Dívida Técnica: um estudo de caso com produtos de código aberto

Igor Rodrigues Vieira[1], Leonardo da Silva Sousa[1], Vinícius Rafael Lobo de Mendonça[1], Cássio Leonardo Rodrigues[1], Auri Marcelo Rizzo Vincenzi[1]

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74001-970 – Goiânia – Goiás – Brazil

{igorvieira, leonardosousa, viniciusmendoca, cassio, auri}@inf.ufg.br

Abstratc. *This paper evaluates the technical debt metaphor in open source systems, considering their status over those systems, in order to demonstrate the feasibility of use this approach to manage and assess the quality of product. This work is related to the evaluation of recent versions of a set of forty projects of Free Software Community, chosen arbitrarily. These projects are evaluated by Sonar platform, which allows collecting static and dynamic metrics, including their Technical Debt. The results are used to evaluate the characteristics of the technical debt on this set of projects, beyond to check if this is in acceptable levels.*

Resumo. *Este artigo avalia a metáfora da dívida técnica em produtos de código aberto, considerando seu estado sobre esses produtos, no intuito de demonstrar a possibilidade de utilização dessa abordagem para o gerenciamento e avaliação da qualidade. Este trabalho está relacionado com a avaliação de recentes versões de um conjunto de quarenta projetos da Comunidade de Software Livre, escolhidos arbitrariamente. Esses projetos são submetidos à avaliação da plataforma Sonar, a qual possibilita coletar métricas estáticas e dinâmicas, entre elas a Dívida Técnica. Os resultados obtidos são utilizados para avaliar as características da dívida técnica sobre esse conjunto de projetos, além de verificar se ela está em um nível aceitável.*

1. Introdução

A metáfora da dívida técnica *(technical debt metaphor)* foi introduzida em 1992 por Ward Cunningham, quando a Wyatt Software lançava um sistema de gerenciamento de portfólios – o WyCASH+, empregando a tecnologia de objetos com o uso da linguagem Smalltalk. Na época, a empresa considerava que esse novo paradigma de programação se adaptava melhor à diversidade e dinâmica do mercado e apresentava capacidade de resposta mais rápida e eficiente às necessidades de ajuste do produto [Cunningham 1992].

Esse cenário evidenciava uma consciente decisão de projeto da equipe em empregar uma nova tecnologia, pouco conhecida pelos desenvolvedores, mas que oferecia vantagem às características do negócio, mesmo considerando que isso poderia exigir reescrita de trechos do código. Assumia-se uma "dívida", em função da sua conveniência ao projeto, mas com a necessidade de pagá-la, sob o risco dos "juros" crescerem a níveis incontroláveis.

Várias pessoas usam a metáfora da "dívida" para descrever muitos outros tipos de débitos e males do desenvolvimento de software, abrangendo amplamente tudo o que está no caminho da implementação, venda ou evolução de um sistema de software ou qualquer coisa que adicione desgaste aos esforços para seu desenvolvimento. A

generalização do uso do termo para outras questões do desenvolvimento de software torna o conceito da dívida técnica (DT) diluído atualmente [Kruchten 2012].

A dívida passa a ser uma âncora na produtividade e manutenção do software e impede a capacidade de adicionar novas características eficientemente. Essa dívida pode resultar do fato de tomar decisões conscientes durante o projeto e a implementação, favorecendo a conveniência, mas que podem afetar negativamente a "saúde" do software a longo prazo. A dívida pode ocorrer também de forma mais presente por meio de uma lenta decadência causada pelo fato de repetidamente remendar um sistema existente sem a devida refatoração [Eisenberg 2012].

