# Alinhando Perspectivas de Qualidade em Código Fonte a Partir de Estudos Experimentais - Um Caso na Indústria

**Talita V. Ribeiro[1], Guilherme H. Travassos[1]**

[1]COPPE/UFRJ – Universidade Federal do Rio de Janeiro (UFRJ)
Caixa Postal 68511– CEP 21.945-970 – Rio de Janeiro – RJ – Brasil

{tvribeiro,ght}@cos.ufrj.br

***Abstract.*** *Rework activities consume substantial amounts of a software development project's budget, however some rework could be avoided. This paper presents an overview of the research strategy and the development activities used to support a medium software development company on reducing risks of rework activities. By following a research strategy based on experimental studies, a set of evidence-based coding guidelines for readability and understandability have been proposed. Such guidelines are intended to align the different developers' source code quality perspectives, and, therefore, reduce the risks of source code reconstruction during software maintenance activities.*

***Resumo.*** *Atividades de retrabalho são responsáveis por grande parte dos custos de um projeto de desenvolvimento de software, sendo que parte poderia ser evitada. Este artigo apresenta a estratégia de pesquisa, guiada por diferentes estudos experimentais, utilizada para auxiliar uma empresa de desenvolvimento de software de médio porte a reduzir riscos com atividades de retrabalho. Diretrizes de codificação para legibilidade e compreensibilidade baseadas em evidência e adequadas para o contexto organizacional foram propostas como medida para alinhar diferentes perspectivas de qualidade em código fonte e, assim, reduzir o risco de reconstrução de código fonte na organização, sem, contudo, deixar de serem aplicáveis a outros contextos organizacionais.*

## 1. Introdução

Atividades de retrabalho são transversais aos processos de construção e manutenção de software. Estas atividades estão relacionadas principalmente com solicitações de clientes para correção e/ou evolução de regras de negócio e requisitos (GOPAL, MUKHOPADHYAY e KRISHNAN, 2002). Retrabalhar significa reinvestir esforço e, por conseguinte, custo adicional para atividades que já foram realizadas, seja para corrigir problemas ou ajustar o software (CASS, JR. e OSTERWEIL, 2003). De 40 a 50% do esforço dos projetos de desenvolvimento são gastos em retrabalho que poderia ser evitado (BOEHM, ROMBACH e ZELKOWITZ, 2005), tais como: escolha inadequada da abordagem de desenvolvimento de software (com mais ou menos interação com o cliente) e a falta de sistemática durante o planejamento e execução dos testes que acarreta em mais de 70% de retrabalho nesses tipos de atividades (BOEHM, 2006).

Conroy e Kruchten (2012) argumentam que o retrabalho é um problema recorrente durante o desenvolvimento de software, porque aspectos humanos envolvidos nesse processo são comumente negligenciados para a análise de suas causas. Exemplos são as diferentes inferências que os desenvolvedores podem fazer sobre o domínio do problema do software ou ainda as diferentes perspectivas de qualidade que cada um pode ter para o

software a ser construído; fatores que podem desencadear um desalinhamento conceitual não somente entre equipe de desenvolvimento e *stakeholders*, mas também entre membros de uma mesma equipe de desenvolvimento, provocando retrabalho e até mesmo comprometendo o sucesso do projeto.

Nesse contexto de atividades que poderiam ser evitadas e que se relacionam com aspectos humanos é que a prática de reconstrução de código fonte foi identificada como retrabalho em uma organização de desenvolvimento de software embarcado (Empresa Alfa) – contexto em que a pesquisa, objeto desta dissertação de mestrado, foi desenvolvida. Entendemos como sendo reconstrução de código fonte a atividade de reescrever o código de modo não sistemático para atender a um objetivo de qualidade particular, ou seja, que não necessariamente é conhecido ou pré-definido pela organização e que não é consenso entre os diferentes participantes de uma mesma equipe de desenvolvimento. Essa característica de atender a um objetivo de qualidade individual e específico pode acarretar em sucessivas realizações de reconstrução de um mesmo código fonte por diferentes programadores em diferentes momentos do ciclo de vida do software.

Contrapondo-se à reconstrução tem-se a refatoração de código fonte, a qual objetiva reestruturar o código seguindo um conjunto de passos preestabelecidos para atender a algum objetivo de qualidade previamente definido e conhecido pelos diferentes participantes de uma mesma equipe de desenvolvimento. A melhoria arquitetural e a preservação do comportamento externo do software, garantidas pelas atividades de refatoração segundo Fowler et al. (1999), além de identificarem fatores que diferem refatoração de reconstrução, impõem menores riscos ao desenvolvimento de software para sua realização quando comparadas com a de reconstrução.

Apesar dos menores riscos, a refatoração ainda pode ser vista como uma atividade de retrabalho que também demanda observação e entendimento do código fonte. Assim, a Empresa Alfa solicitou ao Grupo de Engenharia de Software Experimental (ESE) da COPPE/UFRJ, no contexto de um projeto de colaboração entre indústria e academia, auxílio em atividades inicialmente identificadas como sendo de refatoração de código fonte (posteriormente identificadas como de reconstrução) que estavam ocorrendo frequentemente em diferentes projetos de manutenção e evolução de software embarcado nas famílias de produto da empresa. Esta demanda definiu a oportunidade de aplicação de uma estratégia baseada em pesquisa-ação, na qual diferentes estudos experimentais – *survey* exploratório, revisão sistemática da literatura, coleta e análise qualitativa de dados de código fonte e *focus group* – foram conduzidos na organização como forma de melhor entender o contexto organizacional e auxiliar na proposição de uma estratégia de mitigação para os problemas identificados.

Por estar inserido em um contexto de pesquisa para apoiar a resolução de um problema real dentro da organização, as questões e os objetivos de pesquisa desse trabalho de dissertação estão intimamente relacionados aos (que foram) identificados para apoiar a proposição de uma estratégia de ação dentro da Empresa Alfa. A ação em questão teve como objetivo alinhar diferentes perspectivas de qualidade de código fonte – causa identificada para a ocorrência das constantes reconstruções de código fonte na empresa – e, assim, reduzir riscos com retrabalho. Detalhes do contexto organizacional onde o problema de reconstrução foi identificado, questões e objetivos de pesquisa para alinhar as diferentes perspectivas de qualidade de código fonte, bem como da metodologia do trabalho – baseada em pesquisa-ação – são apresentados na seção a seguir.

