Defeitos de Software: Estudo, Caracterização e Gestão Automatizada

Autores

Thiago de Andrade Gallo

Orientador

Plinio Roberto Souza Vilela

Apoio Financeiro

Pibic

1. Introdução

Este trabalho busca algumas respostas sobre a relação entre fatores humanos envolvidos no processo de desenvolvimento de software que possam vir a ter algum impacto em sua qualidade. Através de revisões bibliográficas (JURISTO; MORENO, 2003) foram procurados alguns parâmetros nessa linha, porém a informação é muito escassa, o que veio a causar uma dúvida de porque esses fatores são pouco conhecidos. Deu-se então a elaboração detalhada de um projeto de experimento seguindo as recomendações de experimentação em Engenharia de Software (WOHLIN et al, 2000), visando explorar algumas hipóteses, relacionadas à experiência de programadores e a forma com que produzem código, e se isso pode ter alguma influência na incidência de defeitos no software final. Este experimento foi pensado para responder algumas questões iniciais e tornar possível a especificação de uma ferramenta de coleta de analise automatizada para experimentos similares no futuro. A principal hipótese era a de que a expressão facial de programadores teria uma tendência a se modificar quando o programador estivesse trabalhando em partes mais complicadas do código, o que poderia criar uma correlação entre a expressão facial e uma maior probabilidade de ocorrência de defeitos no software. Após a análise dos dados experimentais foi possível perceber que a experiência contribui para um desenvolvimento mais rápido e planejado de código, mesmo que para problemas pequenos e relativamente simples como os propostos. Não foi possível evidenciar se as expressões faciais têm realmente uma alteração significativa nos momentos de inserção de defeitos no software, porém ficou claro que alguns programadores apresentam uma alteração na expressão quando cometem enganos e quando se concentram em certas partes do problema.

2. Objetivos

Este trabalho visa extrair experimentalmente os parâmetros gerais pertinentes à construção de uma ferramenta automatizada para análise da relação Desenvolvimento de Código versus Tempo . Tal ferramenta será utilizada para rastrear as alterações em um código fonte ao longo de uma linha do tempo em que este foi desenvolvido, a fim de determinar uma relação entre os padrões de programação do desenvolvedor e a incidência de defeitos de software. Em um futuro experimento tal ferramenta possibilitará uma amostragem maior de programadores, onde diversas variáveis poderão ser analisadas, de forma a caracterizar fatores psicológicos e ambientais de influência significativa na incidência de defeitos. Os requisitos para a construção da ferramenta serão levantados através da observação dos hábitos de produção de diferentes programadores em diversos níveis de experiência. Desta forma este trabalho se trata de um experimento exploratório, em busca das características mais marcantes no processo de resolução de um problema computacional, bem como traços comportamentais comuns aos programadores, por exemplo, a freqüência de utilização do mouse ou teclado para determinadas tarefas. Com a análise de um grupo de

programadores dividido em níveis de experiência, busca-se também como objetivo secundário verificar a relação entre os defeitos no software final e as expressões faciais dos programadores quando produzem esses defeitos. Os defeitos, rastreados na análise dos programas produzidos, serão utilizados para gerar marcos ao longo da linha do tempo do experimento, nestes marcos as expressões faciais dos programadores serão analisadas, buscando evidenciar uma relação entre o momento em que o engano foi cometido e a expressão do individuo naquele determinado momento.

