., 2015a]. Em algumas situações, para obter os dados que necessita, o desenvolvedor deve registrar um comentário para que o responsável do relato inclua as informações necessárias [Zimmermann et al., 2009b]. A melhoria da qualidade do relato pode implicar na redução do custo do processo de garantia de qualidade bem como aumentar a confiabilidade do software com a redução gradativa de bugs [Tu & Zhang, 2014].

Identificação de RM Duplicadas: O processo de identificação de RMs Duplicadas consiste em avaliar se determinado relato já foi realizado em outro momento. Quando uma RM é identificada como duplicada ela deveria ser relacionada com a sua “cópia”. Uma delas é definida como RM Mestre e as demais RMs Filhas. Geralmente a Mestre é aquela que foi primeiramente incluída no repositório de erros. Alguns estudos revelam que entre 10% e 30% das RMs podem ser classificadas como duplicadas [Anvik et al., 2005, Cavalcanti et al., 2013, Runeson et al., 2007]. Por conta do grande número de RMs repetidas uma das soluções é o *Agente de Triagem* analisá-las manualmente com objetivo de evitar que elas cheguem as desenvolvedores [Anvik et al., 2005]. Em alguns casos esta solução não é viável. O processo de identificação de RMs duplicadas requer: (i) um prévio conhecimento do conjunto de relatos existentes anteriormente no projeto; (ii) a busca manual em toda base de dados da FGRM [Banerjee et al., 2012, Lerch & Mezini, 2013, Hindle et al., 2016]. Ambas as estratégias consomem tempo e não garantem que falsos positivos possam ocorrer [Kaushik & Tahvildari, 2012]. Os falsos positivos podem ainda acarretar na desconsideração de problemas relevantes.

Atribuição (Triagem) de RM: A atividade de atribuição de RM, cuja principal atividade é conhecida como *triagem*, tem como objetivo encontrar o desenvolvedor mais capacitado para manipular uma dada RM [Cavalcanti et al., 2014]. Existe a premissa de que a escolha do desenvolvedor apropriado é crucial para obter em menor tempo a solução da RM [Di Lucca et al., 2002]. Estudos também discutem que o processo de atribuição deve considerar fatores tais como a carga de trabalho do desenvolvedor, a prioridade da RM, dentre outros [Aljarah et al., 2011].

Classificação da RM: Independentemente do tipo e tamanho de um projeto é importante determinar qual tipo de manutenção deverá ser realizada. Geralmente este tipo de classificação é feita com base no texto que corresponde ao relato da RM. A diversidade de categorias pode tornar complexa a tarefa pelo fato de que em muitos casos não é fácil determinar os limites entre os tipos [Antoniol et al., 2008]. Por exemplo, a uma classificação incorreta de um defeito que na verdade trata-se de uma melhoria pode acarretar em atrasos no projeto [Cavalcanti et al., 2014].

Estimativa de Esforço da RM: A gestão de custo e esforço de um projeto de manutenção de software passa pelo controle do esforço necessário para solucionar suas RMs. Os estudos que discutem o esforço para solução de uma RM utilizam três formas de estimativa [Cavalcanti et al., 2014]: determinar o tempo para solucionar novas RMs; definir os artefatos que são impactados por determinada RM (Análise de Impacto); prever o número de novas RMs que poderão fazer parte do projeto. A literatura sobre análise de impacto é bastante abrangente e pode envolver o estudo de artefatos tais como documentos de requisitos e arquiteturas de softwares, código fonte, registros (logs) de teste e assim por diante [Cavalcanti et al., 2014]. Apesar da inerente imprecisão deste tipo de trabalho é importante salientar que estimar o esforço de uma RM é importante para o gerenciamento do projeto porque ajuda alocar recurso de forma mais eficiente [Bhattacharya & Neamtiu, 2011] e melhorar a previsão do custo necessário para o lançamento de futuras versões do sistema [Vijayakumar & Bhuvaneswari, 2014].

Recomendação de RMs: Em alguns projetos, um membro experiente da equipe, geralmente ensina os recém-chegados o que eles precisam fazer para solucionar uma RM. Todavia, alocar um membro experiente de uma equipe por um longo tempo para ensinar um novo membro da equipe nem sempre é possível ou desejável. A premissa é que o mentor poderia ser mais útil fazendo tarefas mais importantes [Malheiros et al., 2012]. Por exemplo, quando um novo desenvolvedor entra na equipe seria interessante que ele resolvesse as RMs que tivessem um menor nível de dificuldade. Posteriormente,

quando o desenvolvedor ganhasse experiência, poderia aumentar o grau de dificuldade relacionado à RM que ele deve tratar. Este tipo de processo ocorre com certa frequência em projetos de código aberto, onde a contribuição de desenvolvedores fora do projeto é fundamental. No entanto, encontrar um defeito apropriado ao nível de conhecimento do desenvolvedor, bem como a correção apropriada para o mesmo requer uma boa compreensão do projeto [Wang & Sarma, 2011].

Para facilitar a inclusão de novos desenvolvedores alguns estudos vêm se dedicando em desenvolver sistemas de recomendação de RMs [Malheiros et al., 2012, Wang & Sarma, 2011]. Estes sistemas podem ajudar o recém-chegado a solucionar uma RM mediante a apresentação de outras de código fonte potencialmente relevante que o ajudará na solução da RM do qual ficou responsável [Malheiros et al., 2012].

2.2 As Ferramentas de Gerenciamento de Requisições de Mudança (FGRM)

Dentro da disciplina de Gerenciamento da Configuração do Software a atividade de controle de configuração é responsável por gerenciar mudanças ocorridas durante o ciclo de vida de um produto de software [Tripathy & Naik, 2014]. Entre as atividades deste processo estão determinar quais alterações serão feitas, definir o papel responsável por autorizar certos tipos de mudança e aprovar desvios relativos aos requisitos iniciais do projeto [Abran et al., 2004]. De uma forma mais ou menos estruturada tais ações ocorrem em diferentes tipos de projeto de software, seja manutenção com características tradicionais (vide Seção 2.1.2.1) ou ainda naqueles que utilizam os métodos propostos pelos agilistas.

Por conta do volume das RMs e os diversos desafios relacionado com sua gestão é necessária a utilização de ferramentas com o objetivo de gerenciá-las. Esse controle é geralmente realizado por Sistemas de Controle de Demandas (SCD)- Issue Tracking Systems, que auxiliam os desenvolvedores na correção de forma individual ou colaborativa de defeitos (bugs), no desenvolvimento de novas funcionalidades, dentre outras tarefas relativas à manutenção de software. Não existe na literatura uma nomenclatura padrão para este tipo de ferramenta. Em alguns estudos é possível verificar nomes tais como Sistema de Controle de Defeito- Bug Tracking Systems, Sistema de Gerenciamento da Requisição- Request Management System, Sistemas de Controle de Demandas (SCD)- Issue Tracking Systems e outros diversos nomes. Todavia, de modo geral, o termo se refere às ferramentas utilizadas pelas organizações para *gerenciar as Requisições de Mudança*. Estas ferramentas podem ainda ser utilizadas por gestores,

analistas de qualidade e usuários finais para atividades tais como gerenciamento de projetos, comunicação, discussão e revisão de código. Neste trabalho utilizaremos o termo **Ferramentas de Gerenciamento de Requisições de Mudança** (FGRM) ao nos referir a este tipo de softwares.