## 2. Contexto, Questões, Objetivos e Metodologia do Trabalho

### 2.1. A Empresa Alfa

A Empresa Alfa é uma empresa de desenvolvimento de soluções de hardware, firmware, software embarcado, desktop, web e mobile para os domínios de rastreamento de veículos e inteligência ambiental. Possui subdivisões localizadas em diferentes cidades do Brasil e do Exterior e, por conta disso, alguns de seus produtos são desenvolvidos de forma distribuída. O foco principal do desenvolvimento de software da empresa é o de firmware (linguagem C) e software embarcado (Lua) para rastreadores. Ao longo dos 16 anos de mercado, a empresa desenvolveu diferentes tipos de rastreadores que se diferenciam tanto em relação aos componentes de hardware, quanto em funcionalidades de firmware e software embarcado, porém que mantêm um conjunto mínimo de componentes de hardware e funcionalidades de software em comum.

As similaridades entre os diferentes produtos possibilitaram o reaproveitamento de código fonte nos seus diferentes produtos ao longo desses 16 anos, na medida em que houve evolução dos componentes de hardware ou mesmo o surgimento de novas oportunidades de atuação no mercado de rastreamento de veículos. É importante enfatizar que o reaproveitamento de código de um produto para outro não é feito de forma sistemática e é possível encontrar trechos de código em comum que sofreram modificações em um produto e que não sofreram modificações em outro produto.

Outra característica de contexto importante da organização está relacionada com a rotatividade, a maioria interna, dos desenvolvedores (cerca de 26 somente trabalhando com programação). Após a criação do setor de desenvolvimento para o domínio de inteligência ambiental nos últimos dois anos, muitos desenvolvedores da empresa do domínio de rastreamento migraram, aos poucos, para o desenvolvimento de soluções para esse novo domínio. Nesse período, também ocorreram muitas contratações, porém também houve saída de membros da equipe.

Nesse cenário, o líder da equipe de desenvolvimento de software salientou um problema recorrente na organização nos últimos anos: a grande quantidade de esforço que as atividades de "refatoração" de código fonte estavam demandando durante tarefas de manutenção de software sem, no entanto, proporcionar ao código os benefícios esperados de uma refatoração. Com o objetivo inicial de mitigar esse problema de retrabalho, o projeto de cooperação entre a Empresa Alfa e o Grupo ESE foi estabelecido.

### 2.2. Refatoração ou Reconstrução? Diagnóstico do Problema

O contexto inicialmente observado, apesar de importante para uma compreensão inicial do problema, não se mostrava suficiente para dar como causa definitiva do retrabalho a refatoração de código fonte. A indicação de realizações frequentes de atividades de refatoração – mensais, semanais e até mesmo diárias – evidenciou que algo mais, além da refatoração, poderia estar acorrendo durante as atividades de manutenção de software. Com isso, e antes mesmo de propor qualquer estratégia de ação com foco em refatoração de código, surgiu a necessidade de entender: "*o que representa qualidade em código fonte para os programadores da Empresa Alfa?*"; "*o que é entendido ser refatoração para os programadores da Empresa Alfa?*"; "*que situações-problema têm levado os programadores da Empresa Alfa a realizarem atividades de refatoração?*".

Essas questões iniciais induziram a condução do primeiro estudo experimental na

empresa: um *survey* exploratório para a captura da percepção dos desenvolvedores em relação à qualidade e refatoração de código fonte. O *survey* foi escolhido como uma estratégia adequada para o diagnóstico da organização principalmente pela característica distribuída das equipes de desenvolvimento e dos participantes do projeto de cooperação em questão. Os objetivos do *survey*, definidos utilizando a abordagem *Goal-Question-Metric* (GQM) proposta em (BASILI, CALDIERA e ROMBACH, 1994), foram os seguintes:

- **Analisar** a percepção de qualidade de código fonte dos programadores da empresa Alfa **com o propósito de** caracterizar **com respeito ao** entendimento comum sobre qualidade e priorização de características de qualidade em código fonte **do ponto de vista** dos programadores da empresa Alfa **no contexto de** desenvolvimento de firmware, software embarcado e software.
- **Analisar** a percepção de refatoração de código fonte dos programadores da empresa Alfa **com o propósito de** caracterizar **com respeito ao** entendimento comum sobre o que representa e objetiva a refatoração de código fonte e para que ela é realizada no contexto da organização **do ponto de vista** dos programadores da empresa Alfa **no contexto de** desenvolvimento de firmware, software embarcado e software.

Como forma de atingir a esses objetivos, três questionários foram elaborados. O primeiro para capturar características dos desenvolvedores e dos projetos existentes na organização que pudessem fornecer subsídios para a análise posterior dos dados. O segundo para capturar a percepção de importância para os desenvolvedores de determinadas características de qualidade em código fonte. E o terceiro para capturar a percepção dos desenvolvedores sobre refatoração de código fonte.

Ao todo 21 desenvolvedores responderam aos questionários. Apesar da alta taxa de concordância entre os desenvolvedores em relação às características de qualidade mais importantes para o código fonte – detalhes na Tabela 1 –, a lista de motivos que os faziam "refatorar" o código é bem diversa e não necessariamente expressa a preocupação em atender as características de qualidade mais importantes, estando mais relacionada com correção de problemas de legibilidade e compreensibilidade de código fonte – detalhes em (RIBEIRO, 2014).

**Tabela 1. Características de qualidade em ordem de importância para os desenvolvedores**

| % (valor agregado/máximo valor possível) | Característica de Qualidade | Percepção de Qualidade |
|---|---|---|
| 98,33 | Corretude/Efetividade | IMPRESCINDÍVEL |
| 80,00 | Confiabilidade | |
| 73,33 | Manutenibilidade | |
| 71,67 | Testabilidade | |
| 66,67 | Conformidade com Padrões | MUITO IMPORTANTE |
| 65,00 | Compreensibilidade | |
| 63,33 | Não Redundância | |
| 63,33 | Extensibilidade | |
| 58,33 | Legibilidade/Usabilidade | DESEJÁVEL |

Em relação à percepção de refatoração de código fonte, esta foi indicada pelos desenvolvedores como sendo sinônimo de correção de defeitos, inserção de novas funcionalidades e ainda a reconstrução de código fonte, conceitos variados e não condizentes com os existentes na literatura técnica de refatoração (FOWLER *et al.*, 1999) – detalhes

na Figura 1. Os resultados desse estudo acrescidos da confirmação de informações (entrevistas) junto aos desenvolvedores permitiram identificar que de fato ocorria na organização a reconstrução do código fonte e não sua refatoração, e que isso era devido à falta de consenso entre os programadores sobre qualidade no código fonte, principalmente do ponto de vista de legibilidade, fazendo com que um mesmo código fosse alterado diversas vezes por programadores diferentes na empresa, haja vista a alta rotatividade dos membros das equipes.

