Análise comparativa de IDEs de programação Java: IntelliJ e Eclipse

Daniel M. Assis

Resumo—A produtividade é um dos aspectos mais importantes no desenvolvimento de softwares corporativos. Em uma realidade onde as empresas precisam cada vez mais entregar software com qualidade em prazos cada vez menores, as IDEs são ferramentas das mais importantes, provendo o programador com recursos valiosos para otimizar suas atividades. Este artigo apresenta um comparativo de duas das IDEs Java mais utilizadas no mercado pela ótica da produtividade do programador.

Keywords—IDE, IntelliJ, Eclipse, Plugins, Produtividade, Java, Programação

1 Introdução

Cada vez mais, na indústria de software, é preciso entregar novas funcionalidades e correções em tempo hábil e competitivo, ao mesmo tempo que uma grande ênfase à qualidade precisa ser observada. Um dos grandes desafios dos programadores de software é, portanto, realizar entregas de forma produtiva. Neste ínterim, o trabalho aqui proposto situa o programador de software em termos de seu papel na produtividade do projeto, e então, após demonstrar que esta produtividade está diretamente ligada ao uso de ferramentas IDE[1], apresenta um estudo comparativo das duas IDEs Java mais utilizadas no mercado.

2 As ferramentas e a produtividade

A produtividade em programação refere-se a aspectos e metodologias de desenvolvimento de software que afetam a quantidade e qualidade do código-fonte produzido por um indivíduo ou time[2]. Existem vários aspectos a serem considerados, como quantidade de código, detecção e prevenção de erros, estimativa do custo de software, complexidade do sistema sendo construído.

Banker e Kauffman[3] sugerem que a medição de produtividade pode ser realizada por meio da relação entre o tamanho da aplicação e o trabalho consumido durante o desenvolvimento:

$$\text{Produtividade} = \frac{\text{Tamanho da aplicação desenvolvida}}{\text{Trabalho consumido durante desenvolvimento}}$$

Nesta relação, faz-se evidente a importância da otimização do trabalho a ser realizado pelo programador de software, com consequências diretas na produtividade. Para esta otimização do esforço, são utilizadas ferramentas profissionais e especializadas.

3 A IDE como ferramenta principal do programador

Dentre as ferramentas mais utilizadas pelo programador de software, a IDE consolida-se como a principal, de acordo com a pesquisa "Developer Productivity Report 2013"[4].

Uma IDE é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software, visando agilizar o processo de desenvolvimento[1]. Permite integrar diversas ações que são comuns aos desenvolvedores, como debug, edição, compilação, consulta à base de dados, dentre outras. Esta capacidade de reunir várias funcionalidades num único aplicativo é um aspecto de produtividade da IDE, contanto que as funcionalidades sejam consideradas úteis e satisfatórias pelos programadores. Há a possibilidade de uma dada IDE oferecer diversas

• *Daniel M. Assis (daniel.medeiros.assis@gmail.com)*

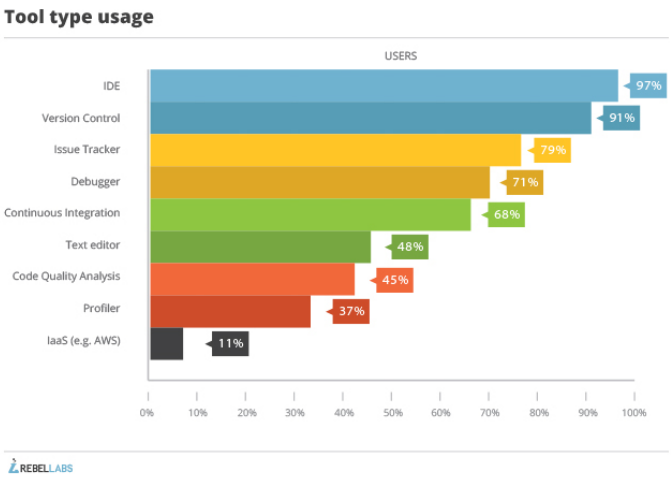

Figura 1: IDE é a ferramenta mais utilizada[4]

funcionalidades que não sejam relevantes, de forma que nem sempre a IDE com mais itens deve ser considerada a melhor em termos de produtividade.

As duas IDEs mais utilizadas no mercado Java atualmente, conquistando a grande maioria da preferência dos programadores, são *Eclipse* e *IntelliJ* [5].

