

# Capítulo 5

## Sugestões de Melhorias para FGRMs

### 5.1 Introdução

No levantamento descrito no Capítulo 4 os profissionais consultados se mostraram, em geral, satisfeitos com as funcionalidades disponibilizadas pelas FGRMs. Cerca de 90% dos participantes fizeram uma avaliação positiva da ferramenta que utiliza, além disso, uma quantidade equivalente disse que recomendariam a FGRM para um novo projeto.

Não obstante, naquela mesma pesquisa, apresentamos uma lista de possíveis novas funcionalidades para as FGRMs e perguntamos ao participante se ele sente falta de alguma. O resultado desta pergunta foi que cerca de 85% responderam positivamente, ou seja, os participantes se mostraram interessados na inclusão de pelo menos uma das funcionalidades propostas nas FGRMs que utilizam. A partir da análise deste resultado é possível inferir que os desenvolvedores estão satisfeitos com a ferramenta utilizada, contudo, *não conhecem ou não têm acesso ao potencial de funções que este tipo de software poderia oferecer.*

Diante do exposto, entendemos que podemos contribuir com o estado atual das funcionalidades das FGRMs propondo um conjunto de melhorias. As sugestões foram compiladas utilizando a literatura da área e os resultados obtidos nesta dissertação, especialmente com base nos Capítulos 3 e 4, na Seção 2.3 e nos estudos que propõem melhorias para as FGRM [Zimmermann et al., 2009a, Bettenburg et al., 2008b, Singh & Chaturvedi, 2011]. Estas recomendações podem ser utilizadas por pesquisadores interessados em conduzir estudos sobre o avanço da produtividade dos desenvolvedores mediante o uso das FGRMs. Além disso, os responsáveis

pelo desenvolvimento deste tipo de software, podem utilizar algumas das recomendações para a criação ou aperfeiçoamento das funcionalidades da FGRM que é responsável. Na mesma linha, os profissionais envolvidos com Manutenção de Software podem criar extensões (plugins) para as FGRMs com base no que foi proposto de modo a aplicar essas melhorias sugeridas em sua rotina de trabalho.

Este capítulo está organizado da seguinte forma: a Seção 5.2 apresenta as sugestões de melhorias propostas, cada sugestão foi seguida de uma breve discussão de como foi obtida e dos motivos de sua implementação; na Seção 5.3 realizamos a avaliação do que foi recomendado com base na opinião de profissionais que contribuem com projetos relacionados ao desenvolvimento de FGRMs; na Seção 5.5 discutimos os resultados obtidos no processo de avaliação; na Seção 5.6 apresentamos as ameaças à validade; encerramos o capítulo com um resumo na Seção 5.7.

## 5.2 Sugestões de Melhorias para FGRMs

As sugestões propostas neste capítulo não estão vinculadas exclusivamente à melhorias de funcionalidades existentes nas FGRMs. As recomendações podem representar o desenvolvimento de um novo comportamento ou a melhoria de outros já existentes. Cabe-nos ressaltar que o conjunto proposto não é exaustivo e é baseado nos resultados desta dissertação. Além disso, não houve compromisso com as dificuldades operacionais relacionadas com a implementação das funcionalidades.

Ao longo da elaboração das sugestões verificamos que não conseguíramos implementar todas elas, especialmente por entendermos que a análise desta complexidade está fora do escopo desta dissertação. Entretanto, como prova de conceito, a recomendação descrita na Seção 5.2.1 foi implementada como extensão de uma FGRM, conforme pode ser visto no Capítulo 6. É possível que algumas das recomendações propostas já estejam implementadas de maneira parcial ou integral. Contudo, não é possível validar esta premissa por conta de volume de ferramentas disponíveis quando esta dissertação foi escrita. Além disso, o processo de validação pode nos alertar sobre a eventual proposição de uma funcionalidade existente. Cada sugestão foi estruturada como uma seção deste capítulo onde apresentamos a *justificativa, o benefício gerado, as limitações e exemplos de utilização*.

### 5.2.1 Suporte à Qualidade do Texto Relatado

**Sugestão #01:** As FGRMs deveriam suportar algum tipo de retorno (feedback) relacionado com a qualidade do texto relatado.

**Justificativa:** Conforme discutido na Seção 2.1.3 o responsável por reportar uma RM pode ser tanto um usuário do sistema quanto um membro da equipe de desenvolvimento ou manutenção. Por esta razão, podemos encontrar Reportadores com diferentes níveis de conhecimento sobre o sistema. Do ponto de vista dos desenvolvedores, um problema recorrente é baixa qualidade do texto no relatado de uma RM, como por exemplo a falta de informação necessária para sua solução. Alguns estudos demonstram que a falta de informação, tais como as etapas para reproduzir a falha e o registro de pilha de ativação (stack track), dificultam mais o andamento do projeto do que RMs duplicadas [Bettenburg et al., 2008b, Bettenburg et al., 2007]. Neste sentido, as FGRM poderia fornecer uma funcionalidade capaz de reduzir o número de relatos de baixa qualidade através, por exemplo, de um retorno ao Reportador da qualidade do que foi informado no relato da RM.

**Benefício Gerado:** Com este tipo de funcionalidade uma FGRM poderia reduzir o tempo de análise de determinada RM porque o responsável por criá-la estaria ciente das informações necessárias à sua solução. Neste caso, o programador seria diretamente beneficiado já que teria a sua disposição um relato mais completo. Não obstante, um trabalho adicional seria dado ao reportador que em algumas situações devem rever as informações incluídas na RM.

**Limitações:** Alguns estudos sobre melhoria da qualidade do relato da RM focam prioritariamente nas que descrevem defeitos do software. Conforme discutimos na Seção 2.1.4, uma RM pode também estar relacionada com um pedido de melhoria. Uma dificuldade inicial é separar de forma automatizada as duas dimensões das RMs. Da mesma forma é necessário definir padrões distintos para analisar a qualidade do relato por conta de suas diferentes características e necessidades de uma RM.

**Exemplo de Utilização:** Após o usuário relatar um problema em determinada FGRM, a ferramenta de imediato avalia o texto corresponde ao atributo *relato* da RM gerada e apresenta ao responsável algumas dicas de como a informação fornecida poderia ser melhorada, como por exemplo através da inclusão de um arquivo com o histórico de execução do software (log). Como prova de conceito foi implementada uma extensão para uma FGRM com estas características. Os detalhes estão descritos no Capítulo 6.