A metáfora da DT também é empregada na metodologia de desenvolvimento ágil. Kruchten et al. [Kruchten et al. 2012] consideram que algumas equipes de desenvolvimento ágil acreditam estar completamente imunes à DT pelo fato de usarem um processo de desenvolvimento iterativo. Contudo, o efeito pode ser contrário, uma vez que desenvolver e entregar rapidamente o produto, sem tempo para pensar em aspectos do projeto, refletir sobre os custos a longo prazo ou adotar análises sistemáticas, incluindo testes automatizados, levam alguns projetos a rapidamente aumentarem suas DTs.

Ferramentas de análise estática e dinâmica dão suporte à identificação e análise da dívida, uma vez que possibilitam a automatização do processo de verificação do código. Contudo, Kruchten et al. [Kruchten et al. 2012] admitem que essa abordagem pode levar ao perigo de desconsiderar grande quantidade de DT potencial não detectadas por essas ferramentas, tal como lacunas tecnológicas existentes.

É perceptível na literatura que não há pretensão de eliminar a DT, mas sim mantê-la em um nível gerenciável, pois ela é inerente à dinâmica do desenvolvimento de software, especialmente o processo de tomada de decisão [Kruchten et al. 2012]. A metáfora significa que, quando o projeto gerencia adequadamente a DT, ela pode ajudá-lo a alcançar seus objetivos mais cedo do que seria possível de outra forma. A gestão da DT envolve rastrear, tomar decisões fundamentadas e evitar seus piores defeitos. Dessa forma, o grau que um projeto ou empresa precisa para gerenciar a sua dívida depende claramente do seu contexto [Lim et al. 2012].

O objetivo deste artigo é analisar a metáfora da DT em sistemas da Comunidade de Software Livre, levando em consideração a facilidade de acesso ao código fonte de uma grande quantidade de projetos e o interesse de realizar, ainda que de forma preliminar e exploratória, uma análise sobre a situação da DT em sistemas dessa natureza.

O restante deste trabalho está organizado da em seis seções. Na Seção 2 são apresentados alguns trabalhos relacionados sobre a aplicação da metáfora da DT. A Seção 3 aborda alguns fundamentos e conceitos importantes para compreensão desse estudo. Na quarta seção é apresentado o método empregado para o desenvolvimento do estudo de caso. A Seção 5 traz os dados coletados de um conjunto de 40 produtos de código aberto (PCA). Na Seção 6 é apresentada a avaliação dos resultados do estudo de caso. A Seção 7 apresenta as considerações finais e propostas para trabalhos futuros.

2. Trabalhos Relacionados

O processo de desenvolvimento de software constitui-se uma atividade complexa e requer diversos tipos de decisões de projeto que podem afetar atividades futuras do desenvolvimento [Siebra et al. 2012].

Kruchten et al. [Kruchten et al. 2012] estabelecem uma forma de organização para a concepção e visualização da dívida que pode ser comparada ao processo de melhoria de software em um dado estado. O panorama sobre a DT proposto faz uma distinção entre elementos visíveis, tais como adição de novas funcionalidades e correção de defeitos, e elementos invisíveis (ou visíveis apenas aos desenvolvedores), relacionados às lacunas tecnológicas e questões de arquitetura e codificação. Ainda sobre os elementos visíveis são destacadas questões de evolução e de qualidade.

No estudo conduzido por Eisenberg [Eisenberg 2012], a plataforma Sonar [Sonar 2012] foi inicialmente selecionada para medir a DT pelo fato de possuir o *Plugin Technical Debt*. Entretanto, o autor considerou difícil aplicar, de maneira prática e significativa, a avaliação da DT apresentada pela plataforma Sonar. Assim, apesar de considerar que ela apresenta uma interface robusta e outras características desejáveis, ele não empregou o cálculo da DT utilizado pela plataforma Sonar [Sonar 2012] no estudo, mas utilizou-se das métricas oferecidas por ela para o cálculo do débito.

