Uma Visão Geral do Bugzilla, uma Ferramenta de Acompanhamento de Alterações

Christian Robottom Reis
<kiko@async.com.br>
Async Open Source
R. Dona Alexandrina, 2534
São Carlos, SP
13566-290

Resumo

Ferramentas de acompanhamento de alterações permitem registrar e gerenciar modificações de software. Estas ferramentas são utilizadas para apoiar a fase de manutenção de software, proporcionando ao processo maior visibilidade, comunicação direcionada, e um registro histórico da evolução da base de código. Bugzilla é uma das ferramentas de acompanhamento de alterações mais populares entre as disponíveis como software livre. Inicialmente desenvolvida para apoiar o desenvolvimento do browser Mozilla, é atualmente utilizada em um grande número de projetos de software tanto livres quanto proprietários. Este artigo oferece uma visão geral do Bugzilla, incluindo uma introdução a ferramentas de acompanhamento de alterações, e uma descrição de alterações implementadas para promover melhor usabilidade.

Abstract

Change-tracking tools provide a means to register and manage change requests and software modifications. These tools are mainly used to support the maintenance phase, promoting better visibility, focused communication, and a historical record of the evolution of the codebase. Bugzilla is one of the most popular change-tracking tools available as free software. Initially developed to aid development of the Mozilla browser, it is currently used in a large number of software projects, both free and proprietary. This article provides an overview of Bugzilla, including an introduction to change-tracking, and a description of changes implemented in the tool to promote better usability.

1 Introdução

A fase de manutenção de software é caracterizada por um processo constante de solicitações de alteração, acompanhado da posterior implementação e integração destas. Estas alterações podem ser causadas por uma série de motivos[5], dois dos mais comuns sendo a solicitação de uma nova funcionalidade requerida pelo usuário, e o informe de um defeito no funcionamento esperado do software. O processo pelo qual é realizada uma alteração pode ser descrito da seguinte forma:

1. Um usuário identifica um problema ou necessidade de alteração no software.
2. O usuário informa por algum meio sua solicitação à equipe responsável pela manutenção do software.
3. Uma ou mais pessoas desta equipe fazem triagem deste informe, e decidem a procedência e viabilidade da solicitação.

4. Caso seja julgada procedente, a forma de implementação é discutida e implementada pela equipe.

5. Implementada a alteração, é aplicada ao código conforme o processo de integração praticado. Este processo pode incluir revisão, testes de integração, e outras práticas de garantia de qualidade.

Para auxiliar este processo de constante requisição, alteração e integração, existe uma atividade de apoio à manutenção de software denominada **acompanhamento de alterações**. Esta atividade dedica-se ao registro e gerência de alterações em software, e está presente em todo o ciclo de vida da alteração, indo desde sua requisição até sua integração. A atividade existe para promover visibilidade e registro histórico ao processo, além de melhor comunicação entre os envolvidos na manutenção. Participam da atividade diversos elementos da equipe:

- Desenvolvedores estudam as solicitações, discutem com os responsáveis pelos requisitos do sistema, e codificam as modificações.

- A equipe de garantia de qualidade gerencia estes informes para realizar sua triagem e a revisão das alterações propostas, e faz testes com o produto da alteração para garantir ausência de regressões.

- A gerência de projeto escalona alterações de acordo com a sua necessidade, e decide quais serão incluídas nos próximos lançamentos do software.

Para auxiliar o trabalho dos diversos grupos envolvidos na manutenção de software, existem softwares denominados ferramentas de acompanhamento de alterações[1], que registram solicitações, fornecendo um veículo de comunicação direcionado para a discussão da sua implementação.

Como dito acima, o ciclo de vida de uma alteração tem início no momento em que é identificada e solicitada pelo usuário do sistema. A ferramenta de acompanhamento de alterações geralmente utiliza a solicitação de alteração (ou **informe de alteração**) como o seu elemento fundamental. A esta solicitação é associado um número seqüencial, e uma ou mais das seguintes informações:

- uma descrição do problema ou necessidade identificada,

- o nome e endereço de uma ou mais pessoas envolvidas, incluindo o informante, o responsável pela garantia de qualidade, e o desenvolvedor envolvido com a alteração,

- discussão e comentários sobre a procedência e implementação da alteração,

- dados de prioridade para a alteração,

- arquivos anexos associados à alteração, contendo casos de teste e código-fonte.