### 5.2.2 Acesso Facilitado ao Código Fonte Incluído nas RMs

**Sugestão #02:** As FGRMs deveriam fornecer busca pelo código fonte contida no relato, comentários ou anexos das RMs.

**Justificativa:** As RMs permitem a inclusão de fragmentos de código fonte em seu relato ou comentários em diversas etapas do seu ciclo de vida. O código pode ser incluído durante a criação das RMs, nas discussões realizadas para a sua solução ou mesmo quando ela é concluída, onde recebe o nome de *patch*. Os fragmentos de código podem ser utilizados para descrever uma falha ou apresentar uma solução candidata. Apesar de sua importância este tipo de informação não é facilmente recuperada no contexto de uso de uma FGRM. O estudo de Damevski e outros [Damevski et al., 2016], que analisa o problema da localização de funcionalidades no código fonte (feature location), ressalta a importância da facilidade de acesso ao código fonte que pode propiciar a redução do tempo que a tarefa de manutenção é concluída bem como aumento na qualidade das alterações realizadas.

**Benefício Gerado:** O código fonte incluído em uma RM tem a comunicação como um dos seus objetivos. Com um fácil acesso a esta informação um desenvolvedor pode obter exemplos de solução para RMs similares. Este tipo de funcionalidade pode ser vantajosa em comparação com a uma busca estruturada, presente em grande parte das FGRMs, por permitir a recuperação com base em elementos específicos da linguagem de programação utilizada pelo software que a FGRM suporta, como por exemplo classes, funções, constantes e outros.

**Exemplo de Utilização:** Em certas ocasiões, uma RM em análise pode ter similaridades com outras solucionadas anteriormente. A similaridade pode ser devido a afetarem o mesmo módulo do sistema, por exemplo. Neste contexto, um desenvolvedor poderia realizar uma busca por código fonte na RM solucionada com o objetivo de encontrar fragmentos de código que ajudem no entendimento e solução da RM atual.

### 5.2.3 Ranqueamento pela Reputação do Reportador

**Sugestão #03:** As FGRMs deveriam permitir um ranqueamento das RMs de acordo com a reputação do Reportador.

**Justificativa:** Um problema recorrente em diversos projeto de software é a definição da ordem que um conjunto de RMs deve ser analisado. Conforme discutido na

Seção 3.3.2.3 alguns estudos vêm se dedicando em classificar as RM sob determinados critérios. Por outro lado, é possível observar que determinados *Reportadores* tem por hábito relatar RMs que possuem um maior grau de relevância ao projeto. Por exemplo, existem certos usuários do sistema que geralmente relatam RMs relativas à questões de segurança do sistema. Neste contexto, seria interessante se as FGRM aplicassem algum tipo de métrica de relevância aos seus usuários. Em um segundo momento, esta mesma métrica poderia ser utilizada com a finalidade de ranquear as RMs. Neste sentido, conforme a conveniência, um desenvolvedor poderia analisar primeiro as RMs de Reportadores com um maior grau de reputação.

**Benefício Gerado:** Uma das possíveis atividades desempenhadas pelo *Agente de Triagem* é definir o desenvolvedor mais apto para determinada RM. Ele pode utilizar diversos critérios como por exemplo a prioridade do que foi relatado. Segundo a nossa proposta, o *Agente de Triagem* pode previamente ordenar a lista de RMs sob sua responsabilidade pelo grau de reputação de quem as redigiu. A partir daquele ordenamento ele poderia atribuir primeiro aquelas de Reportadores com um maior grau de reputação. Com base nesta estratégica, é possível que RMs com maior relevância sejam analisadas primeiro. O conceito de relevância pode variar em diferentes projetos. Contudo, uma RM poderia ser classificada como relevante caso descreva um problema que afeta um grande número de usuários ou represente uma falha de segurança do software. Além disso, deve ser redigida de forma clara e fornecer as informações necessárias para sua solução.

**Exemplo de Utilização:** Conforme descrito anteriormente, um *Agente de Triagem* poderia ordenar a lista de RMs do qual é responsável pelo grau de reputação do *Reportador*. Em seguida daria início ao processo de atribuição analisando primeiro as RMs mais bem ranqueadas. Caso as RMs sejam relevantes o profissional poderia realizar a atribuição.

#### 5.2.4 Atalhos para filtros e classificação (rankings) das RMs

**Sugestão #04:** As FGRMs deveriam fornecer atalhos para filtros personalizáveis e classificações (rankings).

**Justificativa:** As RMs atribuídas a determinado desenvolvedor podem estar em diferentes estados. Existem aquelas que ainda não foram analisadas, que estão aguardando informações adicionais ou que estejam aguardando o Analista de Qualidade, por exemplo. Em geral, conforme discutido na Seção 2.3, as FGRMs permitem visualizar a

situação geral das RMs de um desenvolvedor mediante relatórios pré-definidos. Contudo, no levantamento mediante questionário, apresentado no Capítulo 4, identificamos por parte de alguns profissionais a necessidade de um acesso mais facilitado a este tipo de informação. Desta forma, sugerimos uma alteração nas interfaces das FGRMs de modo a fornecer acesso facilitado a filtros personalizáveis e classificações.

- Conforme relato dos participantes eles gostariam:
  - “*The ability to clearly visualize how many tickets are at the to do, in progress, to validate or done steps.*”.
  - “*History tracking, commenting, attachments, priority setting, task assignment, tie in with deployment systems.*”

**Benefício Gerado:** Com um acesso mais rápido às últimas RMs analisadas o desenvolvedor poderia ter uma noção geral do trabalho desenvolvido. Este panorama poderia ajudá-lo no planejamento de suas atividades diárias.

**Exemplo de Utilização:** Ao acessar a FGRM o desenvolvedor tem acesso as últimas  $n$  RMs que ele analisou. Além disso, ele poderia criar os filtros para acessar outras informações que julgar importante no desenvolvimento de suas atividades.