O trabalho citado acima estabelece um conjunto de métricas importantes, bem como limites específicos dos projetos para definir níveis gerenciáveis da DT [Eisenberg 2012]. O autor definiu níveis para cada eixo de qualidade da – partindo do gerenciável ao crítico. Para determinar o status da dívida foi necessário estabelecer prioridades no seu cálculo. O objetivo era reduzir a dívida para o status verde, representando, portanto, um nível aceitável. Isso permitiu estabelecer parâmetros para avaliação e priorização da DT no sistema analisado. Contudo, esse trabalho não estabeleceu níveis para a proporção da DT do projeto.

Siebra et al. [Siebra et al. 2012] afirmam que o estado da arte sobre a DT ainda não culminou em rigorosos modelos de análise para projetos em larga escala. Considerando isso, os autores analisaram um projeto industrial sob a perspectiva das decisões de projeto e eventos relacionados, para melhor caracterizar a existência da DT e mostrar a evolução dos seus parâmetros. O trabalho dos autores teve como objeto de estudo o sistema SMB (Samsung Mobile Business), um aplicativo comercial de software para dispositivos móveis que pode ser executado em diferentes plataformas Samsung dessa natureza. Uma característica importante desse sistema é que ele possuía um elevado número de modificações, sendo conveniente aplicar a abordagem da DT. O método empregado no trabalho consistia da identificação, coleta de dados e análise de três cenários de decisão, que foram tidos como uma abordagem adequada para caracterização da evolução história da DT. A avaliação da DT teve como foco a análise dos cenários e as decisões de projeto associadas a eles. Essa abordagem é importante de ser investigada em trabalhos futuros.

O trabalho apresentado aqui procura sintetizar os estudos já existentes e evoluir o conceito de DT para uma quantidade significativa de projetos, descrevendo as características da DT para um conjunto de produtos de código aberto.

3. A Plataforma Sonar e o Cálculo da Dívida Técnica

O Sonar é uma plataforma aberta para gerenciamento da qualidade do código. A qualidade do software é avaliada de acordo com sete eixos: Comentários (*Comments),* Potencial Erros *(Potential Bugs),* Complexidade *(Complexity),* Testes Unitários *(Units Tests),* Duplicações *(Duplications),* Regras de Codificação *(Coding Rules),* Arquiterura *(Architecture)* e Projeto *(Design)* [Sonar 2012]. A plataforma faz uso de outras

ferramentas de análise estática – CheckStyle [CheckStyle 2012], FindBugs [FindBugs 2012] e PMD [PMD 2012] – para extração de métricas de software.

A plataforma Sonar utiliza o *Technical Debt Plugin* [Technical Debt 2012] para o cálculo da DT do projeto. Primeiramente, são calculados os valores de seis eixos básicos: Duplicação (*Duplication*), Violação (*Violation*), Complexidade (*Complexity*), Cobertura (*Coverage*), Documentação (*Documentation*) e Projeto (*Design*). Em seguida, essas métricas são somadas para fornecer uma métrica global. Os detalhes da composição dessa métrica são descritos abaixo:

- *Proporção da dívida (the debt ratio):* fornece um percentual da DT atual em relação ao total da dívida possível para o projeto.
- *O custo para reembolsar (the cost to reimburse):* determina o custo, em valor monetário, necessário para limpar todos os defeitos em cada eixo.
- *O trabalho para reembolsar (the work to reimburse):* fornece o custo para reembolsar expresso no esforço homens/dia.
- *A repartição (the breakdown):* apresenta um gráfico com a distribuição do débito nos seis eixos de qualidade.

A versão atual do Plugin (1.2.1) realiza o cálculo da dívida conforme ilustrado na Equação (1).

Dívida (em homens/dia) = (custo para corrigir duplicações) + (custo para corrigir violações) + (custo para comentar API pública) + (custo para corrigir complexidade descoberta) + (custo para trazer a complexidade ao limite inferior) + (custo para cortar ciclos em nível de pacote). (1)

A plataforma Sonar permite que os custos sejam configurados pelo usuário. Os valores/pesos utilizados neste estudo são descritos na Tabela 1. Após o cálculo dos custos, multiplicados pelos respectivos pesos, os quais podem ser adaptados segundo o critério da organização/empresa, é possível expressar o valor da DT em termos do esforço homens/dias (considerando a configuração padrão do dia com 8 horas). Essa medida é então multiplicada por $500 (configuração padrão do Plugin) para se obter o custo monetário da DT.