3. Desenvolvimento

Introdução Teórica Definições e Verdades Estabelecidas Para a realização deste trabalho algumas variáveis foram pré-estabelecidas, a fim de focar a análise dos dados nos parâmetros mais cruciais. Durante o trabalho estão expostos alguns conceitos, importantes para a compreensão do processo de teste de software, estes são definidos aqui. Todos os programadores utilizaram o ambiente com o qual estão familiarizados (como exposto no momento da entrevista), de forma que toda a atenção do programador esteja voltada para a resolução do problema. Desta forma espera-se que os programadores alcancem sua plena capacidade, não sendo distraídos pelo ambiente de programação. As características de personalidade de cada programador não foram levadas em consideração para a análise dos vídeos produzidos e suas expressões faciais. Ou seja, não foi feita uma análise prévia da irritabilidade do sujeito, ou quão este é suscetível à pressão. Como um perfil psicológico profundo não está em questão, apenas traços gerais foram considerados na análise facial. Segundo o IEEE Standard Computer Dictionary (1990), no processo de teste de software existe uma distinção entre uma ação humana (engano), sua manifestação (um defeito de software ou hardware), o resultado do defeito (uma falha), e quanto o resultado esta incorreto (o erro). A seguir temos as definições destes termos importantes para este trabalho. Falha A inabilidade de um sistema ou componente de desempenhar suas funções dentre os requisitos de performance. Um resultado incorreto. Por exemplo, um valor computado de 12 quando o resultado correto seria 10. Erro A diferença entre um valor ou condição computado, observado ou medido e o verdadeiro, especificado ou térico, valor ou condição correto. Por exemplo, uma diferença de 30 metros entre o resultado computado e o resultado correto. The difference between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. For example, a difference of 30 meters between a computed result and the correct result. Defeito Um passo, processo, ou definição de dados incorreto. Por exemplo, uma instrução incorreta em um programa de computador. Engano Uma ação humana que produz um resultado incorreto. Por exemplo, uma ação incorreta por parte do programador ou do operador. Objeto do Estudo Como objeto de estudo para este trabalho foram analisados alguns fatores humanos que influenciam no processo de desenvolvimento de software, são eles: os hábitos de desenvolvimento e produção de código, as expressões faciais dos indivíduos enquanto programam, e sua suscetibilidade à limitação de tempo para realização de uma tarefa. Hipóteses Testadas Como exposto anteriormente, o objetivo primário deste trabalho é exploratório, portanto foram procurados padrões comportamentais na produção de código fonte, a fim de determinar os requisitos para a construção de uma ferramenta de apoio à análise da atividade de programação. Alguns pontos específicos são observados: a freqüência com que os programadores tendem a compilar o código; a taxa de utilização do mouse e teclado para desempenhar as tarefas mais comuns, observando qual tem maior prioridade; o estilo de produção de algoritmos para resolução dos problemas propostos, identificando se algum método é utilizado preferencialmente para resolução de problemas do mesmo tipo; observação da quantidade de vezes que o programador altera cada bloco do programa, e sua relação com os defeitos encontrados na versão final. É esperado que o comportamento dos indivíduos varie em relação ao seu nível de experiência para cada um desses itens, apresentando um padrão mais consistente entre os voluntários mais experientes. Além destes pontos específicos, os resultados foram analisados em busca de padrões e comportamentos não previstos inicialmente, que possam ser de alguma forma pertinentes à questão. Em segundo plano temos a hipótese de que ao cometer enganos, inserindo defeitos no software, o programador apresenta uma expressão facial diferenciada do seu padrão, o que possibilitaria rastrear regiões passíveis de conter defeitos através da análise facial do programador enquanto produzia o código. Neste momento não se busca caracterizar os possíveis motivos que causariam esta mudança na expressão, tampouco categorizar as expressões possíveis, e sim, simplesmente verificar a existência desta relação. Inicialmente foi pensado definir dois grupos, um com limitação de tempo para a resolução dos problemas, e outro sem essa limitação, e em cada