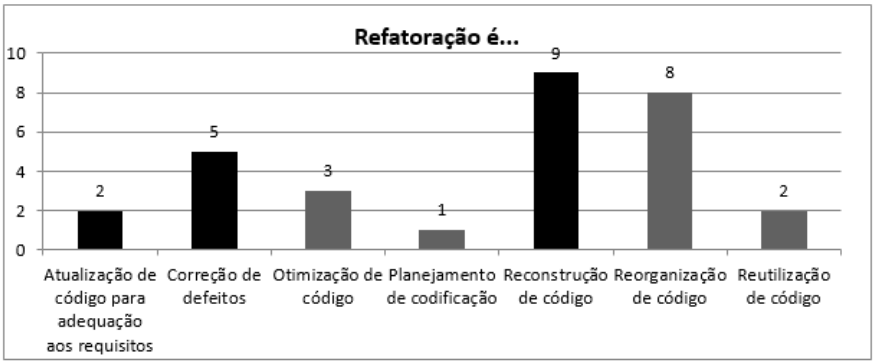


**Figura 1. Respostas codificadas para definição de refatoração**

Diante do novo entendimento sobre o problema, duas questões de pesquisa, focadas na real causa do retrabalho na empresa, foram identificadas e serviram como base para apoiar a elaboração da proposta de mitigação da reconstrução de código fonte: 1) "*como igualar as perspectivas de qualidade de código fonte entre os diferentes programadores da empresa Alfa?*"; 2) "*é possível identificar preditores para reconstrução de código e assim diminuir o índice de retrabalho dentro dos projetos de manutenção da empresa Alfa?*". Uma alternativa possível de ação relacionada com essas questões é o uso de diretrizes de codificação pelos desenvolvedores, uma vez que elas tanto podem ser utilizadas para igualar expectativas de qualidade para um código fonte, como para verificar a conformidade deste frente à qualidade esperada, podendo indiciar a existência de risco de reconstrução de códigos não conformes às diretrizes especificadas.

## 2.3. Objetivos e Metodologia de Pesquisa

O objetivo principal desta pesquisa consistiu em auxiliar a empresa Alfa a alinhar as perspectivas de qualidade em código fonte de seus desenvolvedores, provendo instrumentos que possibilitassem esse alinhamento, a saber: diretrizes de codificação baseadas em evidência para um conjunto específico de características de qualidade em código fonte associadas a uma estratégia de colaboração indústria-academia com foco no *design science* (HEVNER, MARCH e RAM, 2004).

Duas características de qualidade foram selecionadas para compor esse conjunto inicial de diretrizes de codificação: "Legibilidade" e "Compreensibilidade" de código fonte. A primeira por estar diretamente relacionada com as causas de reconstrução de código fonte, e a segunda por estar entre as características de qualidade vistas como mais importantes na organização (Tabela 1) e que ainda não tinha sido tratada em outras frentes de ação do projeto de colaboração Empresa Alfa-Grupo ESE – detalhes em (RIBEIRO, 2014). É importante frisar também que a característica "Conformidade com Padrões" também está sendo considerada nessa proposta de ação, uma vez que as diretrizes representam justamente um padrão de codificação a ser seguido.

Para atender ao objetivo traçado e tendo em vista a natureza de colaboração indústria-academia, a metodologia utilizada neste trabalho foi fundamentada na pesquisa-ação em Engenharia de Software (DOS SANTOS e TRAVASSOS, 2011) composta por 5 fases, como mostra a Figura 2. Para a fase de diagnóstico foram utilizadas: i) informações de contexto e relacionadas com o problema capturadas juntamente com o líder dos desenvolvedores da Empresa Alfa; associadas aos ii) resultados do *survey* conduzido junto aos programadores da empresa (subseção anterior). Destaca-se nessa fase a necessidade de capturar informações de diferentes fontes para a identificação do problema a ser solucionado, como forma de não incorrer no risco de obter um diagnóstico equivocado sobre o problema, ameaçando a validade da pesquisa.

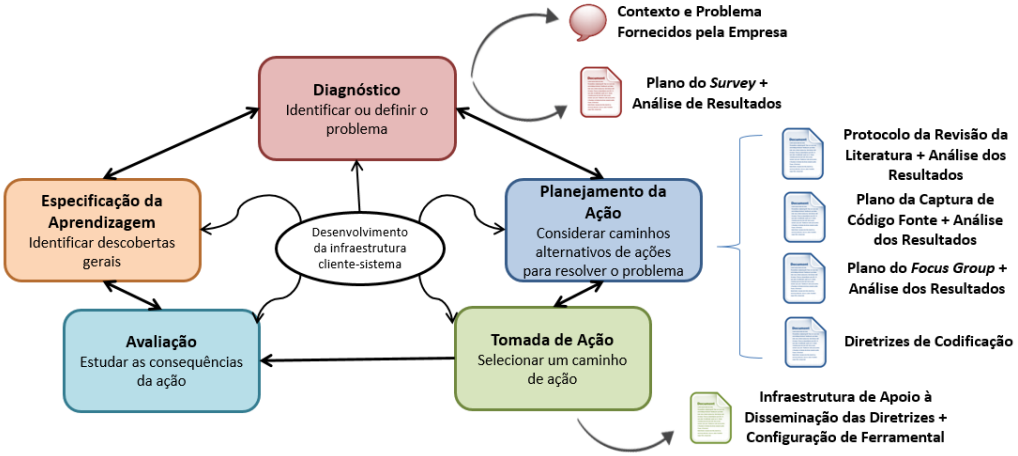


**Figura 2. Metodologia de pesquisa-ação e seus resultados**

O diagnóstico permitiu de fato escolher uma ação – uso de diretrizes de codificação – e iniciar o seu planejamento – organização de diretrizes que pudessem ao mesmo tempo ser baseadas em evidência científica e em necessidades reais da organização. Para tanto, e tendo em vista o rigor científico da metodologia, três estudos experimentais foram conduzidos: i) busca estruturada fundamentada em revisão sistemática da literatura para levantar atributos de código fonte para legibilidade e compreensibilidade baseados em evidência; ii) coleta e análise de informações de código fonte da empresa para identificar atributos de código fonte para legibilidade e compreensibilidade relevantes para o contexto da organização; e iii) avaliação da adequação para o contexto da Empresa Alfa das diretrizes formuladas a partir desses atributos, utilizando para isso o *focus group* como estratégia de observação. Mais detalhes sobre a resolução tomada para a escolha das estratégias de estudo e sobre o encadeamento entre elas podem ser obtidos em (RIBEIRO e TRAVASSOS, 2015).