Tabela 1. Custo padrão de cada componente da DT, em horas

Tipo de custo	Custo (horas)
Custo para corrigir um bloco	2
Custo para corrigir uma violação	0,1
Custo para comentar uma API	0,2
Custo para cobrir uma complexidade	0,2
Custo para dividir um método	0,5
Custo para dividir uma classe	8
Custo para cortar uma aresta entre dois arquivos	4

O cálculo da proporção da dívida (*technical debt ratio*) é basicamente a dívida sobre a Dívida Possível Total (DPT) multiplicado por 100. A primeira coisa a ser calculada é o DPT em cada eixo para depois somá-las:

Duplicação (DU): dado o número de blocos de duplicação e a porcentagem atual de linhas duplicadas, calcular o número de blocos que haveria se fosse 100%. A dívida total possível é o custo para corrigir esses blocos.

Violações (VI): dado o Índice de Observância das Regras (IOC) atual e o número de violações, calcular quantas violações seriam necessárias para trazer o IOC para zero. A dívida total possível é o custo para corrigir esses blocos.

Cobertura (CB): a dívida total possível é o custo de elevar a cobertura para 80%.

Complexidade (CP): a dívida total possível é o custo de dividir cada método e classe.

Comentários (CM): a dívida total possível é o custo para comentar cada API pública.

Projeto (PR): dadas as arestas existentes entre os arquivos, o custo de ter que cortar todas aquelas necessárias para que não se tenham ciclos (índice emaranhado de pacote = 100%).

Para este trabalho, optou-se por utilizar a fórmula empregada pelo *Technical Debt Plugin*, com os valores/pesos estabelecidos na configuração padrão, uma vez que, no momento, não há a intenção de questionar o modo como a DT é calculada pelo Plugin mas sim utilizar os valores para a comparação entre diferentes projetos.

4. Método de preparação do estudo de caso

A análise sobre a DT em PCA, estabelecida neste trabalho, é baseada na avaliação de um conjunto de 40 projetos da Comunidade de Software Livre, escolhidos arbitrariamente entre os avaliados pelo Projeto Nemo [Nemo Sonar 2013] da plataforma Sonar. O objetivo é obter dados para avaliação do estado e das características da DT sobre esse conjunto específico de PCA.

A opção em analisar a situação da DT em PCA deu-se pela facilidade de acesso ao código fonte, bem como pelo interesse em propor uma avaliação, ainda preliminar e exploratória, dessa abordagem sobre projetos dessa natureza. Assim, foram selecionados quarenta produtos que atendiam os seguintes requisitos: a) o produto utiliza o Apache Maven 2 [Maven 2012] como ferramenta de construção, b) o produto é escrito em Java; e c) o produto dispõe de casos de testes unitários. Essas restrições técnicas foram impostas por serem essenciais para o cômputo da cobertura dos testes unitários necessária para o cálculo da DT pelo Sonar.

Em uma etapa seguinte, foram obtidas versões recentes dos projetos nos respectivos repositórios. Os projetos foram submetidos à avaliação da plataforma Sonar, e as ferramentas de análise estática e dinâmica integradas à, permitindo assim realizar a análise do código e geração dos respectivos relatórios/base de dados.

5. Estudo de Caso

Para o trabalho em questão, apenas os dados gerados pelo *Plugin Technical Debt* foram coletados. A Tabela 2 apresenta os dados relacionados à proporção da DT, os valores da sua repartição por eixos de qualidade, bem como a respectiva média, desvio padrão, limiar inferior – identificados por (*) –, e limiar superior – identificados por (#) – para o conjunto de projetos selecionados. Os dados brutos da pesquisa estão disponíveis em: http://www.inf.ufg.br/~auri/dados-artigo-sbqs2013.pdf.