etapa esses grupos seriam trocados, de forma a cada voluntário passar pôr ambas as situações. Esta metodologia foi pensada para avaliar o impacto que um tempo pré-definido para a resolução de um problema pode ter sobre o tempo efetivo necessário para a sua conclusão. Porém mais tarde foi percebido que esta questão não poderia ser avaliada de forma satisfatória nesse experimento, pôr ter um grupo de candidatos relativamente pequeno; além disso, alguns voluntários poderiam não conseguir resolver os problemas no tempo determinado, o que seria bom para esta análise do fator tempo, porém reduziria os dados para as outras análises (como a quantidade de defeitos no programa final entregue), desta forma foi decidido pela retirada desta variável. Metodologia População Estudada A população tomada para este experimento consiste em um grupo de seis programadores. Classificados em três níveis de experiência (iniciante, intermediário e avançado) contendo dois programadores de cada nível. Os indivíduos classificados como iniciantes são aqueles com pouca experiência prática em programação, mas capazes de resolver os problemas mais simples propostos. O grupo intermediário foi constituído por pessoas com certa experiência prática em programação e desenvolvimento de software, porém sem nenhuma (ou pouca) experiência profissional na área. No grupo avançado se encontrão os desenvolvedores profissionais, que atuam na área, ou então professores com extensa experiência em desenvolvimento. A distribuição dos indivíduos do mesmo nível foi feita de forma mais homogênea possível, procurando a seleção de programadores com habilidades equivalentes dentre os membros do seu grupo. Seleção dos Candidatos Inicialmente um grupo de programadores foi convidado a participar voluntariamente do experimento. Em seguida os candidatos foram entrevistados a fim de determinar algumas variáveis. As entrevistas ocorreram individualmente, elas foram filmadas, os pontos-chave anotados. As entrevistas ocorreram de forma mais informal possível, procurando deixar todos os candidatos confortáveis ao responderem as questões, seguindo como guia algumas perguntas pré-definidas (Anexo 1). A importância da participação do voluntário para o experimento foi explicada, assim como as implicações deste desistir no decorrer do processo. Através das perguntas se tornou possível determinar o ambiente de desenvolvimento com que cada programador estava familiarizado; os candidatos foram classificados dentre os níveis de experiência, e um perfil simples de expressões faciais foi traçado, para ser utilizado posteriormente como parâmetro na análise dos resultados. Problemas a Serem Aplicados Cada indivíduo participante do teste teve 3 problemas para resolver em cada etapa, classificados como trivial, moderado e complexo. Os problemas foram previamente testados para checar sua dificuldade. Aplicação dos Problemas A aplicação dos problemas ocorreu em duas etapas, que se deu em dias distintos, separados por o maior período de tempo possível, em média uma semana. Todos os testes foram efetuados no período da tarde, entre as 13 e 19 horas. Os voluntários estavam sozinhos na sala onde o teste foi aplicado, e foram sendo filmados por uma câmera focada em seus rostos (câmera 1)1 e um software de gravação do desktop2 (câmera 2). Os voluntários estavam livres para sair a qualquer momento, para satisfazer necessidades pessoais ou relaxar. Metodologia de Análise dos Resultados A Primeira etapa foi identificar a existência de defeitos nos programas produzidos pelos voluntários. Estes defeitos foram documentados em um registro, como o código do programador3, o número do problema referente ao programa em que o defeito foi encontrado e o erro gerado pelo defeito. Uma busca nas gravações (câmera 2) da aplicação dos testes será feita, buscando todos os momentos em que a parte defeituosa do programa foi escrita ou editada, e esses tempos serão incluídos no registro. É esperado identificar se a quantidade de vezes em que o programador alterou o código tem alguma relação com o defeito na versão final. A segunda etapa será a análise das expressões faciais do programador nos momentos documentados no registro (câmera 1), e sua comparação com as demais expressões presentes em outros momentos do teste e na entrevista. Uma relação entre os defeitos e variações nas expressões tentará ser estabelecida. A terceira etapa é a comparação dos tempos necessários para resolução dos problemas. São feitas as seguintes relações: tempo necessário para resolução de um problema de dificuldade X, no dia 1 e no dia 2, pelo mesmo programador; tempo necessário para resolução de um problema X por programadores do mesmo nível, fazendo uma relação entre os dois subgrupos; tempo necessário para resolução de um problema X por programadores de diferentes níveis, dentro de um mesmo subgrupo. Na quarta etapa os vídeos são separados em grupos de acordo com o nível de experiência dos voluntários a que correspondem, e observados em seqüência, primeiro todos os do dia 1, depois todos os do dia 2. Nesta etapa são observadas as características pessoais de cada programador que se repetem dentro do grupo, documentando a freqüência com que determinadas práticas e padrões aparecem. Supõe-se que programadores experientes altamente familiarizados com um determinado ambiente de programação

tendem a fazer extenso uso de atalhos de teclado para realizar a maior parte das tarefas.