A fase seguinte de tomada de ação refere-se ao uso de fato das diretrizes de codificação para legibilidade e compreensibilidade. Para isso a elaboração de uma infraestrutura de apoio à disseminação das diretrizes entre os programadores da Empresa Alfa e a configuração de ferramental de apoio à análise estática de código fonte foram realizadas. É importante destacar que, embora planejado e comunicado a empresa, as demais fases não puderam ser concluídas no âmbito desta dissertação e são destacadas como trabalhos futuros; por esse motivo, evitamos classificar a metodologia seguida como sendo puramente pesquisa-ação. Acreditamos, no entanto, que o objetivo principal do trabalho de alinhar as diferentes perspectivas de qualidade de código fonte foi atingido uma vez que

o ciclo de organização, avaliação e disseminação das diretrizes de codificação foi finalizado.

## 3. Busca Estruturada por Atributos de Código Fonte para Legibilidade e Compreensibilidade

O objetivo da busca estruturada conduzida foi identificar de forma isenta atributos de código fonte com evidência científica de impacto na legibilidade e compreensibilidade de código e, assim, possibilitar a posterior organização de diretrizes de codificação que levassem em consideração esses atributos. Seguindo a abordagem GQM, o objetivo da busca estruturada é como segue:

- **Analisar** atributos de código fonte **com o propósito de** caracterizar **com respeito ao** impacto (negativo ou positivo) na legibilidade e/ou compreensibilidade de código fonte **do ponto de vista de** programadores de software **no contexto de** desenvolvimento de software.

Três questões de pesquisa foram formuladas com base nesse objetivo: 1) "*quais atributos têm sido utilizados para avaliar a legibilidade e/ou a compreensibilidade de do código fonte?; 2) "como os atributos identificados podem ser mensurados?"; 3) "qual a relação de impacto existente entre os atributos identificados e a característica de qualidade avaliada (impacto negativo ou positivo)?"*. Essas questões guiaram a formulação da seguinte *string* de busca, executada somente na base *Scopus* devido a restrições de tempo do projeto de colaboração: TITLE-ABS-KEY((metric OR measure OR attribute OR predictor OR evaluation OR assessment OR improvement OR style OR standard OR pattern) AND (readability OR comprehensibility OR understandability OR identifier OR naming OR comment) AND ("software quality" OR "software readability" OR "software comprehension" OR "software understanding" OR "program quality" OR "program readability" OR "program comprehension" OR "program understanding" OR "code quality" OR "code readability" OR "code comprehension" OR "code understanding")).

Depois de executar a *string* de busca na *Scopus*, 236 artigos foram retornados. Cada um teve seu título, resumo e palavras chaves avaliados de acordo com os critérios de inclusão e exclusão previamente estabelecidos (ver em (RIBEIRO, 2014)). Em suma, para um artigo ser considerado aceito é necessário que ele apresente atributos de código fonte explicitamente relacionados com legibilidade e/ou compreensibilidade de código e que possua alguma avaliação experimental do impacto do atributo em uma dessas características de qualidade. Um artigo sendo aceito, as seguintes informações são requeridas para extração: i) descrição da característica de qualidade dada pelo autor; ii) atributos de código, suas respectivas descrições, procedimentos de mensuração e escala; iii) tipo de avaliação empírica da qualidade dos códigos fonte avaliados com base nos atributos; iv) descrição geral da avaliação; v) características dos códigos avaliados (linguagem de programação, aplicação real ou acadêmica dentre outras); vi) características dos participantes da avaliação (profissionais ou acadêmicos) – se aplicável; e vii) resultados da avaliação – impacto dos atributos na característica de qualidade avaliada.

Ao final da análise, 18 artigos foram aceitos para terem seus dados extraídos. Ao todo foram identificados 85 atributos de código fonte – incluindo os repetidos – dentre os quais se encontram atributos possíveis de serem coletados quantitativamente e outros, qualitativamente. É importante destacar que nem todos os atributos identificados foram levados em consideração na agregação final dos resultados, haja vista que o objetivo

inicial da revisão era analisar somente atributos que de fato influenciassem (ou negativamente ou positivamente) a legibilidade e/ou compreensibilidade de código fonte. Dessa forma, somente atributos com evidência de impacto nessas características foram considerados, assim, um total de 59 atributos de código fonte – 23 atributos qualitativos e 36 atributos quantitativos –, já removendo repetidos, foram considerados para a formulação das diretrizes iniciais.

Os atributos qualitativos identificados representam características do código fonte que podem apenas ser observadas de forma subjetiva. Estão relacionados com identificadores de variáveis (estilo de escrita), comentários de código (tipos de comentários), indentação (existência ou não) e organização do código (localização de estruturas de programa). Os atributos quantitativos, por sua vez, representam características do código fonte que podem ser mensuradas a partir de números. Estão relacionados com diferentes aspectos do código, principalmente sintáticos, exemplos são atributos de: *layout* – lidam com aspectos visuais do código; comentários – lidam com características específicas dos comentários do código; nomenclatura – lidam com identificadores; programação – lidam com aspectos sintáticos e de lógica do programa. A Tabela 2 exibe um subconjunto dos atributos quantitativos em cada uma das categorias identificadas.