A atividade de acompanhamento de alterações é essencial à garantia de qualidade do software em manutenção[3], de forma que uma ferramenta para apoiar esta parte do processo pode proporcionar diversas vantagens:

- Existe um canal de comunicação bastante direcionado para discutir a procedência e impacto da alteração. Esta comunicação é registrada, e oferece visibilidade ao andamento do processo.

- No momento da implementação, a revisão do código pode ser convenientemente integrada ao informe de erro.

- Relações de prioridade, gravidade e dependência entre alterações podem ser estabelecidas de forma a alocar esforço mais eficientemente.

[1]Este grupo de softwares inclui as ferramentas de acompanhamento de defeitos. O termo em inglês é *change-tracking* ou *defect-tracking*.

- A ferramenta oferece uma forma conveniente de armazenar um histórico de alterações. Acumulando este histórico, a ferramenta pode oferecer uma análise das freqüências das alterações e de suas causas.
- A verificação de regressões causadas por alterações pode ser facilitada, especialmente quando a ferramenta é combinada a um controle de versões como o CVS[1].
- Notificações redundantes de uma alteração podem ser identificadas e registradas.

Por ser utilizada por muitos elementos da equipe de desenvolvimento ao longo de toda a fase de manutenção, é importante que a ferramenta de apoio seja de boa usabilidade, e adaptada a usuários especialistas. Este artigo descreve uma ferramenta específica, Bugzilla, que implementa grande parte da funcionalidade descrita acima.

2 Bugzilla

Bugzilla é uma ferramenta de controle de alterações implementada em uma aplicação Web: é acessada através de um browser WWW. Um exemplo de uma das páginas que compõem o Bugzilla está visível na figura 1.