Nesta mesma pesquisa, quando perguntados sobre *"qual IDE você gostaria de utilizar ou testar para desenvolvimento"*, a preferência foi para o IntelliJ (em relação a qualquer outra IDE) [5].

4 Análise Comparativa

Uma vez estabelecido o papel da ferramenta IDE como sendo essencial para a produtividade do programador, o próximo passo é selecionar alguns critérios de comparação entre as IDEs. Numa pesquisa envolvendo dez programadores de software nível sênior, foram citados quinze funcionalidades como sendo as mais úteis que uma IDE pode apresentar. Cada uma destas funcionalidades foi analisada em ambas as IDEs (em máquinas Windows 7 com 8GB de RAM, e tendo por base alguns projetos multi-módulo de pequeno a grande porte), e as informações foram consolidadas em uma tabela comparativa, relacionando as duas IDEs em relação a cada uma das funcionalidades em questão. Ao mesmo tempo, um descritivo detalhado foi dado, para explicar a forma pela qual cada funcionalidade foi avaliada.

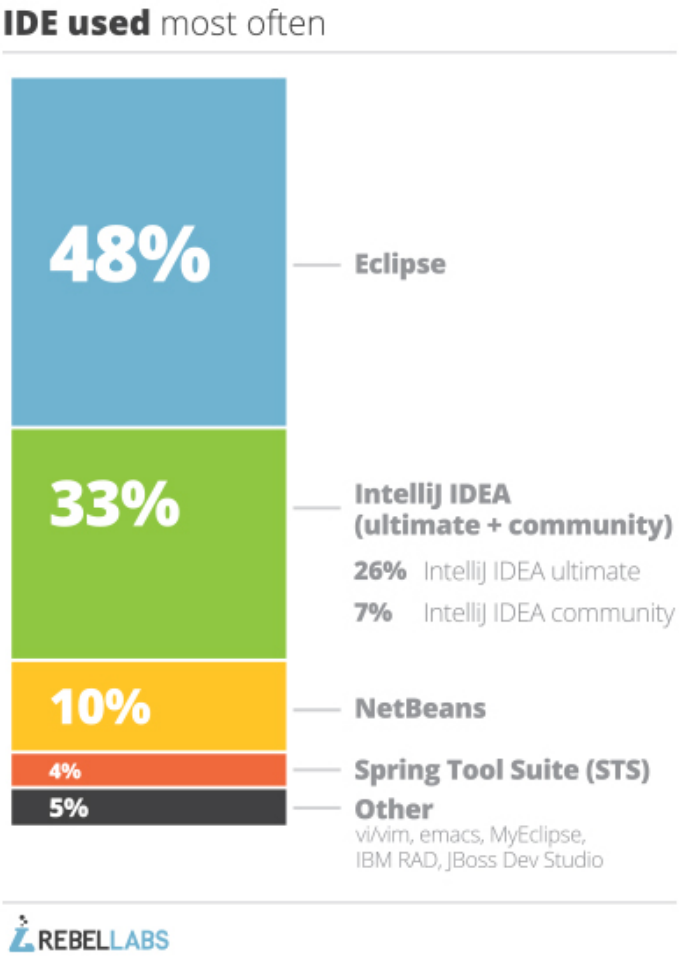

Figura 2: IDEs mais utilizadas [5]

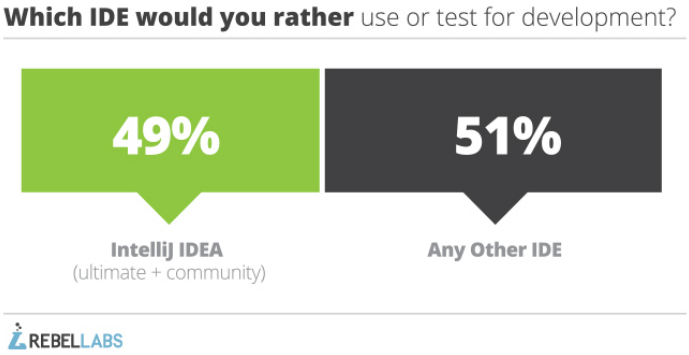

Figura 3: IDEs preferidas para uso ou teste [5]

4.1 Tabela Comparativa e Descritivo

Segue abaixo a tabela que consolida as quinze funcionalidades em relação às IDEs. Exatamente um valor 'X' e um valor '-' são aplicados para cada critério, sendo o 'X' o indicador de que uma dada IDE é superior à sua con-

corrente, e o '-' o indicador de que a dada IDE é inferior à sua concorrente. O julgamento foi aplicado considerando tanto a opinião dos entrevistados quanto a verificação do pesquisador.