4. Resultados

Os candidatos selecionados para o experimento se encontram na faixa etária de 18 a 30 anos, todos com formação acadêmica na área de computação (alguns estão cursando e outros já concluíram). Tanto os iniciantes como os intermediários estavam ainda em seu curso universitário e tinham entre 1 a 3 anos de experiência com programação, todos declararam ter algum tipo de experiência com a linguagem C e 75% deles trabalham além de estudar. Os voluntários classificados como experientes tem ambos mais de 15 anos de experiência com programação e trabalham na área de desenvolvimento e pesquisa de software. Após a realização do experimento ficou claro que para se ter bons dados em relação aos defeitos em software é preciso ter uma amostra bem grande, por parte de cada programador, e também um grupo grande de voluntários, pois como aconteceu, poucos defeitos foram produzidos, já que os problemas eram relativamente simples, de forma a poderem ser resolvidos por todos em tempo hábil. Então tornou-se muito difícil fazer uma análise direta como tinha sido pensado a princípio: fazendo uma relação de uma parte defeituosa do código com o exato momento em que o programador a produziu. A tabela 1 mostra que nenhum programa finalizado teve mais do que um defeito documentado, dentre os 36 programas produzidos, apenas cinco apresentaram defeitos. No anexo 6 estão disponíveis as fichas utilizadas para cadastrar os defeitos, seguindo o modelo do projeto do experimento. Apesar de as expressões faciais parecerem mostrar claramente os momentos em que alguns voluntários tiveram mais dificuldades para resolver os problemas, os dados são insuficientes para fazer uma relação direta com a incidência de defeitos em software. Isso porque não foi possível encontrar defeitos do tipo esperado nesse experimento: ou os candidatos conseguiram resolver o problema, ou não conseguiram. Nos casos em que os problemas não foram resolvidos, o código disponível era insuficiente para ser considerado. Nos casos em que o problema foi resolvido os defeitos encontrados foram gerais ao algoritmo, de forma que não foi possível isolar os blocos de código afetados por esses defeitos. Um resultado que já era esperado e pode ser observado é que em geral os programadores mais experientes resolvem os problemas mais rápido, como mostrado no gráfico 3. Ambos os candidatos do nível experiente escreveram um algoritmo em pseudocódigo antes de implementar o código, o que acabou reduzindo o tempo de codificação. Em contrapartida os outros candidatos escreveram algoritmos em pseudocódigo somente para os problemas mais difíceis, sendo ainda que alguns só o fizeram após tentar codificar diretamente e fracassar. A maior experiência aliada ao fato de escrever em pseudocódigo antes de codificar mostrou uma outra tendência: os programadores mais experientes costumam compilar menos o código. Os voluntários experientes compilaram o programa somente quando acreditavam que ele estava pronto, ou então em pontos que um teste seria importante, e só voltaram a compilar novamente quando julgaram ter corrigido todos os erros mostrados pelo compilador, além de checarem visualmente o próprio código e depurarem as partes suspeitas. Nesse sentido os programadores tem uma abordagem mais orientada a tentativa e erro, eles utilizam o recurso de compilação para testar o código conforme vão desenvolvendo, raramente utilizam o recurso do debugger para isso. Outro fato presente é que a cada erro corrigido uma nova compilação é feita para testar se o problema foi resolvido. A quantidade em que programa foi compilado pelos voluntários está demonstrada no gráfico 2 Apesar de não poder ser explicita em gráficos e tabelas, ao assistir os vídeos foi possível perceber algumas relações interessantes. De fato como pensando inicialmente os programadores mais experientes utilizam mais o teclado do que o mouse para se deslocar dentro do código e para acionar as funções mais comuns da IDE. Neste experimento o mouse praticamente não foi utilizado dentro da IDE pelos programadores intermediários e experientes, e mesmo entre os iniciantes ele foi pouco utilizado em relação ao que se esperava. Isto demonstra que a familiarização com a IDE é de fato um fator que pode diminuir a utilização do mouse, e essa familiarização não demanda anos de prática, já que mesmo os iniciantes apresentaram um certo grau de familiaridade.