### 5.2.5 Suporte à Processos de Integração Contínua

**Sugestão #05:** As FGRMs deveriam fornecer suporte à Processos de Integração Contínua.

**Justificativa:** A solução de determinada RM pode levar a resultados não esperados em outros módulos do sistema mantido. Para minimizar os possíveis problemas recomenda-se a avaliação do impacto da RM. A Análise de Impacto estima o que será afetado no software e na documentação relacionada caso uma mudança proposta seja feita [Arnold, 1996]. A literatura sobre RMs discute algumas soluções com o objetivo de realizar a Análise de Impacto. Alguns exemplos de trabalhos nesta linha são apresentados na Seção 3.3.2.3. Dentro das propostas feitas pelos agilistas uma das possibilidades de reduzir os efeitos colaterais de uma mudança é periodicamente realizar a construção (build) do sistema. A *Integração Contínua* (IC) é a prática de construir (build) e implantar (deploy) imediatamente após um desenvolvedor consolidar (commit) o código fonte para o repositório [Aiello & Sachs, 2010]. Neste sentido, as FGRMs poderiam vincular a solução de uma RM com a execução de processo de IC. Uma discussão

sobre o ciclo de vida das RMs pode ser encontrada na Seção 2.1.4.2. Desta forma, seria possível mapear uma versão do sistema com o conjunto de RMs solucionadas até a sua geração. Este tipo de integração foi descrita como uma funcionalidade ausente nas FGRMs por alguns participantes do levantamento por questionário descrito no Capítulo 4.

**Benefício Gerado:** A integração de uma FGRM com processos de IC traria ao processo de manutenção os benefícios desta prática, tais como a redução de riscos e a facilidade de encontrar e remover falhas [Fowler & Foemmel, 2006]. Além disso, segundo o nosso entendimento, o fato de vincular a solução de uma RM com determinada versão de um sistema, pode minimizar problemas levantados no Mapeamento Sistemático descrito no Capítulo 3, com por exemplo a Localização do Problema (Seção 3.3.2.1) e a Estimativa de Esforço da RM (Seção 3.3.2.3). Em ambos os casos é possível aproveitar da facilidade que a IC fornecer a localização de uma falha que, por exemplo, pode ter sido causada pela solução de uma RM.

**Limitações:** Algumas vezes a mudança de situação de uma RM para *Fechada* pode não representar a sua efetiva solução. Por exemplo, a falha descrita pode ser definida como de baixo impacto e desta maneira não tratada, ou ainda o conjunto de fatores que geraram o problema descrito na RM deixa de existir. Nestas situação não faz sentido que responsável por fechar a RM engatilhe um processo de integração contínua.

**Exemplo de Utilização:** Após um desenvolvedor solucionar determinada RM, mudando a sua situação para *Fechada* por exemplo, a FGRM dispara o processo de compilação do sistema. O resultado da compilação poderia ser exibido em um painel de controle incluindo o responsável pela mudança mais recente no sistema, incluindo compilações que resultaram em falhas. O painel poderia incluir ainda dados da RM que engatilhou o processo de compilação como por exemplo o seu número e título.

### 5.2.6 Suporte além do Texto Simples

**Sugestão #06:** As FGRMs deveriam dar suporte além da especificação de texto simples.

**Justificativa:** Quando uma nova RM é registrada as informações mais relevantes estão no texto correspondente ao atributo *relato*. Além do problema da falta de informação no relato da RM, discutido na Seção 5.2.1, o Reportador enfrenta a dificuldade de expressar a falha ou pedido de melhoria do software. A baixa expressividade do

texto do relato pode estar relacionado pelo fato que algumas ferramentas analisadas utilizam texto simples (vide Seção 2.3). A possibilidade de usar linguagens com semântica de apresentação, como por exemplo Rich Text Format - RTF<sup>1</sup>, ou que permitam realce da sintaxe do código poderiam ajudar ao Reportador a expressar com maior qualidade a falha ou pedido de melhoria.

**Benefício Gerado:** Em um estudo sobre a transcrição de materiais de estudo, Voegler e outros [Voegler et al., 2014] mostraram que o uso da linguagem Markdown pode melhorar a qualidade técnica e a acessibilidade do documento resultante. As FGRMs poderiam se apropriar destas facilidades com o objetivo de melhorar a legibilidade do texto contido na RM. Diante do uso deste tipo de formato, os Reportadores poderiam, por exemplo, incluir no próprio relato uma parte do código fonte do qual ele supõem que está localizada a falha.

**Limitações:** Considerando que os responsáveis por reportar uma RM possuem diferentes níveis de conhecimento sobre o sistema. Neste sentido, é possível que nem todos os Reportadores consigam fazer uso de todas as facilidades oferecidas por um novo formato para relatar uma RM. Além disso, a utilização de uma linguagem que exija conhecimento prévio pode inibir o desejo de relatá-la.

**Exemplo de Utilização:** Conforme discutido na Seção 2.3, algumas FGRMs permitem a utilização de linguagens de marcação para relatar uma RM. O módulo para a criação de uma RM, que no contexto da plataforma Github corresponde ao elemento *issue*, permite o uso da linguagem Markdown como um dos formatos disponíveis para o relato da RM.

### 5.2.7 Classificação Automática pela Urgência da RM

**Sugestão #07:** As FGRMs deveriam fornecer uma classificação automática em termos da urgência da RM.

**Justificativa:** Conforme discutido na Seção 5.2.3 a escolha das RMs que serão analisadas é um problema recorrente nas equipes de manutenção. Algumas equipes de manutenção de software estão adotando práticas propostas pelos agilistas [Svensson & Host, 2005b]. Naquele contexto, um problema comum, é que as iterações (sprint) estão sujeitas à interrupções por demandas urgentes de

<sup>1</sup>[https://msdn.microsoft.com/en-us/library/aa140280\(office.10\).aspx](https://msdn.microsoft.com/en-us/library/aa140280(office.10).aspx)

clientes [Bennett & Rajlich, 2000]. Uma possível solução é a utilização de uma reserva de tempo (buffer) que não é alocada no planejamento da iteração de modo a atender eventuais demandas não esperadas [Schwaber & Beedle, 2002]. Para apresentação da solução proposta ainda é necessário definir quais RMs devem ser priorizadas durante a iteração, o que geralmente é realizado manualmente. As FGRM poderiam suportar à priorização automática de RMs urgentes.