As RMs são controladas por uma FGRM na forma de um fluxo de trabalho de modo a identificar, descrever e controlar a situação de cada RM. Em geral, os objetivos de um projeto adotar uma FGRM para gerenciar suas RMs são os seguintes [Tripathy & Naik, 2014]:

- Disponibilizar um espaço comum para a comunicação entre as partes interessadas.
- Identificar de forma única e controlar a situação de cada RM. Esta característica simplifica o processo de relatar uma RM e fornece um melhor controle sobre as mudanças.
- Manter uma base de dados sobre todas as mudanças sobre o sistema desenvolvido ou mantido pelo projeto. Esta informação pode ser utilizada para monitoramento e métricas de medição.

No momento em que este trabalho estava sendo desenvolvido, estudos discutem o fato que as FGRMs não apenas ajudam as organizações a gerenciar, atribuir, controlar, resolver e arquivar as RMs [Bertram et al., 2010]. Em alguns casos, este tipo de ferramenta se tornou o ponto focal para comunicação e coordenação para diversas partes interessadas, dentro e além da equipe de manutenção. As FGRMs servem como um repositório central para monitorar o progresso da RM, solicitar informações adicionais da pessoa responsável por redigir a requisição e o ponto de discussão para potenciais soluções de um defeito (bug) [Zimmermann et al., 2009a].

Em projetos de código aberto, as FGRMs são um importante espaço onde a equipe de desenvolvimento interage com a comunidade. Como consequência é possível observar o fenômeno da participação dos usuários no processo de solução da RM: eles não apenas submetem a RM, mas também participam da discussão de como resolvê-la. Desta forma, o usuário final ajuda nas decisões sobre a direção futura do produto de software [Breu et al., 2010b].

2.2.1 Modelo Conceitual do Contexto das FGRMs

As FGRMs vêm sendo utilizadas por diversos projetos com características próprias. Neste sentido, este tipo de software deveria oferecer diferentes funcionalidades a fim

de atender esta demanda. Apesar da variedade de ferramentas disponíveis⁴ é possível encontrar atributos comuns que permitem a compilação de um modelo conceitual.

Nós construímos um modelo conceitual com base na literatura da área, em especial nos trabalhos de [Cavalcanti et al., 2014, Singh & Chaturvedi, 2011, Kshirsagar & Chandre, 2015]. Nós sintetizamos os dados através da identificação de temas recorrentes da definição de FGRMs encontradas nos artigos. Foram encontrados quatro principais conceitos que estão retratados na Figura 2.9 como um diagrama baseado na UML. Esta figura foi derivada dos estudos primários e consiste em uma generalização dos elementos utilizados com frequência nos artigos. Os conceitos envolvidos no modelo estão descritos a seguir.

Projeto: Projeto de software para o qual a FGRM visa suportar. Ele é composto pelos atributos *Componentes de Software*, *Artefatos* e *Contexto de Desenvolvimento*.

- *Componente de Software* representa um ou mais módulo que fazem parte do sistema que a FGRM suporta.
- *Artefatos* são os objetos utilizados ou produzidos no desenvolvimento do software tais como código fonte, documentação, casos de teste e etc.
- *Contexto de Desenvolvimento* representa os atributos que interferem no processo de desenvolvimento e manutenção de software. Nele está contido o processo de desenvolvimento (por exemplo métodos ágeis, cascata, iterativo e etc), as ferramentas utilizadas (compiladores, ferramentas debug e build) e outros.

Repositório de RM: Trata-se da base de dados onde as RMs são armazenadas e gerenciadas. Cada item nesta base é uma RM com as características discutidas na Seção 2.1.4.

Repositório de Usuários Representa a base de dados de usuários da FGRM. Nele são gerenciados os dados das pessoas envolvidas no projeto e de seus respectivos direitos de acesso às informações das RMs. Neste caso, esta base inclui tanto a equipe de manutenção quanto as demais partes interessadas.

Fluxo de Trabalho: O *Fluxo de Trabalho* representa o conjunto de regras que gerenciam o processo de resolução de uma RM. É a partir dele que são definidos os diferentes *estados* que uma RM pode assumir desde quando ela é redigida até o momento em que se define que foi solucionada. Este processo é realizado

⁴https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems

pelas *Pessoas* envolvidas no *Projeto* através dos diferentes *Papéis* desempenhados e suas respectivas *Atividades*. Uma discussão mais aprofundada sobre os papéis desempenhados na manutenção de software está disponível na Subseção 2.1.3. De maneira relacionada, os diferentes estados de um ciclo de vida de uma RM estão descritos na Subseção 2.1.4.2.

A partir da Figura 2.9 é possível verificar que um *projeto* pode *definir* o seu *fluxo de trabalho*, como por exemplo resolvendo que uma RM só pode ser considerada Fechada (Closed) - vide Seção 2.1.4.2 - caso ela tenha sido avaliada por um Analista de Qualidade (vide Seção 2.1.3). A partir deste fluxo as RMs podem ser *atendidas* visando à sua resolução o que é feito por uma *pessoa* devidamente registrada no *repositório de pessoas* e com permissão para realizar a ação necessária.