Item	Eclipse	IntelliJ
Velocidade	–	X
Curva de Aprendizado	X	–
Suporte a Plugins	–	X
Estabilidade de Plugins	–	X
Estabilidade da IDE	–	X
Análise de Logs	–	X
Suporte à melhoria de qualidade	–	X
Diff de código	–	X
Integração com AppServer	–	X
Edição de Resource Bundles	–	X
Bind entre código Java e não-Java	–	X
Debug de JavaScript	–	X
Integração de Entidades com Bancos de Dados	–	X
Múltiplos projetos	X	–
Custo	X	–

Velocidade: relativo à quanto tempo uma IDE demora em atividades que bloqueiam o programador. Observou-se lentidão no Eclipse em vários momentos, em projetos grandes, por conta de indexações feitas pelo plugin do Maven, validação de arquivos JavaScript, e outras causas não-aparentes; o desligamento destas funcionalidade é possível, mas nem sempre efetiva, em contraposição ao IntelliJ, que não apresentou nenhum problema de lentidão no mesmo ambiente e sob os mesmos projetos. Outra questão é a que o Eclipse trava em alguns momentos por razões não aparentes, mantendo uma tela branca por vários segundos, sendo necessário finalizar o processo e abrir a IDE novamente, em alguns casos; em nenhum momento, o IntelliJ apresentou travamento. Também sugere-se[6] que a velocidade de Debug do IntelliJ é superior ao Eclipse.

Curva de Aprendizado: relativo a quanto esforço é gasto para aprender a IDE. O Eclipse ganha neste quesito, pelo fato de ser uma IDE mais conhecida pela maioria dos desenvolvedores Java. Ainda que programadores que utilizem IntelliJ sugiram que a curva de aprendizado é bem pequena, Eclipse tema vantagem de já ser mais conhecido.

Suporte a Plugins: relativo à quantidade e qualidade dos plugins disponíveis para a IDE. Eclipse e IntelliJ tem, ambos, um suporte muito grande a plugins. Contudo, programadores entendem que o IntelliJ tem um melhor suporte, pois é comum que plugins no Eclipse, quando em grande quantidade, tornem a IDE mais lenta, que ofereçam travamento e que não funcionem da forma esperada (como no caso do plugin m2eclipse, para integração do Eclipse com Maven, que é algo extremamente importante mas não funciona de forma adequada); o IntelliJ não apresentou estes problemas.

Estabilidade de Plugins: O IntelliJ ganha neste quesito pelo fato de a instalação de muitos plugins não ter ocasionado problemas de travamento ou lentidão perceptíveis. Tais problemas são comumente encontrados no Eclipse.

Estabilidade da IDE: Pelo fato de ser comum no Eclipse o travamento e lentidão por razões diversas (impactando consideravelmente em produtividade), e por tais problemas não terem sido observados no IntelliJ em relação ao mesmo ambiente e aos mesmos projetos, o IntelliJ ganha neste quesito.

Análise de Logs: O IntelliJ possui uma feature nativa que permite que um StackTrace seja copiado de um arquivo de Log e colado dentro da IDE, e então as classes que aparecem neste Log são diretamente mapeadas para classes do projeto aberto na IDE, de forma que basta clicar sobre a classe no log para a IDE abrir a classe Java diretamente na linha de código que apresenta problema. Esta funcionalidade é considerada extremamente útil pelos programadores, em especial em tarefas que envolvam rapidez em resolução, como atendimento a correção de bugs. O Eclipse não possui esta feature.

Suporte à melhoria de Qualidade: relativo ao suporte da IDE para refatoração, testes automatizados e análise estática de código. Em relação à refatoração, observou-se que o IntelliJ é mais inteligente do que o Eclipse, no sentido que é mais capaz de inferir o que o programador quer, oferecendo opções mais adequadas.[7] Sobre testes, IntelliJ possui a vantagem de autodetectar o tipo de framework

(JUnit, TestNG) sendo utilizado e, a partir disto, executar propriamente (Eclipse precisa que a execução seja feita pelo plugin correspondente). Sobre análise estática de código, tanto IntelliJ quanto Eclipse possuem bons plugins para testes, mas IntelliJ sai na frente por possuir um plugin capaz de integrar FindBugs, PMD e Checkstyle, permitindo uma forma única de validar o código e importar regras de validação de ferramentas como o Sonar. O Eclipse lida com os plugins citados independentemente, de forma que a validação do código e a importação de regras faz-se mais trabalhosa. IntelliJ ainda dá feedback instantâneo, enquanto se digita código, através de warnings; no Eclipse, é preciso executar a análise manualmente (alguns plugins permitem a execução automática, mas só depois da classe ter sido salva).[6].