**Benefício Gerado:** Ao realizarmos a priorização automática estamos reduzindo o esforço desempenhado por alguns membros da equipe de manutenção, em especial pelo Agente de Triagem, cujas atribuições estão descritas na Seção 2.1.3. No caso em que as RMs forem corretamente classificadas como urgentes, elas poderão ser solucionadas em um prazo mais curto. Para as equipes que organizam as suas atividades em iterações (sprints) pode ocorrer a redução de iterações interrompidas o que pode ocasionar frustração e desmotivação com relação ao planejamento e objetivos da iteração.

**Limitações:** A priorização automática pode ser vista como um problema de *Classificação de RM*, conforme discutido na Seção 3.3.2.3. Em geral, são utilizadas técnicas de Mineração de Dados com o objetivo de classificar uma RM, o que possui diversas limitações. Neste sentido, o uso de técnicas supervisionadas, ou seja, com suporte de membros da equipe de manutenção, podem apresentar melhores resultados.

**Exemplo de Utilização:** Após uma nova RM ser criada, a FGRM dispara um processo para determinar se ela deve ser classificada como urgente conforme critérios previamente definidos. Em positivo, a ferramenta realiza a priorização da RM através, por exemplo, da atribuição automatizada para o desenvolvedor mais apto.

### 5.2.8 Suporte à tarefas compartilhadas

**Sugestão #08:** As FGRMs deveriam suportar tarefas compartilhadas, permitindo o trabalho colaborativo.

**Justificativa:** Dentro do que é proposto pelos agilistas, o compartilhamento de código é visto como uma boa prática [Meyer, 2014]. Por outro lado, mantenedores tendem a ser tornarem especialistas nos sistemas do qual são responsáveis [?]. A propriedade compartilhada de tarefas permite que a carga de trabalho seja distribuída de forma mais igual, permitindo que os mantenedores sejam capazes de ajudar uns aos outros em períodos de muita atividade. No contexto das FGRMs, as tarefas compartilhadas poderiam ser representadas com a “propriedade” de uma RM por dois ou mais desenvolvedores. Uma

abordagem semelhante a esta sugestão foi realizada no estudo proposto por Banitaan & Alenezi [Banitaan & Alenezi, 2013] no qual os autores construíram uma comunidade de desenvolvedores baseados nos comentários realizados nas RMs. Cada nova RM criada era atribuída para uma comunidade. Os resultados mostraram que a abordagem atingiu uma precisão razoável de atribuição de RMs, além de construir um conjunto de desenvolvedores com experiência em solucionar determinadas RMs.

**Benefício Gerado:** A atribuição de uma RM a mais de um desenvolvedor ajuda a otimizar a carga de trabalho em toda a equipe e pode aumentar o moral do time. Neste contexto, os mantenedores deixam de ser especialistas em determinados sistemas para se tornarem generalistas, trabalhando com outros projetos. Esses benefícios são discutidos na literatura sobre o desenvolvimento e a manutenção que adotam as práticas propostas pelos agilistas [Dybå & Dingsøyr, 2008, Rudzki et al., 2009].

**Limitações:** Uma funcionalidade com suporte ao compartilhamento de tarefas padece dos mesmos desafios e problemas do processo de atribuição automatizada de RMs, conforme discutido na Seção 3.3.2.3. Além disso, a RM deve permitir a divisão atômica de tarefas de modo que cada atividade fique com um único desenvolvedor, o que nem sempre é possível.

**Exemplo de Utilização:** Quando uma nova RM é criada um processo automatizado define dois ou mais desenvolvedores como responsáveis. Posteriormente, a RM é dividida em tarefas que serão compartilhadas entre o conjunto de desenvolvedores para o qual ela foi atribuída.

### 5.3 Avaliação das Melhorias Propostas

Este Capítulo apresentou um conjunto de sugestões que foram construídas tomando como base a literatura da área e os resultados e contribuições desta dissertação. Com o objetivo de avaliar a relevância e o grau de facilidade de implementação das recomendações propostas, conduzimos um levantamento mediante questionário com profissionais que contribuem em projetos de código aberto hospedados no Github. Para realizarmos a coleta dos dados foi utilizado um levantamento (survey) através de um questionário eletrônico. O processo de seleção dos participantes, o desenho do questionário e como foi realizada a sua aplicação estão descritos a seguir.

### 5.3.1 Seleção dos Participantes

O público-alvo deste questionário é o conjunto de profissionais que estejam ligados ao processo de desenvolvimento e manutenção de algum projeto de FGRMs. A escolha se justifica tendo em vista que queremos avaliar a relevância das sugestões propostas e verificar a viabilidade de implementação do que foi recomendado como funcionalidades para as FGRMs. Neste sentido, optamos por um amostra de conveniência composta por profissionais que atuam como *contribuidores* em projetos de código aberto hospedados no Github.

Com cerca de 38 milhões de repositórios<sup>2</sup>, Github é atualmente o maior repositório de código na Internet. Sua popularidade e a disponibilidade de metadados, acessíveis através de uma API, tem tornado Github bastante atrativo para a realização de pesquisas na área de Engenharia de Software.

A seleção dos participantes iniciou com a escolha dos projetos que utilizou a ferramenta de busca avançada oferecido pela plataforma Github. Ela permite procurar por projetos utilizando critérios tais como data de criação, linguagem utilizada no desenvolvimento e proprietário do repositório, dentre outros. Neste estudo utilizamos critérios de seleção baseados em boas práticas recomendadas na literatura [Bird et al., 2009]. O fato de utilizarmos apenas o Github como a única fonte para seleção de participantes implica em certas ameaças à validade do estudo que serão discutidas na Seção 5.6. Os critérios de seleção dos projetos inclui os seguintes requisitos:

- Os projetos devem representar o desenvolvimento de uma FGRM.
- Os projetos devem ter no mínimo seis meses de desenvolvimento, para evitar aqueles que não tenham passado por um tempo de manutenção relevante.
- Os projetos devem ter no mínimo 200 revisões (commits) pelos mesmos motivos da restrição anterior.
- Os projetos obtidos devem estar os 50 mais populares que atendam aos demais critérios e estejam entre os melhores classificados pela ordenação via a opção “most stars”, que representa o número de usuário do Github que mostraram apreciação sobre o trabalho desenvolvido no repositório.