Observando os dados da tabela percebem-se indicadores variados da DT entre os projetos. É importante destacar que a proporção da DT está relacionada à dívida possível total ao projeto. Assim, por exemplo, a proporção da DT do projeto *CheckStyle*

é de 2,5% em relação à dívida possível total desse projeto. O intervalo entre os extremos dos projetos para essa métrica é de 25,1%, sendo o *Apache Abdera* com a maior DT.

Tabela 2. Dados sobre a DT dos produtos de código aberto selecionados.

Projeto	DT (%)	Repartição da Dívida (100 %)					
		CM	CP	CB	PR	DU	VI
CheckStyle	2,5 (*)	4,1 (*)	69,7 (#)	0,0 (*)	0,0 (*)	9,2	16,9
Sonar	4,1 (*)	49,5 (#)	15,7	12,9	14,9	2,2 (*)	4,8
Google Visualization	4,3 (*)	0,4 (*)	50,6 (#)	2,8 (*)	0,0 (*)	18,4 (#)	27,7 (#)
Doxia Aggregator	4,6 (*)	0,3 (*)	46,6 (#)	15,2	4,9	11,9	21,1
Commons Configuration	4,9 (*)	10,5	60,8 (#)	0,0 (*)	14,5	6,7	7,4
Apache Shindig Project	5,1 (*)	22,2	28,8	10,8 (*)	25,6 (#)	1,0 (*)	11,6
Google Http Client Library	5,5 (*)	12,6	31,0	33,4	0,0 (*)	6,6	16,3
Junit	7,2	19,9	17,7	0,0 (*)	40,8 (#)	5,4	16,2
Apache Shiro	7,3	18,4	18,5	32,7	15,3	2,3 (*)	12,7
Apache Rave	7,4	25,9 (#)	18,8	24,4	9,3	7,6	14,0
Apache MyFaces CODI	7,6	7,5	13,6	67,4 (#)	5,3	1,9 (*)	4,3
Jacoco	8,6	30,2 (#)	4,6 (*)	32,6	5,9	6,7	20,0
Jackcess	8,8	22,2	41,4 (#)	0,0 (*)	3,4	2,5 (*)	30,6 (#)
Apache Vysper Parent	8,9	14,2	12,6	10,7 (*)	31,2 (#)	12,1	19,2
Apache Cocoon3	9,0	21,0	16,8	46,1 (#)	5,0	4,8	6,2
PMD	9,0	32,6 (#)	21,6	13,8	19,5	4,2	8,2
Apache Archiva	9,5	24,8 (#)	22,6	39,5	1,2 (*)	5,2	6,7
Xwork Parent	9,6	16,0	23,9	18,4	13,5	18,6 (#)	9,5
Logback Parent	9,6	26,1 (#)	9,3	24,6	17,1	5,6	17,3
Apache Turbine	10,0	0,7 (*)	17,1	39,3	20,4	12,8	9,7
PDFBox Reactor	11,4	3,1 (*)	16,1	39,7	13,6	6,0	21,5
Struts 2	11,7	16,1	16,5	32,4	6,8	13,5	14,7
Apache Tika	12,0	9,3	19,8	12,7	4,6	5,7	47,9 (#)
Apache OpenWebBeans	12,1	14,1	15,5	19,9	40,9 (#)	2,1 (*)	7,5
Apache Karaf	12,1	14,0	16,6	47,2 (#)	0,9 (*)	8,2	13,2
Cayenne	12,6	13,2	13,4	26,0	26,4 (#)	10,4	10,6
Apache Jackrabbit	12,8	5,1	20,2	38,9	12,0	10,8	13,0
Apache Pluto	12,9	21,3	11,1	47,4 (#)	2,5	10,8	6,9
Apache Synapse	13,2	10,4	18,1	38,6	18,8	6,7	7,4
Apache Tobago	13,4	16,3	11,3	41,0	8,1	19,0 (#)	4,2
Commons BCEL	14,1	9,3	15,4	42,0	8,3	7,8	17,3
Struts	15,0	10,0	17,9	43,7	8,4	10,1	9,9
Continuum Project	15,2	22,0	18,9	43,2	0,7 (*)	7,8	7,4
Apache Jena	17,6 (#)	6,6	10,1	14,2	27,5 (#)	19,8 (#)	21,7
Apache Empire DB	18,4 (#)	11,2	15,0	39,4	8,2	9,2	17,1
ColLib	19,6 (#)	8,5	21,6	35,5	20,4	0,0 (*)	14,0
ActiveMQ	20,8 (#)	12,8	12,4	39,3	7,5	20,9 (#)	7,0
Apache Axis2	23,7 (#)	9,6	19,3	34,2	10,4	16,5 (#)	9,9
Open JPA	23,7 (#)	10,8	13,5	43,0	9,7	7,8	15,3
Apache Abdera	27,6 (#)	12,1	7,3 (*)	14,6	2,8	2,9 (*)	60,3 (#)
Média dos Projetos	**11,6**	**14,9**	**21,3**	**27,9**	**12,2**	**8,5**	**15,2**
Desvio Padrão	**5,8**	**9,8**	**13,9**	**16,4**	**10,6**	**5,5**	**11,1**
Limiar Inferior (*)	**5,8**	**5,1**	**7,4**	**11,5**	**1,6**	**3,0**	**4,1**
Limiar Superior (#)	**17,4**	**24,6**	**35,2**	**44,3**	**22,7**	**14,1**	**26,2**