**Tabela 2. Subconjunto dos atributos de código fonte quantitativos e seus impactos para legibilidade/compreensibilidade de código em diferentes estudos encontrados**

| CATEGORIA | ATRIBUTO | (BUSE e WEIMER, 2010)* | (DEYOUNG e KAMPEN, 1979)* | (JøRGENSEN, 1980)* |
|---|---|---|---|---|
| Layout | Indentação (espaço em branco precedente) máxima | ⊖ | - | - |
| Layout | Quantidade média de linhas em branco | ⊕ | - | - |
| Comentário | Extensão dos comentários | - | - | ⊕ |
| Comentário | Volume dos comentários (em caracteres) | - | ⊕ | - |
| Nomenclatura | Tamanho máximo de identificador | ⊖ | - | ⊕ |
| Nomenclatura | Tamanho médio de identificador | ⊗ | ⊕ | ⊕ |
| Programação | Quantidade média de *loops* | ⊖ | - | - |
| Programação | Quantidade média de operadores aritméticos | ⊖ | - | - |

⊖ Impacto negativo na legibilidade/compreensibilidade de código/Inversamente proporcional
⊕ Impacto positivo na legibilidade/compreensibilidade de código/Diretamente proporcional
⊗ Impacto na legibilidade/compreensibilidade não conclusivo
* Trabalho referenciado e descrito em (RIBEIRO, 2014)

De posse dos atributos qualitativos e quantitativos com evidência da literatura de seus impactos na legibilidade e compreensibilidade de código fonte, foi necessário nesse momento identificar quais deles faziam sentido no contexto de desenvolvimento de software da Empresa Alfa. Assim, antes mesmo de agrupá-los para a elaboração de diretrizes, era necessário observar quais atributos de código fonte são vistos como influenciadores da alta, baixa e intermediária legibilidade e/ou compreensibilidade de código fonte dentro da organização, uma vez que só faz sentido prover uma tecnologia de software que esteja alinhada com as expectativas daqueles que a vão utilizar. A seção a seguir relata como ocorreu a captura dos atributos de código fonte no contexto da indústria e como esses atributos foram combinados com as evidências encontradas na literatura para a formulação das diretrizes de codificação.

## 4. Diretrizes de Codificação para Legibilidade e Compreensibilidade

Sendo o objetivo das diretrizes de codificação alinhar as perspectivas de qualidade de código fonte dos desenvolvedores da Empresa Alfa, fez-se necessário entender "*quais atributos de código fonte são relevantes para avaliar a legibilidade e/ou a compreensibilidade de do código sob o ponto de vista dos desenvolvedores da empresa" para, então,* combinar as expectativas destes com as evidências encontradas na literatura técnica, gerando assim as diretrizes. Nesse sentido, foi elaborada uma estratégia para extração de atributos de código que, na percepção da Empresa Alfa, impactassem na legibilidade e compreensibilidade de código fonte, sendo o seu objetivo definido como segue:

- **Analisar** atributos de código fonte **com o propósito de** caracterizar **com respeito ao** impacto (negativo ou positivo) na legibilidade e/ou compreensibilidade de código fonte **do ponto de vista de** programadores de firmware, software embarcado e software da empresa Alfa **no contexto de** desenvolvimento de firmware, software embarcado e software da Empresa Alfa.

Para capturar a perspectiva de legibilidade e compreensibilidade, planejamos a realização da seguinte tarefa, repassada aos programadores da empresa Alfa: "Identificar e extrair 3 trechos de códigos, um de alta, outro de baixa e outro de intermediária legibilidade e compreensibilidade, não produzidos por você, mas com os quais você mantém contato direto em suas tarefas de desenvolvimento na empresa. Os trechos de código devem ser autocontidos, ou seja, devem representar um algoritmo independente de outros trechos de código dentro do software. É importante que a lógica de programação do trecho de código extraído não seja quebrada. O trecho de código deve conter no mínimo 3 comandos consecutivos (sendo pelo menos 1 composto), devendo respeitar as marcas de mínimo e máximo no espaço delimitado no *template* de captura de código. Comentários, espaços, tabulações e linhas em branco existentes não devem ser retirados dos trechos de código, ou seja, o código deve ser extraído exatamente como foi encontrado.".

Como instrumentos de auxílio a essa tarefa, dois formulários distintos foram elaborados. O primeiro formulário contém instruções para a captura de trechos de código de alta, baixa e intermediária legibilidade e compreensibilidade – uma descrição literal sobre legibilidade e compreensibilidade foi fornecida no formulário, como forma a não enviesar a percepção pessoal de cada desenvolvedor. O segundo formulário, cujo acesso pelo programador teve que ser dado somente após a captura dos códigos, contém um conjunto de atributos de código fonte (extraídos da literatura técnica) mapeados em frases a serem avaliados pelo programador em relação a seu impacto na legibilidade e compreensibilidade de código. Esse formulário contém ainda questões abertas para que os participantes indiquem aspectos/atributos de código existentes em códigos de alta e de baixa legibilidade e compreensibilidade.

Os trechos de código fonte coletados (33 ao todo – 11 de alta, 11 de baixa e 11 de intermediária legibilidade e compreensibilidade) foram analisados qualitativamente pela pesquisadora e revisados pelos pesquisadores à luz dos atributos de código identificados com a busca estruturada, e levando em consideração as quatro categorias de atributos detectadas, a saber: *layout*, comentário, nomenclatura e programação (detalhes em (RIBEIRO, 2014)). A Figura 3 a seguir sumariza os aspectos considerados na análise qualitativa. Para essa análise, cada trecho de código recebeu uma *tag* para cada aspecto do código analisando, possibilitando a identificação de similaridades em alto nível entre trechos de código de baixa, alta e intermediária legibilidade e compreensibilidade. Por

exemplo, para o aspecto de uso de parênteses da categoria de *layout*, *tags* como "uso de parênteses somente quando necessário" e "uso de parênteses para enfatizar operandos em expressões" foram elaboradas para caracterizar trechos de código em relação a esse aspecto específico.


**Figura 3. Categorias e aspectos considerados para a análise qualitativa dos trechos de código fonte**

É importante salientar que como a análise dos códigos fonte foi feita de forma qualitativa, objetivamos também avaliar qualitativamente os atributos quantitativos encontrados nos artigos aceitos da busca estruturada. Salientamos também que os atributos identificados nos artigos foram utilizados como guia de análise dos códigos capturados, não sendo, no entanto, limitador da análise. Assim, novos atributos, aspectos e até mesmo categoria (arquivos e pastas) surgiram com a análise qualitativa dos códigos.

Em relação às questões objetivas e abertas do formulário de percepção sobre legibilidade e compreensibilidade, as respostas objetivas foram contabilizadas e agrupadas de acordo com a escala *likert* presente no formulário. As questões abertas tiveram suas respostas codificadas, e as ocorrências de informações (atributos) similares relacionadas com alta e baixa legibilidade/compreensibilidade de código fonte foram contabilizadas para posterior utilização em conjunto com os demais dados coletados.