Após aplicação dos critérios descritos obtivemos os projetos retratados no Anexo A. Com base nos projetos selecionados ficou definido que a amostra a ser utilizada no levantamento seria os respectivos contribuidores. Um contribuidor é alguém

---

<sup>2</sup><https://github.com/features>. Acesso em junho/2016.

que participa efetivamente do desenvolvimento de um projeto, tendo o privilégio de acesso para alterar o código fonte. A solicitação de participação foi realizada por meio de correio eletrônico. O atributo foi coletado de forma automatizada apenas dos usuários da plataforma que disponibilizam esta informação como pública, de modo a preservar a privacidade dos contribuidores. A Figura 5.1 exibe uma página do projeto com seus respectivos colaboradores.

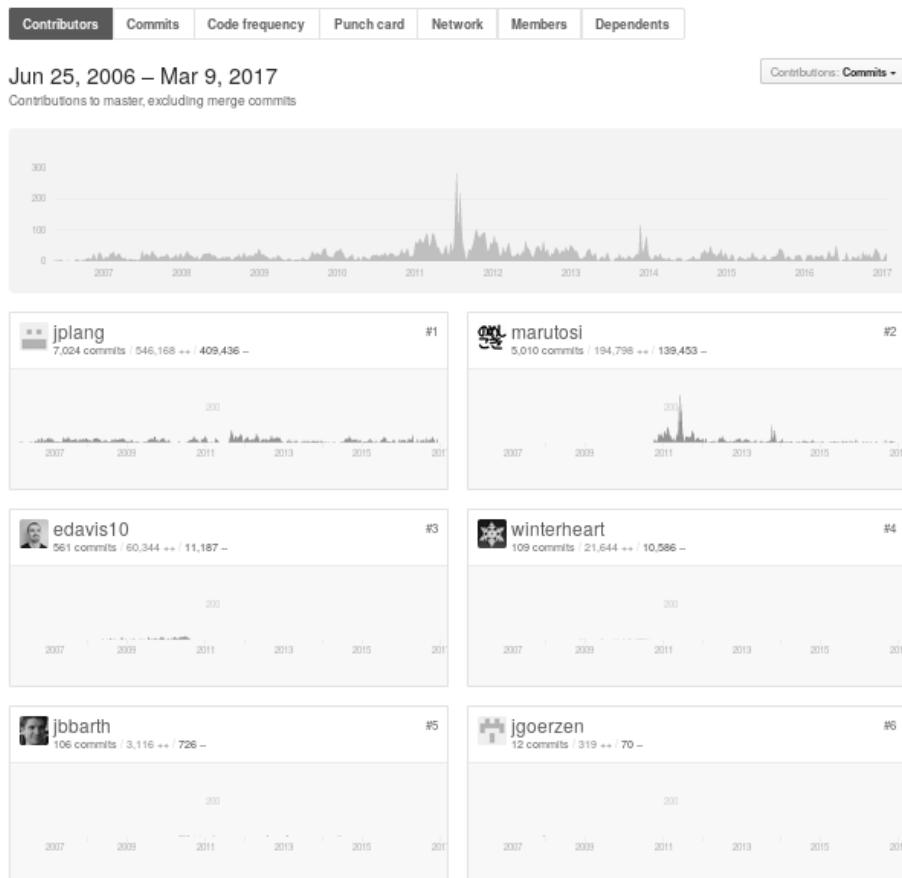


Figura 5.1: Lista de contribuidores do projeto Redmine

### 5.3.2 Desenho do Questionário

A fim de coletar a opinião dos participantes foi utilizado um questionário eletrônico. O questionário foi desenhado com a premissa de ser respondido em um prazo curto, de preferência entre 5 e 10 minutos. Neste sentido as perguntas foram organizadas em dois grupos principais. As questões do primeiro grupo têm por objetivo coletar a opinião dos profissionais sobre a relevância da recomendação proposta e o grau de dificuldade de implementá-la. As perguntas foram estruturadas como uma escala de Likert em

que o respondente deveria fornecer o seu nível de concordância para as declarações que lhe são apresentadas. No segundo grupo estamos interessados na formação de base (background) dos profissionais. Optamos por definir algumas das questões como não obrigatórias por entendermos que a impossibilidade ou falta de interesse em responder determinada pergunta impeça o participante de enviar os dados de outras que foram respondidas.

### 5.3.3 Processo de Aplicação

O questionário foi encaminhado à amostra de interesse através de correio eletrônico. O endereço foi coletado diretamente do projeto hospedado no Github. Foi desenvolvido um *script* na linguagem Python que permitia coletar o endereço de e-mail e automatizar o processo de envio. As mensagens foram personalizadas de modo a identificar o nome do usuário e o projeto do Github com base no template exibido a seguir. O pedido de participação foi enviado para uma amostra de 121 contribuidores.

O processo de envio consistia ainda de uma segunda mensagem de “lembrete” após dois dias. Esta estratégia foi adotada com base em estudos que discutem resultados em que o reenvio pode ser um dos fatores que aumentam a taxa de participação em levantamentos por questionários realizados através da web [Fan & Yan, 2010].

## 5.4 Resultados da Avaliação das Sugestões de Melhorias

Após o envio do formulários aos participantes obtivemos um total de 29 respostas. Como algumas das questões do formulário não eram de preenchimento obrigatório é possível que o número de respostas seja menor. O nível de participação foi similar ao observado em outros estudo em Engenharia de Software que utilizam o levantamento por questionário (survey) como método de coleta de dados [Fan & Yan, 2010]. Nas próximas seções apresentamos os resultados obtidos através do perfil dos participantes, a relevância das sugestões propostas e o grau de facilidade de implementação das mesmas.

### 5.4.1 Perfil dos Participantes

No levantamento realizado incluímos uma questão sobre qual sistema o desenvolvedor contribui. Esta informação é importante porque contribuidores que trabalham

com ferramentas que são “mainstream” podem ter uma maior resistência quanto à inclusão de novas funcionalidades. A Tabela 5.1 exibe o nome dos projetos que tiveram contribuidores que preencheram o nosso formulário. Verificamos nas primeiras posições ferramentas tradicionais como Mantis, Trac e Debbugs. As FGRMs que tiveram menos de duas participações foram agrupados sobre o termo “OUTROS”.