Diff de código: O IntelliJ possui a capacidade de exibir, no mesmo momento em que uma classe é modificada, qual era o estado anterior. Isto fica visível de forma muito simples: basta o programador colocar o mouse sobre uma barra lateral da classe para saber qual o estado anterior de linhas alteradas. Esta feature é considerada muito útil pelos desenvolvedores, que não se sentem intimidados pela quantidade de impacto que sua alteração está realizando. Além disto, também é possível, no IntelliJ, visualizar apenas os arquivos que foram alterados, e agrupar estes arquivos por assunto, o que facilita enormemente a realização de commits. No Eclipse, estas features não estão disponíveis.

Integração com AppServer: relativo à capacidade da IDE de executar e testar a aplicação no servidor de aplicação, sem necessidade de realização do trabalho de forma externa à IDE. O Eclipse possui plugin para este trabalho, contudo, apresenta problemas, e seu uso é desencorajado pelos programadores, que preferem trabalhar com o Application Server por fora da IDE. Já no IntelliJ, as integrações com servidores de integração não apresentaram problemas, e funcionaram satisfatoriamente, sem ocasionar lentidão. Desta forma, o IntelliJ ganha neste quesito. Os Application Servers utilizados no teste foram Websphere e JBoss.

Edição de Resource Bundles: relativo à capacidade de editar arquivos de propriedades da aplicação. No Eclipse, a edição é feita por meio de edição de texto simples. Já o IntelliJ oferece várias features para edição de arquivos deste tipo, tais como agrupamento de bundles por país, realização automática de Unicode e code complete, exibição de propriedades não sendo usadas na aplicação por meio de uma cor diferente que destaca-as das demais. Por isto, o IntelliJ ganhou neste quesito.

Bind entre código Java e não-Java: relativo à capacidade de relacionar classes java com recursos não-Java. IntelliJ possui tal integração, sendo capaz de relacionar classes Java diretamente com outros tipos de arquivo (como JSP, por exemplo) que os referenciam. Este é um recurso muito prático para desenvolvimento de objetos Java em relação à camada de apresentação (e, inclusive, existe um grande suporte à sintaxe em diferentes linguagens, como Javascript, HTML, queries JPA, etc[6]). Esta feature existe de forma mais limitada no Eclipse.

Debug de JavaScript: relativo à capacidade de Debugar arquivos JavaScript. Com IntelliJ, é possível realizar Debug de Javascript utilizando Chrome.

Integração de Entidades com Banco de Dados: relativo ao bind entre entidades mapeadas por framework Objeto-Relacional e as tabelas de banco de dados que são mapeadas. O IntelliJ possui um recurso de autocomplete que permite, na edição da entidade, visualizar as colunas existentes na tabela, de forma a facilitar o trabalho de mapeamento. Tal recurso não encontra-se disponível no Eclipse.

Múltiplos Projetos: Ainda que IntelliJ tenha a capacidade de abrir vários projetos (por meio do plugin do Maven), esta funcionalidade não é intuitiva. Já o Eclipse permite a abertura de projetos em ”Workspaces”, e cada Workspace ainda pode ter ”Working Sets”, para agrupamento dos projetos. Eclipse é melhor neste quesito.

Custo: O Eclipse é uma ferramenta gratuita; já o Intellij é pago, com o valor da licença em 199 dólares para individual e 499 dólares para empresa (dados de setembro/2014). Eclipse ganha neste quesito.