5. Considerações Finais

Atualmente os dados sobre aspectos humanos que afetam o processo de desenvolvimento de software são bem escassos, muitas pesquisas nessa área ainda estão em desenvolvimento, e nenhum conhecimento sólido esta estabelecido. Sem dúvida é muito importante conhecer os fatores psicológicos e comportamentais que podem ser trabalhados para aumentar a produtividade de times de desenvolvimento, por isso pesquisas nessa área são válidas. É importante preestabelecer um projeto sólido com rigor

científico, para que os dados tenham validade. Com isso em vista esse trabalho se concentrou bastante em desenvolver um bom projeto de experimento que pudesse ser reproduzido posteriormente pôr outras pesquisas. Para poder desenvolver um bom projeto um estudo prévio sobre metodologia científica e estudos empíricos em engenharia de software foi realizado. Através do experimento foi possível perceber que os programadores que se encontram no meio acadêmico tem um forte contato com a linguagem de programação C, e que muitos deles já estão atuando no mercado de trabalho mesmo enquanto ainda estão concluíndo os seus cursos universitários. A questão de se um determinado prazo pode afetar a produtividade de um programador foi respondida de forma heterogênea pelos voluntários, não demonstrando nenhuma tendência específica. Outras entrevistas informais com colegas, entretanto, demonstraram que todos acreditam que o fator tempo tenha alguma influência, mesmo que muito pequena para alguns. Um estudo aprofundado com uma grande população seria interessante para desvendar essa questão. Os resultados mais interessantes vieram da análise do comportamento dos programadores experientes versus os iniciantes/intermediários. Os programadores experientes tendem a desenvolver o programa com uma metodologia mais formal e estruturada, organizando o pensamento e construíndo pseudocódigo antes de codificar o programa em si, utilizando ferramentas de depuração e analisando o log de erros do compilador. Isso leva a um desenvolvimento mais ágil em termos de tempo, menos compilações do código durante o desenvolvimento e, em geral, um código mais bem escrito e estruturado. Além disso, os mais experientes preferem utilizar os atalhos de teclado presentes na IDE ao invés do mouse, os iniciantes e intermediários também demonstraram esse comportamento, e certamente também estão familiarizados com as IDEs que escolheram, porém a freqüência de utilização do mouse ainda é maior entre eles se comparados ao voluntários experientes.

Referências Bibliográficas

IEEE STANDARD COMPUTER DICTIONARY: A Compilation of IEEE Standard Computer Glossaries 610. USA: IEEE, 1990. JURISTO, N.; MORENO, A. M. (Org.). Lecture Notes on Empirical Software Engeneering. Singapura: World Scientific Publishing, 2003. TechSmith Corporation. Camtasia Studio 3.0.0. Okemos, MI, c 1999-2005. Disponível em: http://www.techsmith.com/camtasia.asp. Software de captura do desktop. WOHLIN, C et al. Experimental Software Engeneering: An Introduction. Kluwer Academic Publishers, 2000.

Anexos

Gráfico 1. Tempo de desenvolvimento do código por programa, para todos os voluntários.