### 4.1. Agrupamento das Informações Coletadas para Formulação de Diretrizes

A análise qualitativa dos códigos fonte permitiu ter uma visão geral de como cada aspecto presente na Figura 3 se apresentava em trechos de código de baixa, alta e intermediária legibilidade e compreensibilidade. Para a formulação das diretrizes, características de aspectos presentes em trechos de código de alta legibilidade e compreensibilidade e em falta em trechos de baixa legibilidade e compreensibilidade foram traduzidas em recomendações a serem seguidas. Por outro lado, características de aspectos presentes somente em trechos de código de baixa legibilidade e compreensibilidade foram traduzidas em advertências de uso. De modo geral, para a Empresa Alfa, para um código ser legível e compreensível:

- *Layout*: as declarações de variáveis precisam estar distantes de suas referências; cada linha deve conter somente um comando que deve ser curto e separado por linhas em branco de outros comandos; indentação e espaçamento devem ser consistentes; e parênteses devem ser utilizados somente quando necessários para executar um comando ou identificar precedência em expressões aritméticas e lógicas.
- Comentários: os comentários devem ser completos e ter um propósito claro, como uma explicação passo a passo de um algoritmo, uma explicitação de motivos para utilização

de determinadas variáveis em um código, ou uma descrição de comportamentos de funções e procedimentos.

- Nomenclatura: os identificadores devem ser curtos e significativos (mnemônicos), utilizando abreviações conhecidas ou palavras completas.
- Programação: "*go to*" e constantes literais (números) como operandos devem ser evitados.

A junção da análise dos códigos fonte com as respostas objetivas e abertas gerou a versão inicial das diretrizes de codificação para legibilidade e compreensibilidade de código fonte apresentada de forma simplificada (somente título) na Tabela 3 a seguir (detalhes em (RIBEIRO, 2014)). Para cada diretriz que faz referência a atributos identificados na literatura, o impacto (positivo, negativo ou neutro) para a legibilidade e compreensibilidade de código fonte foi destacado em conjunto com o a referência para o estudo.

**Tabela 3. Versão 1 das diretrizes de codificação com evidência da literatura de impacto na legibilidade/compreensibilidade do código fonte**

| DIRETRIZES (Versão 1) | Evidência na Literatura* – Impacto na Legibilidade e Compreensibilidade |
|---|---|
| Utilize os nomes de extensão de arquivos de acordo com o conteúdo do arquivo | - |
| Cada módulo deve ser composto por pelo menos um arquivo de definição e outro de implementação | - |
| Utilize a seguinte ordem de organização para arquivos de definição | - |
| Utilize a seguinte ordem de organização para arquivos de implementação | - |
| A indentação deve ser feita de forma consistente | ⊖ (BUSE e WEIMER, 2008)<br>⊖ (BUSE e WEIMER, 2010)<br>⊕ (JøRGENSEN, 1980)<br>⊕ (MIARA *et al*., 1983) |
| O uso de chaves para identificação de blocos deve ser consistente | - |
| O espaçamento entre operadores e operandos deve ser feito de forma consistente (1 espaço) | - |
| Utilize parênteses para delimitar operandos em expressões | ⊖ (BUSE e WEIMER, 2008)<br>⊖ (BUSE e WEIMER, 2010) |
| Utilize linhas em branco para separar instruções extensas das demais instruções e para separar blocos de instruções de diferentes propósitos | ⊕ (BUSE e WEIMER, 2008)<br>⊕ (BUSE e WEIMER, 2010)<br>⊕ (JøRGENSEN, 1980) |
| Faça apenas uma declaração por linha | - |
| Escreva apenas uma instrução simples por linha | - |
| Tamanho de linha deve ser menor que 80 caracteres | ⊕ (BUSE e WEIMER, 2008)<br>⊕ (BUSE e WEIMER, 2010) |
| Variáveis devem ser declaradas em conjunto e no início do seu escopo válido | ⊖ (SASAKI, HIGO e KUSUMOTO, 2013) |
| Constantes devem ser declaradas em conjunto e no início do seu escopo válido | ⊖ (SASAKI, HIGO e KUSUMOTO, 2013) |
| Selecione um idioma (Inglês ou Português) no qual será redigido os comentários | - |
| Insira um comentário de identificação (cabeçalho) para cada arquivo | - |
| Insira um comentário de sumarização do propósito de cada função | ⊕ (BUSE e WEIMER, 2008)<br>⊕ (BUSE e WEIMER, 2010)<br>⊕ (DEYOUNG e KAMPEN, 1979)<br>⊕ (JøRGENSEN, 1980)<br>⊕ (STEIDL, HUMMEL e JUERGENS, 2013)<br>⊕ (TENNY, 1988)<br>⊕ (WOODFIELD, DUNSMORE e SHEN, 1981) |
| Comentários de linha devem ser utilizados somente em casos específicos | ⊕ (STEIDL, HUMMEL e JUERGENS, 2013) |
| Selecione um idioma (Inglês ou Português) no qual será redigido o código | - |

| | |
|---|---|
| Os identificadores devem ser de fácil memorização | (+) (LAWRIE *et al*., 2006)<br>(+) (LAWRIE *et al*., 2007)<br>(+) (TEASLEY, 1994) |
| Constantes e variáveis globais devem ser identificadas como tais | - |
| Os identificadores de variáveis devem ser consistentes quanto a sua forma de representação | (+) (BINKLEY *et al*., 2009)<br>(+) (BINKLEY *et al*., 2013)<br>(−) (BUSE e WEIMER, 2008)<br>(−) (BUSE e WEIMER, 2010)<br>(+) (DEYOUNG e KAMPEN, 1979)<br>(+) (JøRGENSEN, 1980)<br>(+) (LAWRIE *et al*., 2006)<br>(+) (LAWRIE *et al*., 2007)<br>(+) (SHARAFI *et al*., 2012)<br>(+) (SHARIF e MALETIC, 2010) |
| Os identificadores de constantes devem ser consistentes quanto a sua forma de representação | (+) (BINKLEY *et al*., 2009)<br>(+) (BINKLEY *et al*., 2013)<br>(−) (BUSE e WEIMER, 2008)<br>(−) (BUSE e WEIMER, 2010)<br>(+) (DEYOUNG e KAMPEN, 1979)<br>(+) (JøRGENSEN, 1980)<br>(+) (LAWRIE *et al*., 2006)<br>(+) (LAWRIE *et al*., 2007)<br>(+) (SHARAFI *et al*., 2012)<br>(+) (SHARIF e MALETIC, 2010) |
| As constantes utilizadas devem ser simbólicas | - |
| Evitar código comentado que não é mais utilizado | - |
| Atentar para o tamanho da solução (em código) dado ao problema, isso pode ser um indicativo de que ou o problema não está bem estruturado ou a complexidade estrutural da solução é alta. Em blocos, funções e arquivos | (+) (DEYOUNG e KAMPEN, 1979)<br>(?) (FEIGENSPAN *et al*., 2011)<br>(−) (POSNETT, HINDLE e DEVANBU, 2011) |
| * Evidências em trabalhos referenciados e descritos em (RIBEIRO, 2014) | |