Projeto	Participantes
DEBBUGS	4
MANTISBT	4
TRAC	4
FOSSIL	3
BUGZILLA	2
REDMINE	2
OUTROS	6

Tabela 5.1: Projetos que os participantes contribuem.

Com relação à função desempenhada mais da metade dos participantes são desenvolvedores (53%). O segundo grupo de cargo com maior frequência são aqueles ligados à gerenciamento de equipes (Gerentes de Projetos, Chief Technical Officer e etc.). Verificamos ainda a participação de Engenheiros e Arquitetos de Software. Sobre o tempo de experiência, o percentual de 45% dos respondentes têm entre dez e vinte anos de experiência. A maior parte desempenha suas atividades em equipes de pequeno (2 - 5 pessoas) ou médio porte (mais do 10 pessoas). Quando questionamos sobre qual o tipo de atividade desempenhada pelo participante, cerca de 85% trabalham com desenvolvimento ou manutenção de software. Com base no perfil apurado entendemos que os participantes são capazes de analisar as sugestões de melhorias de modo a contribuir com este estudo.

### 5.4.2 Relevância das Sugestões

As sugestões de melhorias foram apresentadas aos participantes mediante uma escala de Likert. Os resultados podem ser visualizados na Figura 5.2. Em uma primeira análise verificamos que a maioria das recomendações tiveram uma avaliação positiva dos participantes (Concordo ou Concordo Fortemente). A exceção foi para a Recomendação #3 que trata da possibilidade de ordenar as RMs a serem analisadas pela reputação do Reportador. Por outro lado, as Recomendações #6 e #8 tiveram uma boa aceitação dos participantes. A primeira sugestão trata da utilização de uma linguagem, como o Markdown, para relatar uma RM. No caso da segunda recomendação foi proposto o suporte a tarefas compartilhadas, de modo que uma RM tenha mais de um “dono”.

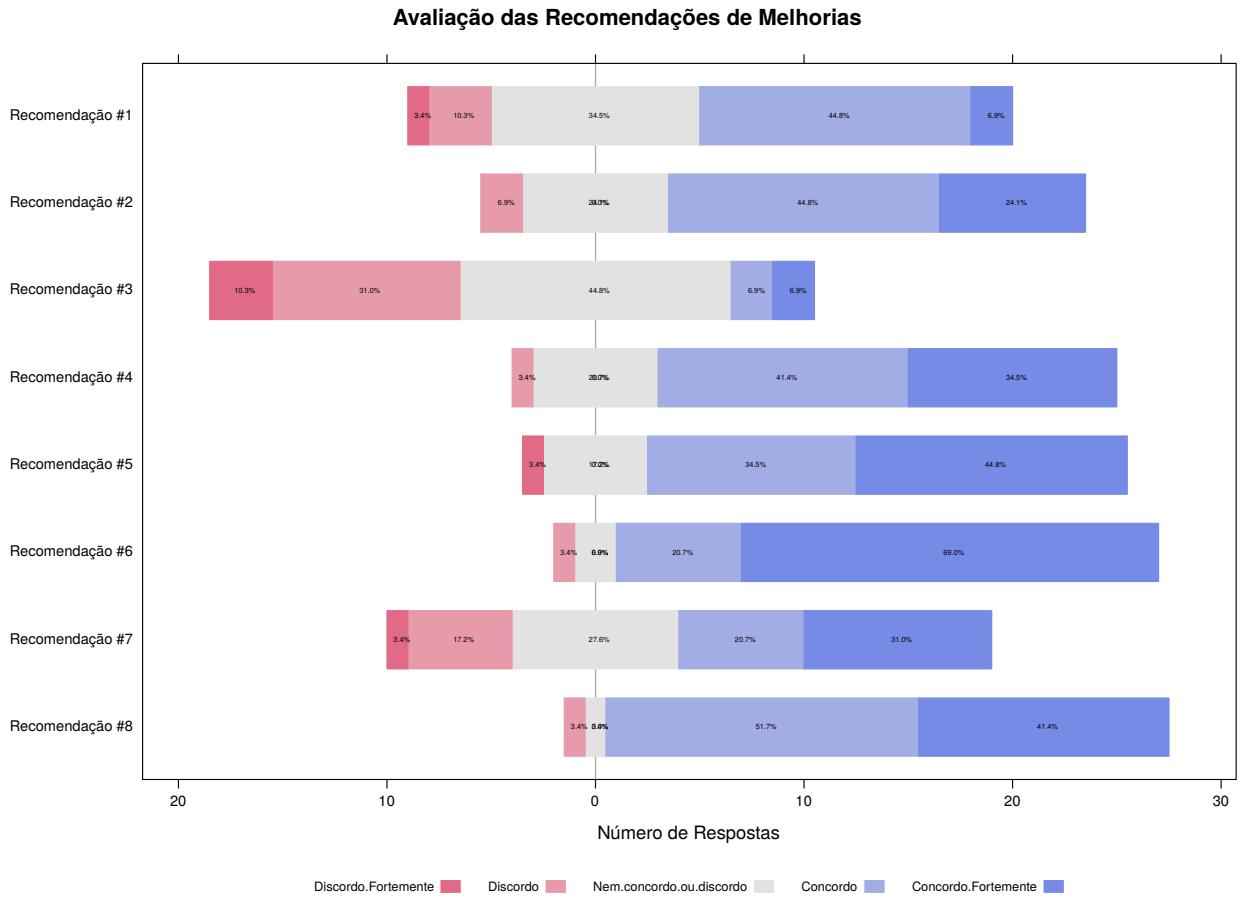


Figura 5.2: Avaliação das Sugestões de Melhorias

Para avaliar quais recomendações tiveram maior aceitação definimos um ranqueamento. O ordenamento consistiu em aplicar pesos para cada item da escala de Likert conforme a Tabela 5.2. O valor é obtido multiplicando a frequência de determinado item da escala pelo seu respectivo peso. A Tabela 5.2 exibe as recomendações ordenadas pelo nível de aceitação dos participantes. É possível visualizar o número de respostas que cada item recebeu.