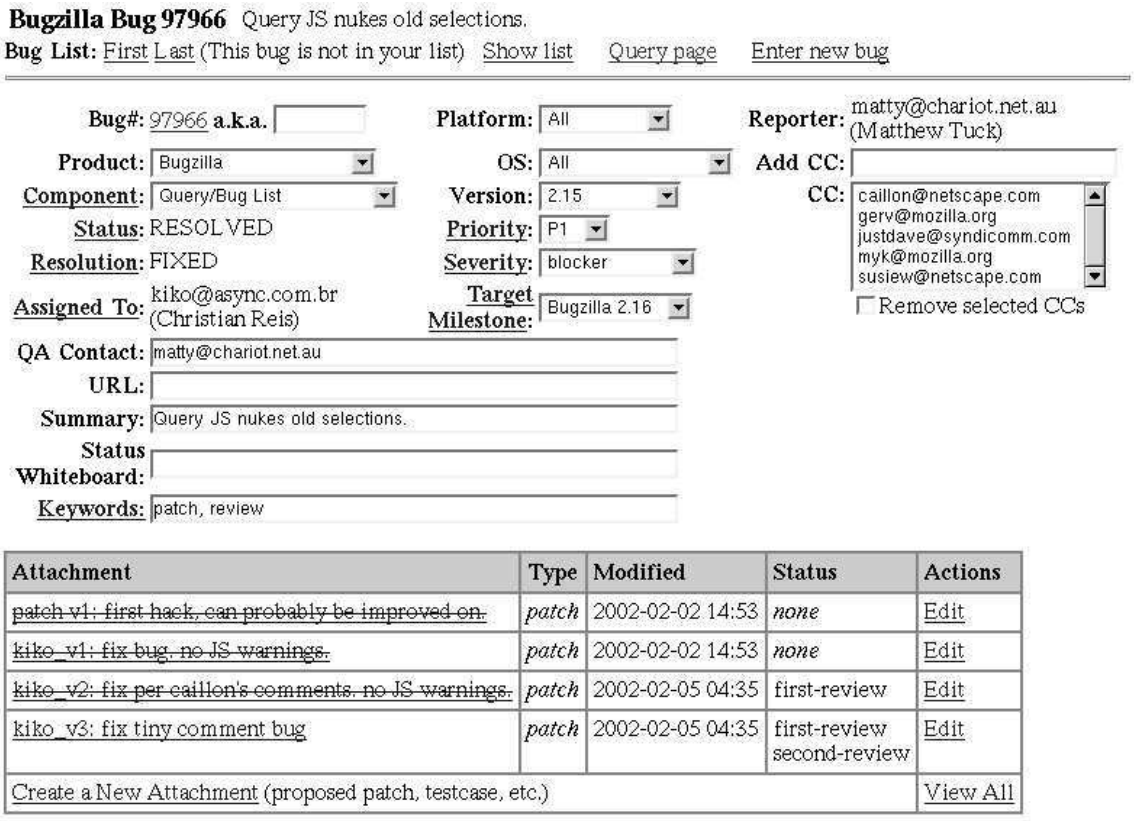

Figura 1: Um bug exibido em um formulário Bugzilla.

Bugzilla é a ferramenta mais importante do conjunto de ferramentas de desenvolvimento construída para apoiar o projeto Mozilla (`www.mozilla.org`). A sua instalação principal, publicamente disponível pelo site `http://bugzilla.mozilla.org/`, possui atualmente mais de 160.000 bugs registrados, com comentários feitos por mais de 8.000 usuários. Este site é utilizado diariamente por

centenas de desenvolvedores envolvidos com o projeto Mozilla, registrando mais de 5.500 visitas diárias na semana de 31 de Fevereiro a 6 de Março, 2002[2].

A ferramenta é utilizada em uma série de projetos, tanto livres quanto proprietários; o grupo de empresas e organizações que utilizam o Bugzilla inclui a NASA, a Redhat.com, a Conectiva, a Ximian, e o projeto GNOME.

Bugzilla é uma ferramenta licenciada como software livre; em outras palavras, possui código fonte publicamente disponível, e pode ser alterada e redistribuída livremente. O fato de ser software livre a torna especialmente adequada a grupos de desenvolvimento buscando uma ferramenta que tenha custo acessível, flexibilidade e garantia de sobrevida. Por este motivo, é de grande valor seu estudo e aprimoramento pela comunidade de ferramentas de apoio a manutenção.

2.1 Conceitos e Funcionalidades do Bugzilla

O elemento de informação básico do Bugzilla é o **bug**, que no contexto do Bugzilla serve para caracterizar tanto defeitos quanto requisições de nova funcionalidade[2]. Cada bug é identificado por um número seqüencial (e, opcionalmente, por um *alias* ou apelido), e é associado ao módulo (produto e componente) e à versão em que está presente.

Bugzilla implementa um ciclo de vida completo para cada bug. Este ciclo tem início no momento em que o bug é informado e passa por iterações de discussão e desenvolvimento antes de ser 'fechado', quando chega ao seu estágio final. Há diversas situações de fechamento possíveis, indo desde a decisão de não-procedência da solicitação até a aceitação e efetiva integração do código da alteração. A tabela 1 indica as fases pelas quais o bug passa, e as atividades do processo que ocorrem durante cada uma delas.

Estado do bug		Atividades
Informado	UNCONFIRMED, REOPENED	Triagem
Confirmado	NEW	Discussão, Alocação
Em Implementação	ASSIGNED	Discussão, Codificação, Revisão
Fechado	FIXED, WONTFIX, INVALID, DUPLICATE	Verificação (e Reabertura)

Tabela 1: Fases do ciclo de vida de um bug (os termos em inglês são códigos de estado Bugzilla.)

Embora a funcionalidade da ferramenta seja extensa, as operações principais oferecidas ao usuário são simples e diretamente ligadas às atividades de manutenção: a criação de um novo bug; sua triagem, revisão, escalonamento, discussão e fechamento; e a obtenção de relatórios e consultas baseados em propriedades do bug. Para suportar estas operações, Bugzilla implementa uma série de características particulares, que incluem:

- **Contas de usuário**: cada usuário Bugzilla possui uma conta própria pela qual é identificado, e que lhe permite acessar e alterar os atributos dos bugs. Criar uma conta é um processo instantâneo e se baseia no email do usuário.

- **Formulário do Bug**: todo bug possui um conjunto completo de informações, incluindo uma descrição, estado atual, prioridade, versão, plataforma, e o endereço das pessoas envolvidas.