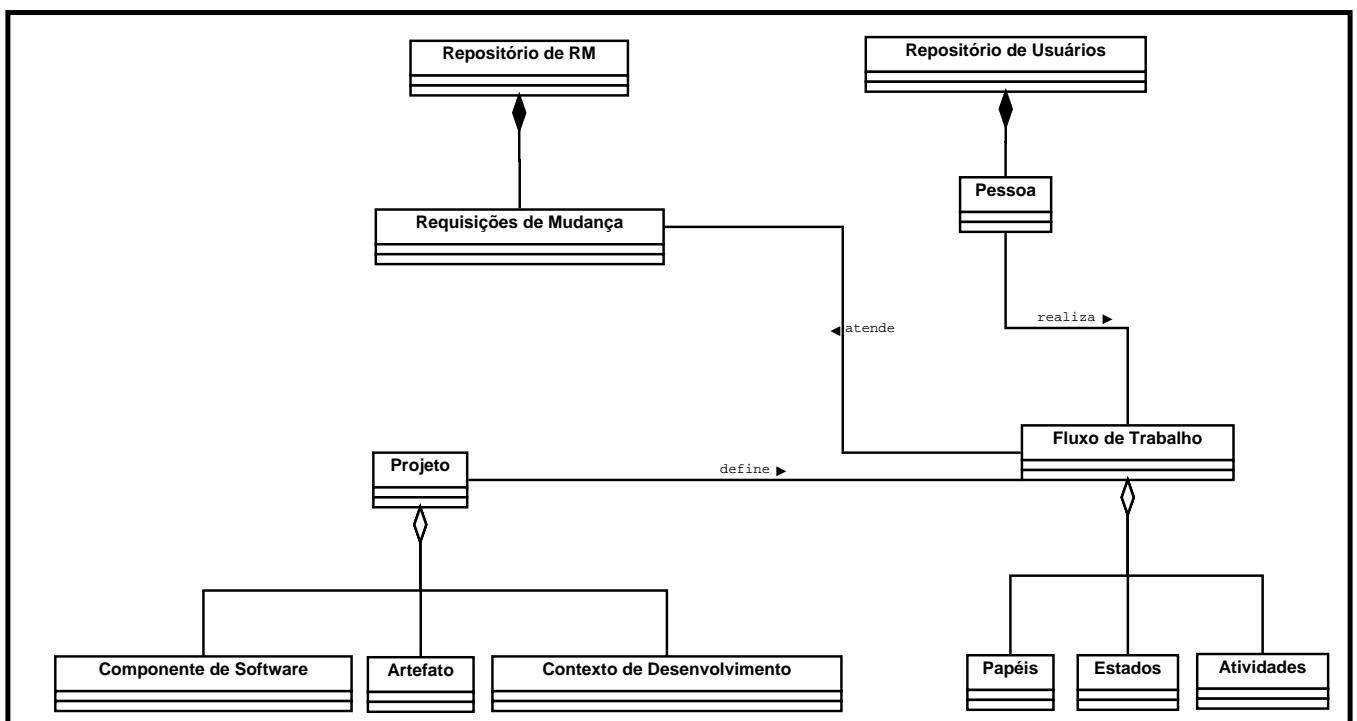


Figura 2.9: Modelo conceitual do contexto de uma FGRM

Conforme exposto, as FGRM desempenham um papel que vai além de gerenciar as Requisições de Mudança. Neste sentido, é importante estudar este tipo de software em busca de como melhorá-lo de modo a atender as diversas necessidades dos seus usuários. Contudo, é importante avaliar as novas funcionalidades propostas na literatura ou ainda mesmo a melhoria das já existentes. Uma possível forma de melhoria é através do uso de extensões. Na próxima seção abordamos esta propriedade de algumas FGRMs que permitem a inclusão e modificação de funcionalidades e comportamentos da ferramenta segundo as necessidades do usuário.

2.2.2 Extensões de Funcionalidades em FGRM

Em determinados domínios de aplicação é interessante desenvolver produtos de software com uma arquitetura que permita o sistema se adaptar às mudanças em seus requisitos. Existe a possibilidade de incluir novas funcionalidades dentro das já existentes no software, todavia, verificamos que sistemas que permitem extensões apresentam os seguintes benefícios:

- Extensibilidade: o software pode ser dinamicamente estendido mediante a inclusão de novos módulos de código que correspondem à novas características;
- Desenvolvimento em Paralelo: Quando os componentes não possuem certas dependências eles podem ser desenvolvidos em paralelo por times diferentes;
- Simplicidade: uma extensão tipicamente tem uma única funcionalidade, desta forma permite um melhor foco para os desenvolvedores.

No escopo deste trabalho, uma extensão é um componente de software que adiciona uma característica ou comportamento específico para um programa de computador⁵. Cabe-nos ressaltar que a nossa definição de extensão inclui aquelas que não estão acopladas ao código de determinada FGRM. Por exemplo, a funcionalidade de atribuição de uma RM está presente na maior parte das FGRMs, todavia, segundo nossa definição, uma proposta de melhoria desta funcionalidade mediante uma atribuição automatizada, por exemplo, será analisada como extensão mesmo que ela não esteja efetivamente funcionando em alguma FGRM. Vamos analisar as extensões de funcionalidade de forma independente se ela é oferecida baseada nos mecanismos de extensão discutidos nesta seção.

Verificamos na literatura alguns estudos em que as soluções propostas já se tornaram extensões de determinadas FGRM. Como pode ser observado no Mapeamento Sistemático realizado no Capítulo 3, a implementação da proposta do estudo em extensão de ferramenta não é o padrão observado. Os softwares que utilizam módulos de extensão têm aspectos de desenvolvimento e de manutenção potencialmente distintos daqueles sem esta característica. Este trabalho de mestrado faz uma contribuição na direção de uma melhor compreensão deste contexto a partir da análise de aspectos específicos das FGRMs.

⁵[https://en.wikipedia.org/wiki/Plug-in_\(computing\)](https://en.wikipedia.org/wiki/Plug-in_(computing))

2.3 Um Estudo sobre as Funcionalidades das FGRMs

2.3.1 Visão Geral

Quando uma empresa ou projeto de software de código aberto decide adotar uma FGRM um possível desafio é encontrar aquela que melhor atenda suas necessidades. Um critério de seleção é o conjunto de funcionalidades oferecidas pelo software. Outras variáveis podem envolver o custo e o suporte pós-venda da ferramenta. De maneira relacionada, o pesquisador que estuda propostas de melhorias para as FGRMs pode estar interessado em analisar o conjunto de funções que permita caracterizar este tipo de software.

O número de FGRMs disponíveis quando esta dissertação foi escrita era bastante elevado. Em uma inspeção inicial, verificamos a existência de mais de 50 ferramentas fornecidas comercialmente ou em código aberto⁶. Apesar das diversas opções disponíveis, ao bem do nosso conhecimento, desconhecemos estudos que avaliem sistematicamente as funcionalidades oferecidas por este tipo de ferramenta. Entendemos que a partir de um conjunto compartilhado de funções e comportamentos seja possível caracterizar as FGRMs, ao mesmo tempo que possibilita avaliar a contribuição de novas funcionalidades propostas na literatura, conforme discutido no Capítulo 3. Com este objetivo realizamos um estudo exploratório para coletar as funcionalidades presentes nas FGRMs. Em um estudo exploratório a preocupação é analisar o objeto em sua configuração natural, deixando que as descobertas surjam da própria observação [Wohlin et al., 2012].

O estudo descrito nesta Seção consistiu na leitura da documentação disponível na Internet de algumas FGRMs de modo a sistematizar as funcionalidades oferecidas por cada ferramenta. As funções foram coletadas e organizadas utilizando a técnica de Cartões de Classificação (Sorting Cards) [Zimmermann et al., 2009b, Rugg & McGeorge, 2005]. Devido ao alto volume de ferramentas disponíveis e ao esforço necessário para analisar a documentação de todas elas, optamos por realizar este estudo com um conjunto de 6 ferramentas que foram escolhidas com a ajuda de profissionais envolvidos em manutenção de software. A opinião dos profissionais foi coletada através de um levantamento por questionário (survey) onde eles selecionavam as FGRM que julgavam mais representativas dentre uma lista previamente definida. A representatividade neste contexto não está no número de projetos que utiliza de-

⁶https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems

terminada ferramenta, mas pelas características que o software possui e que o torna diferenciável dentro do seu domínio.