## 5. Avaliação de Adequação das Diretrizes na Organização: *Focus Group*

Apesar de as diretrizes formuladas terem sido baseadas fortemente nos atributos de código identificados dentro da organização, dez delas ou foram observadas em poucos códigos coletados/mencionados por poucos desenvolvedores da organização ou continham evidências divergentes na literatura técnica, levando a uma baixa confiança de sua aplicabilidade no contexto de desenvolvimento da Empresa Alfa. Dessa forma, era necessário utilizar uma abordagem de avaliação que pudesse caracterizar essas diretrizes em relação a sua aplicabilidade dentro da empresa. O *focus group* foi escolhido como uma estratégia que ao mesmo tempo possibilita essa caracterização e ajuda na disseminação das diretrizes entre os desenvolvedores da empresa, auxiliando no objetivo do trabalho de alinhar as diferentes perspectivas de qualidade de código fonte.

Assim o *focus group* foi planejado com o seguinte objetivo de pesquisa:

- **Analisar** diretrizes de codificação para legibilidade e compreensibilidade **com o propósito** de caracterizar **com respeito à** pertinência e parametrização de uso na empresa Alfa **do ponto de vista de** programadores de firmware, software embarcado e software da empresa Alfa **no contexto de** desenvolvimento de firmware, software embarcado e software da empresa Alfa.

As seguintes questões de pesquisas foram elaboradas: 1) *"as diretrizes de codificação para legibilidade e compreensibilidade de código fonte são aplicáveis ao contexto de desenvolvimento da Empresa Alfa?"*; 2) *"De que forma as diretrizes podem ser modificadas ou parametrizadas para que elas sejam aplicáveis ao contexto de desenvolvimento da Empresa Alfa?"*. Para responder a essas questões de forma particular a cada equipe de desenvolvimento, dois grupos, um para o setor de desenvolvimento de software embarcado e outro para o setor de software web foram dispostos em uma mesma

sala para discutirem sobre a aplicabilidade de dez diretrizes em seus respectivos contextos de desenvolvimento.

A Figura 4 exemplifica o quadro elaborado para a realização do *focus group*. A dinâmica ocorreu como segue: i) os moderadores (pesquisadores) inseriram nos quadros uma diretriz a ser caracterizada e apresentaram sua especificação completa em um projetor; ii) os participantes discutiram entre si sobre a diretriz; iii) cada participante escreveu sua opinião sobre a diretriz, informando em um post-it rosa, o porquê de ela não se aplicar ao seu contexto de desenvolvimento; amarelo, os casos excepcionais que ela não se aplica ao seu contexto de desenvolvimento; verde, em que casos ela se aplica ao seu contexto de desenvolvimento sem restrições; e azul, como ela poderia ser melhorada/alterada para se adequar ainda mais ao seu contexto de desenvolvimento. Todas informações de discussão foram registradas em notas para posterior avaliação dos dados – detalhes do planejamento e execução do *focus group* podem ser encontrados em (DE FRANÇA *et al.*, 2015).

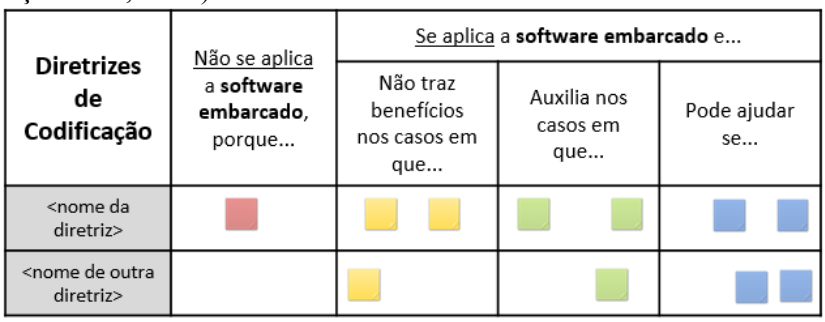


**Figura 4. Quadro utilizado pelo grupo de software embarcado durante a sessão de *focus group***

A maioria das diretrizes foi caracterizada como sendo aplicável ao contexto de desenvolvimento da Empresa Alfa (*post-its* amarelos, verdes e azuis foram majoritariamente utilizados). Apenas uma diretriz ("Utilize os nomes de extensão de arquivos de acordo com o conteúdo do arquivo") foi identificada como não aplicável ao contexto de desenvolvimento de software web. O argumento utilizado para apoiar essa opinião foi a de que código PHP necessita ser escrito junto com HTML e CSS, não tendo, então, como aplicar a diretriz nesse contexto. No entanto, é importante enfatizar que a diretriz pretende justamente evitar o agrupamento de linguagens e tipos de conteúdo, uma vez que os desenvolvedores da empresa acreditam que essa é uma prática prejudicial para a legibilidade do código. Os demais post-its apresentaram basicamente restrições de uso às diretrizes, isso é, casos em que elas não são necessárias; e, também, algumas oportunidades de extensão delas para abranger outros elementos do programa não mencionados.

Como resultado final as diretrizes de codificação foram evoluídas, atendendo às oportunidades de melhoria evidenciadas com o *focus group*. Além disso, um hiperdocumento contendo detalhes, justificativas e exemplos de uso das diretrizes foi criado como forma de servir de consulta em caso de dúvidas. Para consulta rápida e disseminação das diretrizes, um *folder* foi elaborado e distribuído entre as equipes de desenvolvimento. Também a ferramenta *AStyle* foi configurada com instruções para ajustar automaticamente o código fonte da organização a determinadas diretrizes de *layout* formuladas, como exemplo deste último caso, para a diretriz referente a indentação consistente do código fonte, três instruções foram configuradas na ferramenta: --indent=spaces=3, instrução para realizar a indentação de comandos utilizando 3 espaços; --indent-switches,

instrução para indentar o *case* dentro do comando *switch*; e --indent-col1-comments, instrução para indentar os comentários juntamente com o código. Ver (RIBEIRO, 2014) para informações completas de cada uma das diretrizes finais formuladas.