Item da Escala	Peso
Discordo Fortemente	-2
Discordo	-1
Nem concordo ou discordo	0
Concorde	1
Concorde Fortemente	2

Tabela 5.2: Pesos utilizados para ordenar as sugestões propostas.

Com base na Tabela 5.3 verificamos que recomendações que sobressaíram: utilização nas FGRMs de uma linguagem além do texto simples para redigir uma RM

(#3), suporte à tarefas colaborativas (#8) e integração com processo de Integração Contínua (#5). Por outro lado as sugestões de diferenciar as RMs pela reputação do Reportador (#3) e avaliar a qualidade do relato (#1) não foram avaliadas como as mais relevantes pelos participantes. É importar notar que as recomendações que ficaram nas últimas posições tiveram a opção “Nem concordo ou discordo” selecionadas com maior frequência. Este fato pode indicar que não houve uma rejeição pelos participantes, mas possivelmente, uma baixa compreensão do que estava sendo proposto.

Recomendações	Discordo Fortemente	Discordo	Nem concordo ou discordo	Concordo	Concordo Fortemente	Ranking
Recomendação #6	0	1	2	6	20	45
Recomendação #8	0	1	1	15	12	38
Recomendação #5	1	0	5	10	13	34
Recomendação #4	0	1	6	12	10	31
Recomendação #2	0	2	7	13	7	25
Recomendação #7	1	5	8	6	9	17
Recomendação #1	1	3	10	13	2	12
Recomendação #3	3	9	13	2	2	-9

Tabela 5.3: Ranking das sugestões propostas

### 5.4.3 Implementação das Sugestões

Neste etapa do estudo estávamos interessados em avaliar o nível de dificuldade para implementar as sugestões propostas. A informação foi obtida através de uma escala de Likert cujos resultados estão apresentados na Figura 5.3.

As sugestões foram ordenadas pelo grau de dificuldade de implementação. Utilizamos um similar ao descrito na Seção 5.4.2 onde o valor é obtido multiplicando a frequência de um item da escala por valor previamente definido, do qual chamamos de peso. Os pesos estão apresentados na Tabela 5.4.

Item da Escala	Peso
Muito Difícil	-2
Difícil	-1
Neutro	0
Fácil	1
Muito Fácil	2

Tabela 5.4: Pesos utilizados no ranqueamento das sugestões de melhorias

Quando foram questionados sobre a facilidade de implementação das funcionalidades para o suporte do relato de uma RM além do texto simples (#6), a criação de atalhos e filtros personalizáveis (#4) e o ranqueamento das RMs pela reputação do Reportador (#3). Segundo os participantes funções como o suporte a tarefas compartilhadas (#8) e análise da qualidade do relato (#1) foram consideradas de um maior

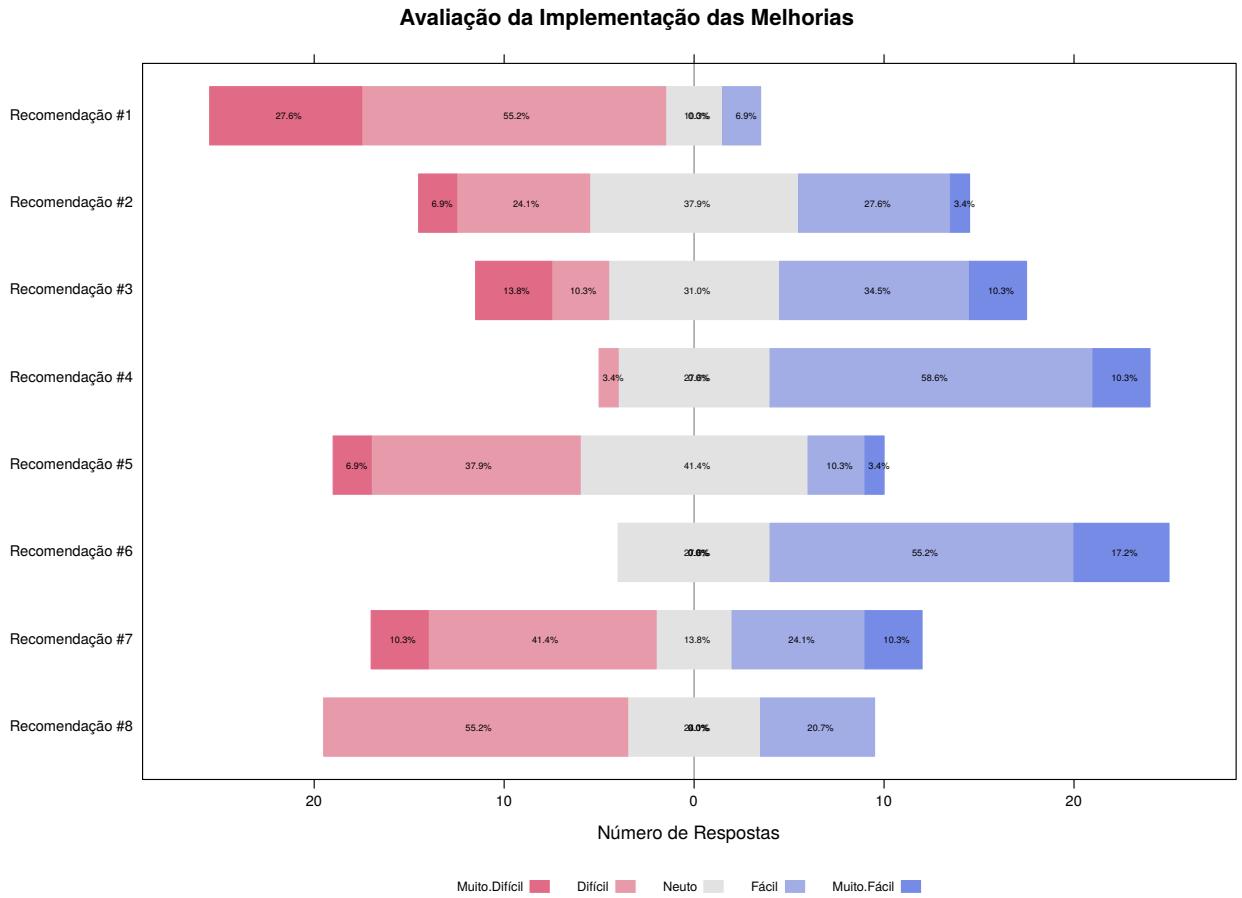


Figura 5.3: Avaliação sobre a implementação das sugestões propostas.

grau de dificuldade de implementação. O fato interessante é que a recomendação #3 foi considerada como uma das mais fáceis de implementar, entretanto, está entre aquelas que tiveram menor aceite entre os participantes. Este fato pode sugerir que sua possível rejeição não estaria ligada à sua complexidade de desenvolvimento, mas a fatores como o não interesse em classificar aqueles que reportam uma RM. Em geral, ao analisarmos a Figura 5.3 é possível verificar que os participantes entenderam que metade das sugestões propostas possuem uma dificuldade média, enquanto a outra metade pode ser considerada com um baixo grau de complexidade.