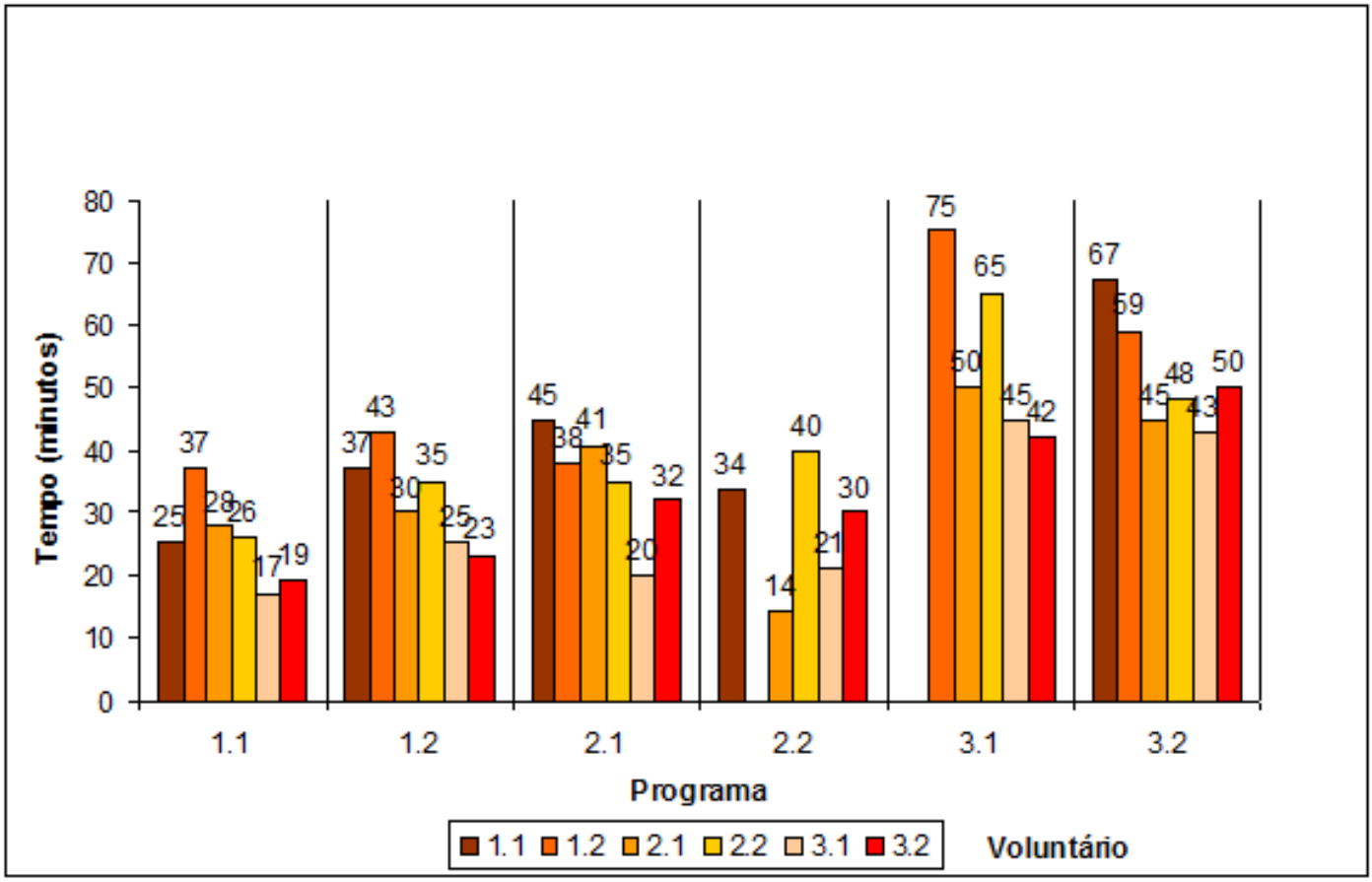

Gráfico 2. Quantidades de compilação do código por programa, para todos os voluntários.