## 6. Considerações Finais

Nesta dissertação de mestrado diferentes estratégias de estudos experimentais foram combinadas para auxiliar uma organização de desenvolvimento de software a alinhar as perspectivas divergentes de qualidade de código fonte de seus desenvolvedores, perspectivas estas que estavam acarretando em constantes atividades de retrabalho durante a manutenção de software. Ao longo das três fases destacadas na metodologia baseada em pesquisa ação seguida – diagnóstico, planejamento e tomada da ação –, todos os produtos de trabalhos gerados se mostram como contribuições diretas deste trabalho para a Empresa Alfa, em um sentido restrito, e também para a comunidade científica e de profissionais da prática, em um contexto mais amplo, uma vez que os protocolos de estudos podem ser adequados e utilizados em outros contextos de pesquisa e organizacionais.

Para a indústria, este trabalho contribuiu com os diferentes diagnósticos fornecidos, bem como com as diretrizes de codificação (hiperdocumento e folder de disseminação das diretrizes) baseadas em evidência para legibilidade e compreensibilidade de código fonte utilizadas para mitigar os riscos com reconstrução de código fonte. Fora do âmbito específico da organização, acreditamos que as diretrizes com evidência na literatura de impacto na legibilidade e compreensibilidade de código também podem ser utilizadas, uma vez que as especificações para o contexto da empresa ficaram a cargo das justificativas e exemplos fornecidos para uso. Convém destacar, contudo, que as diretrizes identificadas somente na organização necessitam de melhor avaliação para sua aplicação em outros contextos. Os planos dos estudos conduzidos na Empresa Alfa podem ser adequados e utilizados para apoiar esse último caso.

Do ponto de vista acadêmico podemos destacar as seguintes contribuições: i) o protocolo da busca estruturada, que pode ser estendido para auxiliar na identificação de atributos de código para outras características de qualidade; ii) o resultado obtido com a busca estrutura que pode ser utilizado em outros contextos de pesquisa que não somente o de formulação de diretrizes; e iii) a própria estratégia de combinação de diferentes estudos experimentais para resolução de um problema real no desenvolvimento de software que pode ser tomada como base em ações a serem realizadas em outros ambientes organizacionais de desenvolvimento de software.

Como limitação principal do trabalho tem-se a não conclusão do ciclo completo de pesquisa-ação, embora seja esperado como trabalho futuro; e, também, a necessidade de avaliação experimental de algumas das diretrizes identificadas somente a partir de dados da indústria. Além disso, as próprias limitações dos estudos conduzidos também podem ser vistas como limitações do trabalho (detalhes em (RIBEIRO, 2014)).

Como consideração final da pesquisa realizada em parceria com a indústria, atentamos para o nível de aparente trivialidade dos problemas identificados; enfatizando que a indústria ainda enfrenta problemas que para muitos na academia já foram resolvidos. Isso nos leva a questionar a aplicabilidade das tecnologias e abordagens disponíveis para a engenharia de software e mesmo a disseminação da importância que determinadas práticas de engenharia de software tem para o processo de desenvolvimento

de software como um todo.

## Agradecimentos

Os autores agradecem à CAPES e ao CNPq pelo apoio financeiro e a Empresa Alfa pela oportunidade de realização dessa colaboração.

## Referências

BASILI, V.; CALDIERA, G.; ROMBACH, H. (1994) The Goal Question Metric Approach. In: _____ Encyclopedia of Software Engineering. Hoboken: John Wiley & Sons, v. 2, p. 528-532.

BOEHM, B. (2006) "A View of 20th and 21st Century Software Engineering". Proceedings of the 28th International Conference on Software Engineering. Shanghai, p. 12-29.

BOEHM, B.; ROMBACH, H. D.; ZELKOWITZ, M. (2005) Foudations of Empirical Software Engineering: The Legacy of Victor R. Basili. Berlin: Springer-Verlag, 432 p. ISBN 978-3-540-24547-6.

CASS, A.; SUTTON JR., S.; OSTERWEIL, L. (2003) Formalizing Rework in Software Processes. Lecture Notes in Computer Science, 2786, p. 16-31.

CONROY, P.; KRUCHTEN, P. (2012) "Performance Norms: An Approach to Rework Reduction in Software Development". Proceedings of thr 25th IEEE Canadian Conference on Electrical and Computer Engineering. Montreal, p. 1-6.

DE FRANÇA, B.; RIBEIRO, T.; DOS SANTOS, P.; TRAVASSOS, G. (2015) "Using Focus Group in Software Engineering: Lessons Learned on Characterizing Software Technologies in Academia and Industry". Proceedings of XVIII Ibero-American Conference on Software Engineering, Track: XVII Experimental Software Engineering Latin American Workshop. Lima, p. 351-364.

DOS SANTOS, P.; TRAVASSOS, G. (2011) Action Research can Swing the Balance in Experimental Software Engineering. Advances in Computers, 83, p. 205-276.

FOWLER, M.; BECK, K.; BRANT, J.; OPDYKE. W.; ROBERTS, D. (1999) Refactoring: Improving the Design of Existing Code. 1ª. ed. Boston: Addison-Wesley Longman Publishing Co., ISBN 0-201-48567-2.

GOPAL, A.; MUKHOPADHYAY, T.; KRISHNAN, M. S. (2002) The Role of Software Processes and Communication in Offshore Software Development. Communication of the ACM, 45, n. 4, p. 193-200.

HEVNER, A.; MARCH, S.; RAM, S. (2004) Design Science in Information Systems Research. MIS Quarterly, 28, n. 1, p. 75-105.

RIBEIRO, T. (2014) Alinhando Perspectivas de Qualidade em Código Fonte a partir de Estudos Experimentais – um Caso na Indústria. Dissertação de Mestrado: COPPE / Universidade Federal do Rio de Janeiro, 161 p.

RIBEIRO, T.; TRAVASSOS, G. (2015) "On the Alignment of Source Code Quality Perspectives through Experimentation: An Industrial Case". Proceedings of III International Workshop on Conducting Empirical Studies in Industry. Florence, p. 26-33.