- **Comentários**: como cada bug armazena consigo uma lista de comentários, é possível acompanhar exatamente qual foi a discussão e *rationale* que motivaram a alteração e sua implementação. O histórico de comentários é uma fonte rica de informação sobre a alteração, e permite que se acompanhe no futuro qual foi exatamente a razão pela qual aquela implementação foi feita.

- **Gerenciador de anexos**: ao bug podem ser anexados um ou mais arquivos. Em geral, são incluídos o código proposto da alteração, imagens exemplo e casos de teste.

[2]O termo bug é normalmente associado apenas a defeitos de software.

- **Interface de consulta**: como o número de bugs pode vir a ser grande (no caso do projeto Mozilla, o número já supera os 160.000), é fornecida uma interface que permite buscar por bugs através das suas características. É possível, inclusive, efetuar alterações simultâneas em múltiplos bugs com base nos resultados desta busca.

- **Integração com correio eletrônico**: cada bug reportado e cada alteração efetuada podem ser enviados por email para um ou mais recipientes cadastrados.

A alta demanda por funcionalidade e facilidade de uso, gerada em grande parte pelo seu principal usuário, o projeto Mozilla, tem estimulado seu desenvolvimento. Participam da equipe Bugzilla diversas pessoas que trabalham à distância, coordenadas por correio eletrônico, IRC e pelo próprio Bugzilla. Nossa experiência com a ferramenta indica que é uma solução eficiente e bastante completa para apoiar a manutenção de software.

2.2 Tecnologias utilizadas no Bugzilla

O Bugzilla é composto de diversas páginas WWW, implementadas como *scripts* CGI. A linguagem principal de desenvolvimento é Perl, e o código gerado (como resultado do processamento do CGI) inclui HTML, Javascript e CSS. O Template Toolkit, uma linguagem de especificação de modelos de documentos HTML, é utilizado para a elaboração das páginas; seu uso permite que a interface do Bugzilla seja facilmente customizada e internacionalizada.

Uma das características das ferramentas de apoio a desenvolvimento do projeto Mozilla é o fato de que todas estão integradas: a partir de uma listagem de modificações no CVS, obtida pela ferramenta Bonsai (`bonsai.mozilla.org`), é possível identificar e acessar o bug associado àquela alteração. Esta propriedade auxilia a tarefa de identificar as causas de regressões funcionais ou de performance.

3 Funcionalidades implementadas na ferramenta Bugzilla

Como desenvolvedor do Bugzilla, este pesquisador tem contribuído com diversas implementações ao longo dos últimos dois anos. A maior parte destas trata de adaptações da interface com o usuário para permitir melhor usabilidade e padronização. Além destas alterações, tem feito revisões e opinado freqüentemente sobre outras alterações de código.

As seções seguintes descrevem algumas das alterações mais importantes implementadas. Como o próprio Bugzilla utiliza uma ferramenta de controle de alterações, em cada seção está colocado o número do bug correspondente; para visualizá-los, basta visitar `bugzilla.mozilla.org`.

3.1 Atualização dinâmica de controles do formulário de busca (bug 96534)

Esta tarefa foi realizada em resposta a solicitações dos usuários do site `bugzilla.mozilla.org`. O formulário de busca possui um conjunto de controles HTML do tipo SELECT com produto, versões, componentes e versão-alvo, visível na figura 2. A funcionalidade originalmente implementada permitia que, ao selecionar um produto, as listas de versão, componente e versão-alvo atualizassem para conter apenas os elementos respectivos àquele produto. No entanto, para instalações Bugzilla com um grande número de produtos e componentes, esta atualização demorava muito além do tempo de resposta aceitável para um website interativo[6]: em certas ocasiões, mais de 50 segundos.

Para resolver esta questão, foram reprojetadas as estruturas de dados onde estavam armazenadas as listas de produtos, componentes, versões e versões-alvo. Além disso, foram implementadas funções Javascript para sintetizar vetores e ordená-los. Para dar uma idéia do nível de revisão e preocupação com qualidade do projeto, é interessante ressaltar que foram produzidas 13 versões da alteração até que a modificação fosse aceita pela equipe de revisores Bugzilla.