2.3.2 Objetivo do Estudo

O objetivo deste estudo é apresentar e discutir as principais funcionalidades das FGRMs que dão suporte ao desenvolvimento e manutenção de software. O pronto de partida é o conjunto de sistemas escolhidos por meio de um levantamento (survey). Acreditamos que o resultado permitirá uma melhor compreensão deste tipo de software tomando como base o conjunto de funções que eles oferecem aos seus usuários. Em um segundo momento também é possível propor novas funcionalidades ou melhorias das existentes tendo em vista a possibilidade de determinar o conjunto mínimo de comportamentos deste tipo de ferramenta.

2.3.3 Metodologia

Para determinarmos o conjunto de funcionalidades das FGRMs realizamos um estudo exploratório dividido nas três etapas listadas a seguir. O resultado obtido em cada etapa foi utilizado para subsidiar as atividades da etapa subsequente. O início de uma nova fase do trabalho era precedido de uma avaliação geral com o objetivo de verificar possíveis inconsistências e avaliação das lições aprendidas.

- (i) Seleção das Ferramentas
- (ii) Inspeção da Documentação
- (iii) Agrupamento das Funcionalidades

2.3.3.1 Seleção das Ferramentas

A primeira etapa consistiu da definição das ferramentas que seriam utilizadas no estudo. A partir de uma pesquisa na Internet obtivemos um conjunto inicial de 50 ferramentas⁷. Devido ao esforço necessário e a dificuldade de realizar a análise em cada uma optamos por escolher um subconjunto de sistemas que fossem mais representativos, tomando como base a opinião de desenvolvedores de código aberto e código proprietário, que tenham utilizado alguma FGRM. Segundo o nosso entendimento não houve a necessidade de analisar todos as ferramentas disponíveis. Um conjunto mínimo, avaliado como representativo pelos participantes do levantamento, poderia

⁷https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems

fornecer as informações necessárias para subsidiar este estudo. A representatividade no escopo do levantamento corresponde a opinião do profissional sobre notoriedade que a ferramenta possui dentro do seu domínio de aplicação em comparação com as demais que lhe foram apresentadas ou outras do qual o profissional tenha prévio conhecimento.

2.3.3.2 Desenho do Levantamento por Questionário

Para coletar a opinião dos profissionais utilizamos um formulário estruturado em duas partes: a formação de base do participante (background) e a avaliação das ferramentas. Na primeira, estávamos interessados em conhecer as características do respondente. Esta informação é relevante para analisar de forma separada os dois grupos de profissionais em que o questionário foi replicado. Na segunda, apresentamos as ferramentas e solicitamos aos participantes que avaliassem a relevância de cada uma delas através de uma escala do tipo Likert [Robbins & Heiberger, 2011]. Foi disponibilizado aos participantes um campo de texto livre em que era possível registrar outras FGRMs que ele entenda relevante, mas que não estava na lista que lhe foi apresentada. Para estas ferramentas que foram citadas de forma espontânea foi atribuído um grau de importância igual “Muito relevante” conforme Tabela 2.1.

Antes da aplicação o questionário foi validado em um processo de três etapas. Na primeira parte foi solicitado a avaliação por dois pesquisadores experientes da área de Engenharia de Software. Após as alterações uma nova versão do formulário foi enviada para dois profissionais que trabalham com manutenção de software. O critério utilizado para seleção dos profissionais foi o tempo de experiência com desenvolvimento e manutenção de software, que era em média de 10 anos. O formulário foi modificado com as sugestões dos profissionais e isso finaliza a segunda etapa de validação. A última etapa consistiu na realização de um piloto com dez profissionais que trabalham em um setor manutenção de software de uma empresa pública de informática. Trata-se de uma amostra de conveniência devida a nossa facilidade de acesso a estes desenvolvedores. Os profissionais tiveram que preencher o questionário, contudo, foram adicionadas questões em que era possível inserir sugestões de melhoria. Como o público-alvo do questionário poderia incluir desenvolvedores de diferentes nacionalidades foi criada uma versão em língua inglesa.

No caso deste levantamento por questionário, a população é o conjunto de profissionais que trabalham com desenvolvimento e manutenção de software e que tenham uma razoável experiência de uso com as FGRMs. A caracterização e estratificação desta população não é simples. Neste sentido, visando minimizar possíveis enviesamentos, replicamos o questionário em dois grupos:

Grupo 01: Profissionais que participam de fóruns e discussões sobre desenvolvimento e manutenção de software na rede social Stack Overflow.

Grupo 02: Profissionais relacionados a grupos que contribuem em projetos de código aberto.

2.3.3.3 Critérios de Seleção

Antes da seleção as FGRMs foram categorizados como *"ferramentas"* e *"serviços da internet"*. Para incluir determinado software em um dos grupos utilizamos a respectiva documentação do software. A primeira categoria representa os softwares que são capazes de serem implantados na infraestrutura do seu cliente e permite algum grau de personalização de pelo menos um dos componentes, como por exemplo, o banco de dados utilizado. No segunda estão os software que oferecem a gerência das RMs mediante uma arquitetura do tipo Software como Serviço (Software as Service) [Fox et al., 2013], onde certos tipos de alterações no comportamento do software são mais restritas. Acreditamos que ao escolher ferramentas dos dois tipos poderíamos cobrir grande parte do domínio de aplicação das FGRMs. Optamos por escolher *03 ferramentas* de cada categoria.

O grupo final de FGRMs foi selecionado pela frequência que cada grau de relevância apareceu no formulário, conforme a Tabela 2.1. Por exemplo, se uma ferramenta X teve a opção “Muito relevante” por três profissionais ele recebe uma pontuação igual a 15. Caso uma ferramenta não estivesse na lista, mas foi informada pelo participante de uma forma espontânea recebia uma pontuação igual a 5. Após o cálculo de pontuação de cada ferramenta, ordenamos do maior para o menor valor e escolhemos as três melhores posicionadas de cada categoria.

#	Grau de Relevância	Peso
1	Não conheço a ferramenta	1
2	Nada relevante	2
3	Pouco relevante	3
4	Pouco relevante	4
5	Muito relevante	5

Tabela 2.1: Graus de Relevância

2.3.3.4 Inspeção da Documentação

Nesta etapa do trabalho realizamos a leitura do material disponível na Internet para cada uma das ferramentas que foram selecionadas conforme critérios descritos na Subseção 2.3.3.3. Entre estes materiais utilizamos manuais do usuário e do desenvolvedor e notas de lançamento. Para cada uma das FGRMs optamos por estudar a última versão estável do software a fim de analisarmos o que há de mais novo disponível aos usuários. A documentação de algumas ferramentas, em especial aquelas que adotam uma arquitetura cliente/servidor e necessitam de um certo grau de administração, dividem as funcionalidades do software entre aquelas com foco no usuário final e administradores. Nestes casos, optamos por coletar as funcionalidades cujo foco seja o usuário da FGRM, tendo em vista que administradores deste tipo de software não estarem entre no público-alvo desta dissertação.