## 5.5 Discussão

Em geral podemos considerar que as sugestões propostas tiveram uma boa aceitação dos participantes. Em média 32% dos participantes avaliaram as recomendações “Concordo” ou “Concordo Fortemente”. Além disso, por volta de 22% em média selecionaram a opção “Nem concordo ou discordo”. Segundo o nosso entendimento as respostas

Recomendações	Muito Difícil	Difícil	Neutro	Fácil	Muito Fácil	Ranking
Recomendação #6	0	0	8	16	5	26
Recomendação #4	0	1	8	17	3	22
Recomendação #3	4	3	9	10	3	5
Recomendação #2	2	7	11	8	1	-1
Recomendação #7	3	12	4	7	3	-5
Recomendação #5	2	11	12	3	1	-10
Recomendação #8	0	16	7	6	0	-10
Recomendação #1	8	16	3	2	0	-30

Tabela 5.5: Ordenamento das sugestões pelo grau de dificuldade.

poderiam ser alteradas para uma visão mais positiva, com por exemplo “Concordo” ou “Concordo Fortemente”, caso fosse possível fornecer mais detalhes ao participantes.

Com relação as recomendações propostas, verificamos que a utilização de uma linguagem além do texto simples, como por exemplo o Markdown, foi muito bem avaliado. Este tipo de recomendação tem como principal objetivo aumentar o poder de expressão do relato, tal como a possibilidade do destaque da sintaxe do código fonte. Conforme pode ser observado na Seção 2.3 trata-se de uma funcionalidade encontrada em uma das FGRMs analisadas. Entretanto, o nosso resultado demonstra, com base na amostra utilizada, que deveria ser expandida para outras ferramentas.

O suporte à tarefa compartilhadas (sugestão #8) também foi muito bem avaliada. Esta recomendação surgiu das tentativas de implantação das propostas dos agilistas [Svensson & Host, 2005b]. A recomendação defende que uma determinada RM não tenha um “dono”, mas que a responsabilidade seja dividida entre dois ou mais membros da equipe. Esta divisão de tarefas pode resultar na melhor distribuição do conhecimento entre a equipe. A prevalência deste tipo de funcionalidade pode estar relacionada com o desejo das equipes de manutenção de utilizar algumas das práticas dos agilistas. Seria necessário um aprofundamento da opinião dos participantes para confirmarmos esta hipótese. Apesar da sua popularidade entre os participantes, esta recomendação ficou entre aquelas com maior grau de dificuldade de implementação.

Por outro lado, as sugestões que tem algum tipo de relação com a interface das FGRMs (sugestões #6, #4, #3 e #2) foram consideradas como mais “fácil” de implementar com mais frequência. Entretanto, a sugestão sobre a qualidade do relato (#1) ficou entre aquelas com maior grau de dificuldade. Esta classificação pode ser decorrente do carácter subjetivo que a qualidade do relato pode ter. Em cada projeto a qualidade do relato pode ter características distintas o que pode dificultar o seu suporte.

## 5.6 Ameaças à Validade

Para avaliarmos as sugestões propostas utilizamos um levantamento através de uma amostra de conveniência. Apesar da taxa de resposta está dentro da faixa observada na literatura, o total de participantes não nos permite extrapolar os resultados para todos os contextos em que as FGRMs estão inseridas. Adicionalmente, os critérios utilizados para seleção, como por exemplo, seis meses de desenvolvimento ou ter no mínimo duzentas revisões, não nos permite afirmar que escolhemos os projetos mais representativos para o nosso público-alvo.

Ao utilizarmos apenas projetos públicos hospedados no Github pode ter causado algum tipo de direcionamento, como por exemplo foco em projetos de código aberto. Além disso, não há garantias que os critérios utilizados para seleção, como por exemplo seis meses de desenvolvimento ou ter no mínimo 200 revisões (commits), não nos permite afirmar que escolher os projetos mais representativos para o nosso público-alvo.

A estrutura das perguntas do formulário podem ter causado impacto na quantidade de respostas ou na opção escolhida pelos participantes. Esta situação pode ter ocorrido especialmente quando apresentamos as sugestões. No caso de escrevermos de maneira extensa a explicação corremos o risco do sujeito não ler e não responder. Entretanto, se o texto fosse escrito de forma “concisa” corremos o risco de ficar vago, o que pode ter impacto nas respostas. A Tabela 5.1 detalha os projetos que os participantes contribuem. É possível observar que os participantes trabalham com ferramentas que são “mainstream” e portanto têm o viés de visão mais “tradicional”.

Em um dos comentários um dos participantes citou que algumas das sugestões propostas podem ter funcionalidades similares em outras FGRMs. Acreditávamos que a escolha destas 08 sugestões envolveria as principais características, mas existe possibilidade de ficar alguma característica relevante de fora. Infelizmente, posteriormente, descobrimos que ficaram de fora algumas dessas características.

## 5.7 Resumo do Capítulo

O objetivo do levantamento foi avaliar a proposição de funcionalidades para FGRMs. A avaliação foi realizada com base na opinião de desenvolvedores que contribuem para projetos de código aberto deste tipo de software. Em geral, as sugestões de melhorias foram bem avaliadas. Mais de 30% avaliaram as recomendações de forma positiva (“Concordo” ou “Concordo Fortemente”). Quanto a dificuldade de implementação, metade delas foi escolhida como fácil. Estes resultados nos permite considerar a im-

plementação de pelo menos uma das sugestões como Prova de Conceito. O próximo capítulo foi construído com este objetivo.