5 Relatório de produtividade com IntelliJ

Uma feature nativa bastante interessante do IntelliJ é a possibilidade de medir a produtividade. O programador pode visualizar como está utilizando a IDE, de forma a poder melhorar sua própria produtividade pela otimização do uso da ferramenta. IntelliJ mantém um histórico de todas as ações e coleta as estatísticas, que são convertidas num relatório para análise. [8]

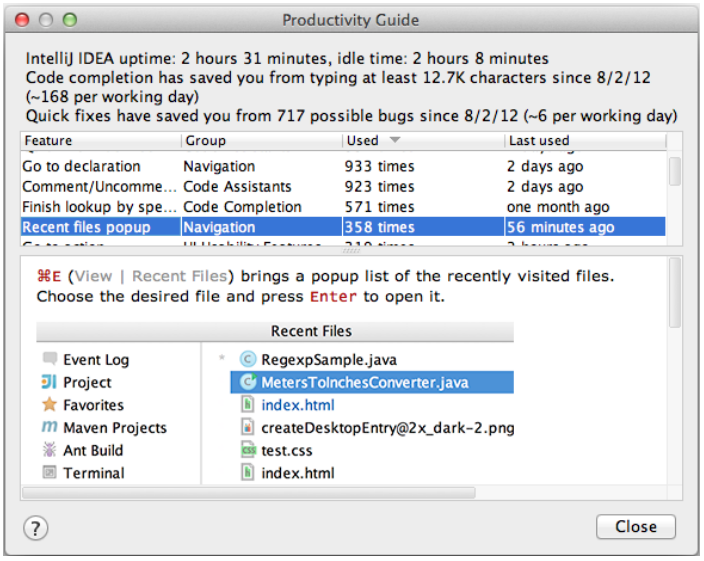

Figura 4: Relatório de Produtividade do IntelliJ [8]

6 Considerações Finais

Quando sugere-se que uma determinada feature não existe em uma IDE, parte-se do pressuposto que não existe a funcionalidade nativa e nem um plugin que a implemente e que funcione de forma satisfatória. Esta definição partiu do conhecimento dos entrevistados, que são programadores que atuam ou já atuaram com Eclipse e IntelliJ, bem como de pesquisas realizadas online.

Outros aspectos ficaram de fora da pesquisa mas que merecem destaque são: detecção automática de frameworks (facets automáticos), autosave, falta de perspectives, chained completion, live templates.[8]

7 Conclusão

Este artigo demonstrou a importância da IDE como ferramenta de programação de software dentro do contexto de produtividade e, a partir desta definição, realizou uma análise comparativa de duas das IDEs preferidas do mercado. A premissa é que uma boa IDE faz o programador mais produtivo, contudo o conceito é subjetivo na medida em que a produtividade depende das funcionalidades que o programador quer e precisa[8] - e, por isto, os critérios utilizados na análise partiram de uma pesquisa envolvendo programadores.

Considerando tais critérios, observou-se que, a despeito da popularidade, o Eclipse ficou atrás do IntelliJ. Sob a ótica dos plugins, ainda que possa ser sugerido que o Eclipse, com um conjunto bem específico de plugins, possa equiparar-se ao IntelliJ em relação aos critérios utilizados, seria uma afirmativa subjetiva sem que houvesse a enumeração de todos estes plugins e seu devido teste, e ainda deveria ser considerado o fato de que o Eclipse apresenta instabilidade quando do uso de determinados plugins. Além disto, aspectos como ”Estabilidade da IDE”e ”Velocidade”não podem ser equiparados, visto que não se trata de uma opção para o desenvolvedor, e sim um design da própria ferramenta.

Referências

[1] “Ambiente de desenvolvimento integrado,” http://pt.wikipedia.org/wiki/Ambiente_de_desenvolvimento_integ

[2] “Programming productivity,” http://en.wikipedia.org/wiki/Programming_productivity.

[3] G. P. Sudhakar, A. Farooq, and S. Patnaik, “Measuring productivity of software development teams,” 2011, http://www.sjm06.com/SJM ISSN1452-4864/7_1_2012_May_1_170/7_1_65-75.pdf.

[4] “Developer productivity report 2013 - how engineering tools and practices impact software quality and delivery,” http://zeroturnaround.com/rebellabs/developer-productivity-report-2013-how-engineering-tools-practices-impact-software-quality-delivery/9/.

[5] “Java tools and technologies landscape for 2014,” http://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-for-2014/6/.

[6] “Why we dropped eclipse in favour of intellij,” https://plumbr.eu/blog/why-we-dropped-eclipse-in-favour-of-intellij.

[7] “Why dea is better than eclipse,” http://java.dzone.com/articles/why-idea-better-eclipse.

[8] "Getting started with intellij as an eclipse user," http://zeroturnaround.com/rebellabs/getting-started-with-intellij-idea-as-an-eclipse-user/.