A Tabela 2.2 apresenta as ferramentas analisadas e o elo de ligação para a documentação utilizada neste estudo. Para aquelas ferramentas que apresentam documentação em mais de um idioma optamos por utilizar aquela escrita em inglês por entendermos que seja a mais atualizada.

Nome da Ferramenta	URL
Bugzilla	https://www.bugzilla.org/features/
Github Issue Tracking System	https://github.com/blog/411-github-issue-tracker
Github Issue Tracking System	https://github.com/features
Github Issue Tracking System	https://guides.github.com/features/issues/
Gitlab Issue Tracking System	http://docs.gitlab.com/ce/user/project/labels.html
Gitlab Issue Tracking System	https://about.gitlab.com/2016/08/22/announcing-the-gitlab-issue-board/
Gitlab Issue Tracking System	https://about.gitlab.com/solutions/issueboard/
Gitlab Issue Tracking System	https://docs.gitlab.com/ee/user/project/description_templates.html
Gitlab Issue Tracking System	https://docs.gitlab.com/ee/user/project/issues/automatic_issue_closing.html
Gitlab Issue Tracking System	https://docs.gitlab.com/ee/workflow/issue_weight.html
Gitlab Issue Tracking System	https://docs.gitlab.com/ee/workflow/milestones.html
Gitlab Issue Tracking System	https://docs.gitlab.com/ee/workflow/time_tracking.html
JIRA	https://br.atlassian.com/software/jira/features
MantisBT	https://www.mantisbt.org/wiki/doku.php:mantisbt:features
Redmine	http://www.redmine.org/projects/redmine/wiki/Features

Tabela 2.2: Documentações utilizadas no processo de coleta de dados.

Os dados obtidos da leitura do material disponíveis para cada ferramenta foram sistematizados por meio de técnica denominada *Cartões de Classificação - Sorting Cards*. Trata-se de uma técnica de elicitação de conhecimento de baixo custo e com foco no usuário. Ela é utilizada na área de arquitetura da informação para criar modelos mentais e derivar taxonomias a partir dos dados utilizados [Just et al., 2008]. Ela envolve a categorização de um conjunto de cartões em grupos distintos de acordo com um critério previamente definido [McGee & Greer, 2009]. O estudo de Maiden e outros [Maiden & Rugg, 1996] sugere que a técnica de Cartões de Classificação é uma

das mais úteis para aquisição de conhecimento de dados, em contraste ao conhecimento de comportamento ou de processo.

Existem três principais fases dentro do processo de Classificação dos Cartões: (*i*) preparação, no qual participantes ou o conteúdo dos cartões são selecionados; (*ii*) execução, onde os cartões são organizados em grupo significativos com um título que os descreve; e por fim, (*iii*) análise, no qual os cartões são sistematizados para formar hierarquias mais abstratas que são usadas para deduzir resultados. No processo tradicional de Cartões Ordenados cada declaração realizada por um participante resulta na criação de exatamente um único cartão [Just et al., 2008]. Contudo, no nosso caso, foi realizada a divisão da documentação da ferramenta por cada funcionalidade encontrada. Neste sentido, cada funcionalidade obtida mediante a inspeção da documentação foi mapeada em único cartão.

Os cartões foram organizados de modo que continham o nome e a versão da ferramenta analisada; a URL da documentação utilizada; o nome da funcionalidade coletada, que consiste de uma descrição breve conforme existente na documentação; descrição detalhada da funcionalidade, cujo objetivo é facilitar o processo de agrupamento que será descrito na próxima seção. Nas Figura 2.10 e 2.11 é possível visualizar, respectivamente, a documentação de uma funcionalidade da FGRM Bugzilla e o cartão que foi gerado para a mesma.

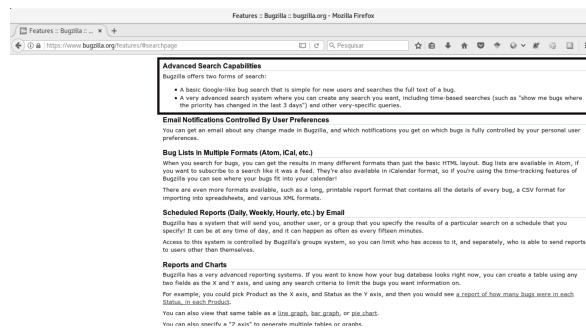


Figura 2.10: Exemplo de documentação de uma funcionalidade da FGRM Bugzilla

2.3.3.5 Agrupamento das Funcionalidades

Esta etapa tem por objetivo agrupar as funcionalidades que aparecem com nomenclatura distintas em diferentes ferramentas, mas que apresentam o mesmo significado. Cabe ressaltar que o agrupamento de algumas funcionalidades pode depender de uma análise subjetiva do responsável pela atividade. Neste sentido, a fim de evitar algum tipo de viés o agrupamento foi realizado em duas etapas:

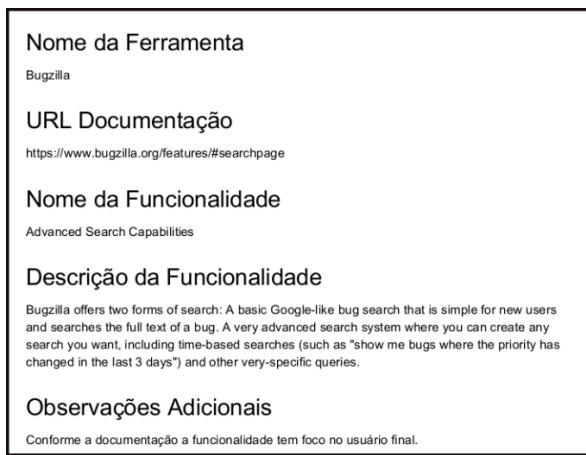


Figura 2.11: Exemplo de um cartão ordenado para uma funcionalidade da FGRM Bugzilla

Análise Individual Neste etapa o autor e um outro especialista realizam de forma separada os agrupamentos.

Analise Compartilhada Em um segundo momento tanto o autor quanto o especialista discutem as possíveis divergências até que um consenso seja obtido.

Após o processo de agrupamento foi possível realizar a categorização das funcionalidades das ferramentas. Os resultados do processo de agrupamento são apresentados e discutidos nas próximas seções.

2.3.4 Resultados

Nesta seção inicialmente mostramos os resultados inicialmente apresentando o perfil dos participantes e em seguida exibimos as categorias resultantes do processo de ordenamento dos cartões.

2.3.4.1 Perfil dos Participantes

Ao final do levantamento realizado com profissionais obtivemos um total de 52 respostas. Os profissionais que participaram são em sua maioria desenvolvedores conforme pode ser verificado na Figura 2.12.