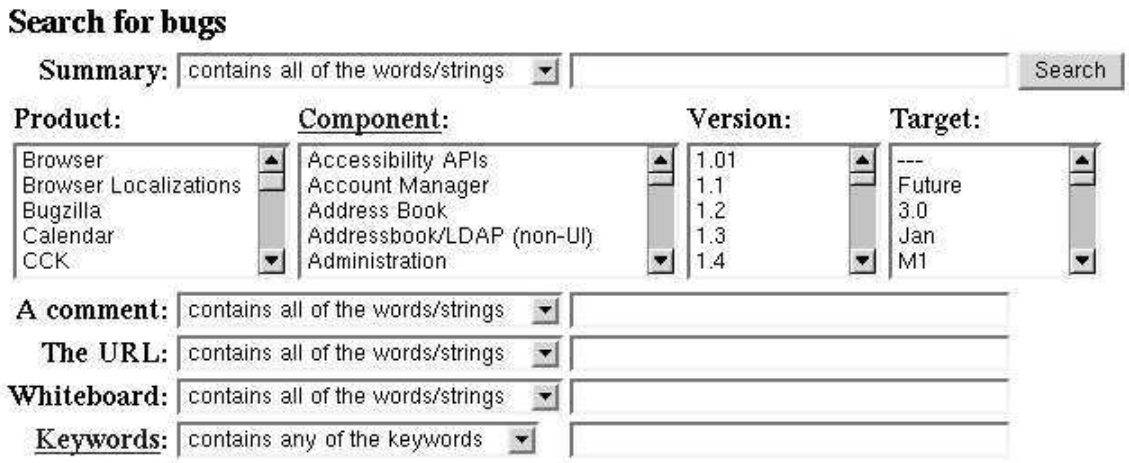

Figura 2: uma parta da interface de busca do Bugzilla.

3.2 Reprojeto da interface de listagem de dependências (bugs 83058 e 104340)

O Bugzilla possui uma interface que exibe os bugs de forma gráfica em uma árvore de dependências. A página original exibia listas de dependências de uma forma não-selecionável, o que ocasionava problemas quando visualizando árvores com grande profundidade, que ocorrem freqüentemente em projetos grandes. Foram implementadas alterações na barra de controle de visualização dos bugs para permitir exibição seletiva, com base em um projeto inicial de Andreas Franke. A nova interface está visível na figura 3.