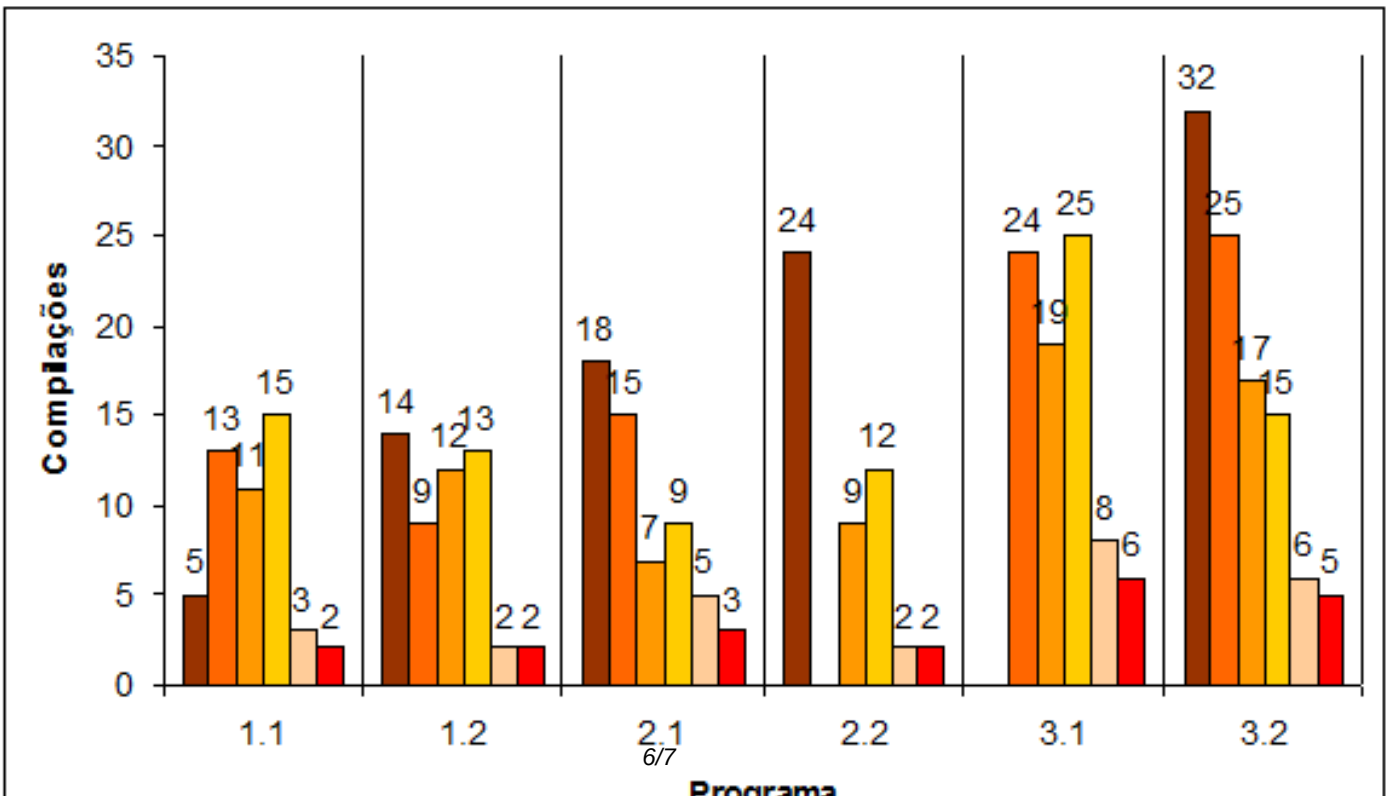

Tabela 1. Quantidades de defeitos encontrados nos programas.

	Problema					
Programador	1.1	1.2	2.1	2.2	3.1	3.2
1.1	1	0	0	0		0
1.2	0	1	0		1	0
2.1	0	0	0	0	1	0
2.2	0	1	0	0	0	0
3.1	0	0	0	0	0	0
3.2	0	0	0	0	0	0

Tabela 2. Tempo em minutos necessário para o desenvolvimento do programa.

	Problema						
Programador	1.1	1.2	2.1	2.2	3.1	3.2	Total (minutos)
1.1	25	37	45	34		67	208
1.2	37	43	38		75	59	252
2.1	28	30	41	14	50	45	208
2.2	26	35	35	40	65	48	249
3.1	17	25	20	21	45	43	171
3.2	19	23	32	30	42	50	196

Tabela 3. Quantidades de compilação do código antes da finalização do programa.

	Problema					
Programador	1.1	1.2	2.1	2.2	3.1	3.2
1.1	5	14	18	24		32
1.2	13	9	15		24	25
2.1	11	12	7	9	19	17
2.2	15	13	9	12	25	15
3.1	3	2	5	2	8	6
3.2	2	2	3	2	6	5