O grupo de respondentes também incluem Engenheiros de Software, Gerentes de Equipe e Arquitetos de Software que, junto com os Desenvolvedores, representam mais de 80% do total. Com relação a experiência verificamos que a maior parte possui entre 3 e 10 anos (60%). Na segunda posição temos os participantes com 10 - 20 anos de experiência (17%). Com relação ao tamanho da equipe de que os participantes fazem

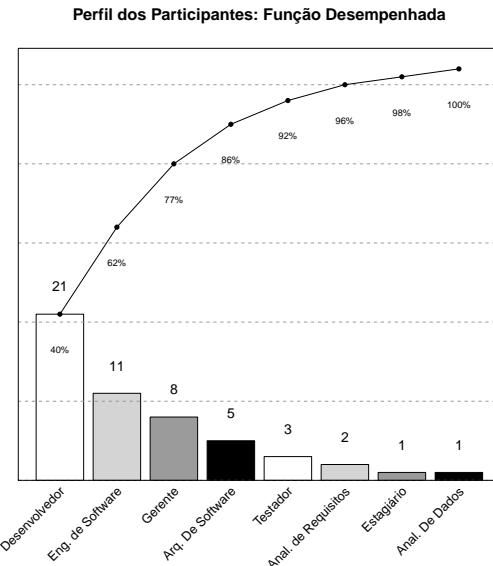


Figura 2.12: Funções desempenhadas pelos participantes

parte, verificamos uma prevalência de equipes de médio (mais do que 10 membro) e pequenas (2 a 5 membros) porte. Por sua vez, estas equipes estão predominantemente em empresas privadas de software, que representou 37 participantes. Com relação ao local de trabalho verificamos ainda que o segundo posto em número de participantes ficou para empresas que pertencem ao setor governamental, do qual tivemos 11 participantes. O restante é composto por um profissional que se dedica a projetos de software livre e um estudante.

Em geral, podemos caracterizar o participante típico como um desenvolvedor entre três e dez anos de experiência trabalhando em uma empresa privada de desenvolvimento de software que com uma equipe de aproximadamente dez membros. Este perfil profissional tem o conhecimento necessário para nos ajudar no processo de escolha das ferramentas.

2.3.4.2 Ferramentas Escolhidas

O processo de seleção resultou nas ferramentas apresentas na Tabela 2.3. Conforme pode ser observado foi escolhido três softwares de cada tipo (ferramenta e serviço da Internet). É importante perceber que as FGRMs *Github* e *Gitlab* não estavam na lista inicial, contudo, apareceram neste resultado final. Isso decorre de atribuirmos o maior peso para aquelas ferramentas que foram citadas pelos participantes de maneira espontânea.

Ferramenta	Classificação	Versão	URL
Bugzilla	Ferramenta	5.0.3	https://www.bugzilla.org
Mantis Bug Tracker	Ferramenta	1.3.2	https://www.mantisbt.org
Redmine	Ferramenta	3.3.1	http://www.redmine.org/
JIRA Software	Serviço	7.2.4	https://br.atlassian.com/software/jira
Github Issue System	Serviço	-	https://github.com/
Gitlab Issue Tracking System	Serviço	-	https://gitlab.com/

Tabela 2.3: Ferramentas utilizados no estudo

2.3.4.3 Espectro de Funcionalidades das FGRMs

Após a inspeção da documentação e validação dos dados obtivemos um total de 123 cartões. Nós sistematizamos os cartões manualmente tendo em vista que não existem ferramentas ou métodos capazes de automatizar o processo de construção deste tipo de hierarquia. Como o nosso objetivo é derivar tópicos a partir de um conjunto inicial de cartões, optamos por realizar um *ordenamento aberto*. Neste tipo de abordagem, os grupos são estabelecidos durante o processo de classificação dos cartões em oposição a outra forma de utilização da técnica onde a sistematização dos cartões ocorre com base em grupos pré-determinados. Ao final do processo compilamos os tópicos de modo a construir um espectro de funcionalidades para as FGRM que pode ser observado na Figura 2.13, no qual temos três dimensões de funcionalidades que são compostas por diferentes categorias de funcionalidades.

A figura foi construída com base nas categorias de funcionalidades exibidas na Tabela 2.4 em que é possível verificar ainda a frequência que cada uma das categorias apareceu no conjunto de cartões coletado.

Categoria de Funcionalidades	Frequência
Operações de CRUD	24
Visualização e Monitoramento de RMs	22
Segurança da Informação	17
Fluxo de Trabalho	15
Interfaces de Notificação	13
Extensão de Funcionalidades	8
Triagem de RMs	6
Gerenciamento de Artefatos	6
Integração com Sistemas de Controle de Versão	4
Gerenciamento da Informação	4
Internacionalização da Ferramenta	3
Auditoria	1

Tabela 2.4: Frequência de cada categoria de funcionalidade no conjunto de cartões obtidos.