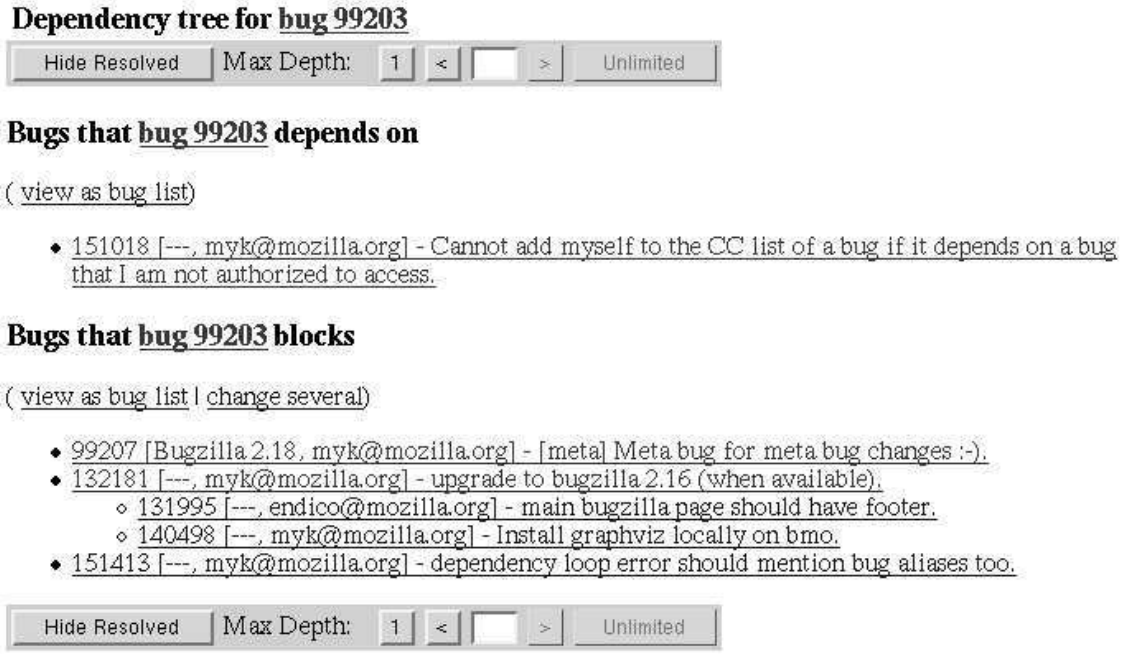

Figura 3: a nova interface de visualização de dependências.

3.3 Implementação de uma nova página inicial (bug 130835)

A página inicial do Bugzilla atualmente não oferece navegação consistente ou mesmo eficiente para o uso do site. Em conjunto com Matthew Thomas, projetista de interfaces para o Bugzilla e para o Mozilla, foi especificada uma nova página inicial que exibe listas de bugs relevantes para o usuário, e algumas estatísticas de uso da ferramenta. A nova página inclui também um mecanismo de navegação mais conveniente.

Este trabalho ainda está em desenvolvimento, embora a interface proposta já tenha sido anexada ao bug e aprovada pela equipe de revisores. A implementação com base na interface especificada já foi iniciada e está sendo revisada de acordo com o processo Bugzilla.

4 Conclusões

Bugzilla é uma ferramenta que cumpre sua função de forma completa: tem funcionalidade extensa, suporta diversos aspectos do processo de software, e está em desenvolvimento constante. Como parte deste trabalho, foram implementadas mudanças no software com grande benefício para um grupo numeroso de usuários. A participação no projeto suportou a pesquisa do processo de software do projeto Mozilla, que culminou em [2]. Finalmente, ofereceu uma oportunidade de interação com um projeto de software livre importante, aprofundando o conhecimento existente sobre seu processo.

Um dos objetivos principais da ferramenta Bugzilla é apoiar desenvolvimento geograficamente distribuído. Como já discutido na literatura, esta forma de organização da equipe oferece desafios particulares e requer ainda bastante estudo[4]. Entretanto, desenvolvimento distribuído oferece uma vantagem peculiar que é em geral ignorada: como toda comunicação é feita por escrito, é extremamente fácil capturar e armazenar o histórico da análise, projeto e implementação; basta oferecer um repositório e uma ferramenta integrada para seu uso. Bugzilla cumpre esta função, e é interessante observar como é, *de fato*, usado diariamente como a ferramenta principal de apoio ao desenvolvimento do Mozilla.

Finalmente, vale ressaltar que a base de código do Bugzilla oferece flexibilidade grande, e que a equipe de desenvolvimento é dedicada e eficaz. O projeto busca novos contribuidores, e é aberto a sugestões e implementação de alterações. Trabalhos futuros incluem integração maior com notificações de alteração CVS, maior facilidade de navegação (como extensão do trabalho da página inicial), e internacionalização.

Referências

[1] B. Berliner. CVS II: Parallelizing software development. In *Proceedings of the USENIX Winter 1990 Technical Conference*, pages 341–352, Berkeley, CA, 1990. USENIX Association.

[2] Christian Reis and Renata Pontin de Mattos Fortes. An Overview of the Software Process and Tools in the Mozilla Project. In *Proceedings of the Open Source Software Development Workshop*, pages 155–175, Newcastle Upon Tyne, United Kingdom, February 2002.

[3] Michael Fredericks. Using Defect Tracking and Analysis to Improve Software Quality. 1999. URL: Disponível online em `http://citeseer.nj.nec.com/fredericks99using.html`. Última visita em Junho de 2002.

[4] James D. Herbsleb and Rebecca E. Grinter. Splitting the organization and integrating the code: Conway's law revisited. In *Proceedings of ICSE*, pages 85–95, Los Angeles, May 1999. IEEECSP.

[5] B. P. Lientz, E. B. Swanson, and G. E. Tompkins. Characteristics of application software maintenance. *Communications of the ACM*, 21(6):466–471, 1978.

[6] J. Nielsen. *Designing Web Usability: The Practice of Simplicity*. New Riders Publishing, Indianapolis, IN, 2000.