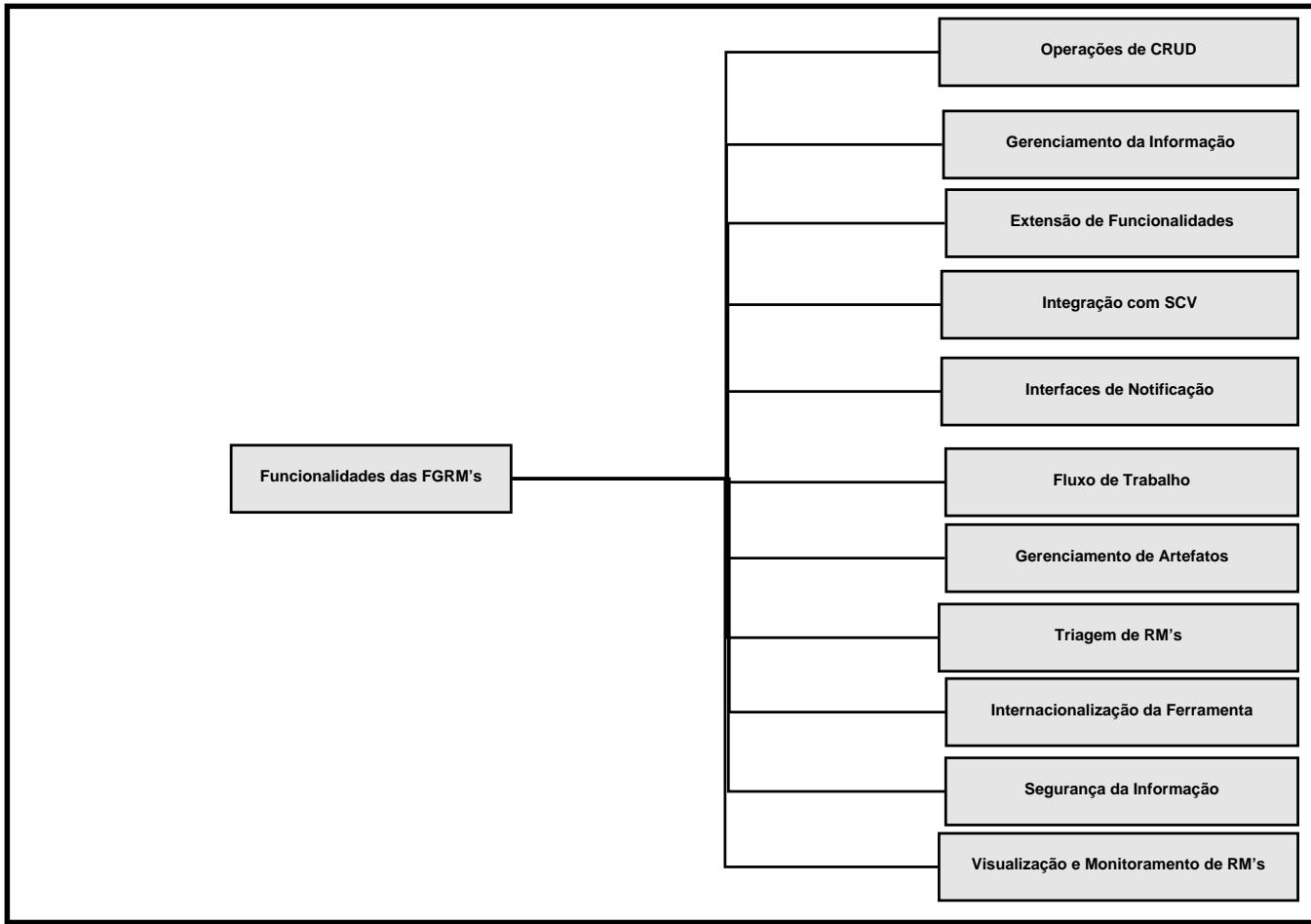


Figura 2.13: Modelo de funcionalidades básicas das FGRMs

O gerenciamento das RMs formam as funcionalidades centrais de uma FGRM. De uma maneira geral, uma das primeiras responsabilidades de uma FGRM é gerir a *criação, consulta, atualização e destuição* de uma RM. Estas funções podem ser agrupadas em um termo único denominado *Operações de CRUD* (acrônimo de Create, Read, Update e Delete na língua Inglesa). As principais categorias de funcionalidades que foram encontradas para a dimensão de *Gestão da RM* estão descritas a seguir.

Operações de CRUD: Nesta categoria estão as funcionalidades que dão suporte à criação, consulta, atualização e destruição das RMs. Com relação à criação verificamos que algumas FGRMs permitem a definição de *campos personalizáveis* para o preenchimento da RM. A ferramenta Bugzilla suporta a atuação sobre um campo personalizado de modo à capturar e pesquisar dados que são exclusivo do projeto ao qual pertence. Estes campos podem ainda ser exibidos com base no valor de um outro, para usá-los apenas quando for de interesse.

Esta categoria também agrupa as funcionalidades relacionadas a busca de RMs e a localização de duplicados. Durante o processo de criação de uma RM, uma das ferramentas possui a funcionalidade para detecção automatizada de duplicados. Para criar uma nova RM algumas ferramentas possuem diferentes *interfaces de entrada* de modo que uma RM pode ser criada através do envio de e-mail, utilizando dispositivos móveis ou mediante formulários próprios criados em qualquer site da web.

Verificamos ainda que algumas FGRMs permitem que o relato da RM seja realizado em linguagem de marcação como o Markdown⁸, que permite dentre outras opções a inclusão de código fonte com a sintaxe realçada. Isso possibilita visualizar de forma mais clara partes do código fonte podem ser incluídas na RM. Neste mesmo tópico encontramos funcionalidades para recuperar uma RM utilizando o texto relatado na RM, mediante filtros personalizáveis ou por meio de uma Linguagem de Domínio Específico (Domain-Specific Language - DSL em inglês) baseada em SQL.

Gerenciamento da Informação Dentro de um projeto de desenvolvimento ou manutenção de software gerenciar uma RM por vez não é muito eficiente. Neste sentido, é necessário que as FGRMs suportem RMs de forma agregada permitindo o gerenciamento em massa da informação armazenada. Este tópico contempla as funcionalidades que se dedicam ao armazenamento e consistência das informações contidas na FGRM. As ferramentas possuem funcionalidades para *suportar múltiplas bases de dados*, como os diferentes Sistemas de Gerenciamento de Banco de Dados disponíveis no mercado. Além disso, a ferramenta Bugzilla oferece funcionalidade própria para validação de consistências dos dados armazenados.

As FGRMs devem fornecer recursos através dos quais outras ferramentas possam interagir e manipular a informação que elas armazenam. Nesta dimensão estão as funcionalidades que permitem manipulação externa dos dados contidos nas RMs ou mesmo o desenvolvimento de novas funções ou comportamentos da FGRM mediante o uso de APIs⁹ e extensões.

Extensão de Funcionalidades As funcionalidades que compõem este grupo têm por objetivo entender o conjunto de funcionalidades oferecidas através de uma arquitetura de plugins ou mediante o suporte de APIs. Algumas ferramentas como o Github permitem realizar as atividades de gestão de uma RM mediante a utilização de uma API própria. No caso do Bugzilla e do Mantis é permitido o acesso à informação das RMs através de Webservice.

⁸<https://en.wikipedia.org/wiki/Markdown>

⁹https://en.wikipedia.org/wiki/Application_programming_interface

Integração com Sistemas de Controle de Versão As FGRM podem acessar os repositórios de código de fonte, gerenciados mediante um Sistema de Controle de Versão (SCV), permitindo que o usuário navegue pelo seu conteúdo, visualize e procure o conjunto de alterações realizadas. As ferramentas também possibilitam acesso à diferentes tipos de SCV, tais como Git, SVN, Mercurial e etc.

Interfaces de Notificação Neste tópico estão as funcionalidades oferecidas pelas FGRMs para notificar as diversas partes interessadas envolvidas em determinado projeto de software. As FGRMs podem notificar através de e-mail, RSS, Twitter e chats.

Esta dimensão foi criada para agrupar as funcionalidades que dão suporte ao processo de manter software, demonstrando que as FGRM gerenciam, além da própria RM, as pessoas e artefatos que colaboraram no desenvolvimento e manutenção de software.

Fluxo de trabalho Nesta categoria que dão suporte ao processo de trabalho adotado no desenvolvimento e manutenção de software. Nela estão incluídos funcionalidade para gerenciamento de tarefas e suporte à múltiplos projetos. Também é possível personalizar o fluxo de trabalho adotado. Esta customização é realizada através da definição de *situações* próprias que se adéquem às necessidades do projeto.

Gerenciamento de Artefatos O processo de manutenção de software pode consumir ou gerar diversos artefatos, tais como documentos de requisitos e arquiteturais dos software, código fonte, registros (logs) de teste e assim por diante [Cavalcanti et al., 2013]. Em alguns contextos, devido ao volume de artefato gerados, é importante que a FGRM dê suporte para armazenamento e recuperação deste ativos do processo de software. As FGRMs possuem funcionalidades que interagem diretamente com a documentação de software, geralmente no formato de Wikis. Além disso, algumas ferramentas permitem uma melhor visualização de anexos incluídos na RM, como por exemplo arquivos no formato CSV.

Triagem de RMs Este tópico descreve as funcionalidades relacionadas com o processo de triagem de RMs. O processo de atribuição de RM, também conhecido como triagem, possui como principal objetivo encontrar o desenvolvedor mais capacitado para manipular uma dada RM. As FGRMs dão suporte a esta atividade principalmente através da categorização das RMs. Todas as ferramentas analisadas permitem algum tipo de classificação através do uso de etiquetas.

Internacionalização da Ferramenta Neste tópico estão as características das FGRM que ajudam no desenvolvimento e/ou adaptação de um produto, em geral softwares de computadores, para uma língua e cultura de um país. As FGRM possuem tradução para diversos idiomas e também possuem funcionalidades que permitem à colaboradores criarem novas traduções.

Segurança da Informação Neste grupo estão as funcionalidades de uma FGRM que estão diretamente relacionadas com proteção de um conjunto de informações, no sentido de preservar o valor que possuem para um indivíduo ou uma organização. Assim as ferramentas oferecem funcionalidades para suporte à confidencialidade, integridade e autenticidade da informação armazenada.

Visualização e Monitoramento de RMs Em diversos contextos, devido ao volume das RMs, é importante que as partes interessadas na manutenção de software, possam visualizar e monitorar a situação das requisições que serão analisadas em determinado período. Neste contexto, as FGRM oferecem funcionalidades para visualizar a informação das RMs mediante quadro como aqueles utilizados nas metodologias Kanban ou SCRUM. Existem funcionalidades que permitem ao usuário visualizar um conjunto específico de RMs. Neste mesma categoria estão as funcionalidades para geração de relatório que ajudam aos gerentes do projeto na tomada de decisão.

2.3.5 Discussão

Para algumas funcionalidades não há uma separação clara em qual categoria ela pode ser encaixada, como por exemplo a possibilidade que algumas FGRM fornecem de personalizar os campos que compõem uma RM. Esta função está relacionada com a criação da RM (Operação de CRUD), contudo, também faz parte da definição de processo de trabalho próprio de um projeto, o que poderia categorizá-la como Fluxo de Trabalho. Esta mesma situação ocorre com as funcionalidades de deleção de uma RM que foram classificadas como *Operações de CRUD*, mas que tem relação com a categoria de *Segurança da Informação* já que para realizar tal ação o usuário deve ser identificado (login realizado no sistema) e autorizado para tal.

A análise das funcionalidades nos permite verificar que as tarefas das FGRM evoluíram de simplesmente gerenciar as RM para colaborar no processo de desenvolvimento e manutenção de software. Todavia, esta evolução não é tão rápida quanto o necessário. As ferramentas apresentam um suporte bem estabelecido para atividades relativas à gestão da RM, como por exemplo a criação de uma nova RM. Contudo,

ainda é bastante escassa funcionalidades que minimizem os problemas que ocorrem quando as RMs são geradas, como por exemplo, duplicadas ou baixa qualidade do relato.

É possível verificar que as FGRM oferecem funcionalidades que dão suporte a todo o ciclo de vida de uma RM, conforme discutido na Subseção 2.1.4.2. Todavia, grande parte do esforço fica a cargo do usuário da ferramenta, o que pode resultar em atrasos em situações em que se tem muitas RM para gerenciar. Um exemplo deste problema ocorre no processo de atribuição do Desenvolvedor responsável por solucionar determinada RM. Conforme discutido no Capítulo 2 esta atividade fica sob a responsabilidade do *Agente de Triagem*. Ele deve realizar a escolha de forma manual tendo em vista que as FGRM não apresentam funcionalidades que sejam capaz de ‘recomendar’ o desenvolvedor mais apto.

As FGRMs possuem funcionalidades que permitem a realização do papel ao qual este tipo de software se propõe. Não obstante, devido à sua crescente importância, seria necessário que este tipo de ferramenta incorpore funções e comportamentos que ajudem no processo de desenvolvimento e manutenção de software, especialmente em áreas como busca de duplicados, melhoria do relato e atribuição e classificação automatizadas.

2.3.6 Ameças à Validade

Classificar envolve categorização, e há uma literatura sofisticada sobre categorização, taxonomia e semântica, todas as quais são potencialmente relevantes [Rugg & McGeorge, 2005]. Em grande parte dos estudos a generalidade dos resultados é muitas vezes sacrificada pela riqueza e complexidade dos dados analisados. Neste sentido, podemos afirmar que o processo de classificação é, por natureza, uma avaliação subjetiva.

Uma ameaça à validade do trabalho está no processo de seleção das ferramentas. Apesar da escolha ter sido realizada com suporte de profissionais envolvidos em manutenção de software, não podemos garantir que o número de respondentes pode suportar que foi escolhido as ferramentas mais relevantes dentre aquelas disponíveis. Neste mesmo sentido, a formula que foi utilizada para definir as mais relevantes podem conter um enviesamento sobretudo pela forma que os pesos foram adotados, ou seja, não há como garantir que o fato de um participante entender que uma determinada ferramenta é muito relevante (peso igual a 5) mereça ser ponderado cinco vezes mais que uma outra que não é conhecida (peso igual a 1). Todavia ao bem do nosso conhecimento não há técnicas para classificação que não tenha influência da subjetividade.

Com relação à técnica de classificação utilizando Cartões de Ordenamento temos dois pontos principais de ameaças aos resultados. Como a extração dos dados foi realizada de forma manual pode ter ocorrido algum tipo de equívoco no processo como por exemplo a não coleta de determinada ferramenta por mero esquecimento. Todavia, um número pequeno de ferramentas foi selecionada tendo em vista a limitação desta extração manual. Um segundo ponto encontra-se na classificação dos cartões. Apesar do processo ter sido realizado em pares pode ter ocorrido uma classificação de forma incorreta o que pode acarretar em limitação dos resultados apresentados. Esta situação pode ocorrer porque para algumas funcionalidades não há uma fronteira clara para qual grupo ela pertence.

2.4 Resumo do Capítulo

Neste capítulo relacionamos a disciplina de Manutenção de Software com as FGRMs. Para tanto apresentamos e discutimos os principais conceitos relacionados e que foram utilizados durante os demais capítulos desta dissertação. Além disso, apresentamos um estudo realizado com a ajuda de levantamento por questionário. A partir deste levantamento foi obtido as principais funcionalidades existentes nas FGRMs. O conjunto comum de funcionalidades que foi encontrado será utilizado durante a dissertação para discutir a proposição de novas funcionalidades ou melhorias das existentes.