Analisando os dados por eixo de qualidade é possível avaliar quais deles exercem maior impacto na distribuição da DT para cada projeto e considerando esse conjunto de PCA. Para análise dos dados foi calculada a média (11,6%) da proporção da DT dos projetos selecionados e o desvio padrão (5,8%) dessa mesma métrica. Com isso, calculou-se o limiar inferior (obtido pela subtração do valor do desvio padrão sobre a média – 5,8%) e o limiar superior (obtido pela adição do valor do desvio padrão sobre média – 17,4%). O mesmo procedimento foi realizado para os eixos de qualidade que compõem a repartição da DT. Com isso, foram estabelecidos três grupos de projetos: os com melhor desempenho, abaixo do limiar inferior; os com desempenho mediano, dentro do intervalo de confiança; e aqueles com pior desempenho, acima do limiar superior, conforme pode ser observado na Tabela 2.

6. Resultados

Os projetos com melhor desempenho foram: *CheckStyle*, *Sonar*, *Google Visualization*, *Doxia Agregator*, *Commons Configuration*, *Apache Shindig Project* e *Google Http Client Library*. Já os projetos com maiores valores da dívida foram: *Apache Jena*, *Apache Empire DB*, *ColLib*, *ActiveMQ*, *Apache Axis2*, *Open JPA* e *Apache Abdera*. Observa-se que os projetos com menores ou maiores valores da proporção da DT não apresentam o mesmo desempenho para todos os seis eixos de qualidade, uma vez que a dívida é repartida entre esses eixos e, naturalmente, algum deles terá maior peso.

Os dados da Tabela 2, referentes à média dos projetos, demonstram que os eixos de qualidade "Complexidade" e "Cobertura" apresentam maior relevância na composição da dívida, representando 49,23% do total enquanto os outros quatro eixos juntos totalizam 50,77%. É possível observar ainda que os eixos "Duplicações" e "Projeto" possuem menor impacto na composição da dívida (20,69%), o que pode sugerir uma tendência de boas práticas da Comunidade de Software Livre sobre esses eixos de qualidade. Já os outros dois componentes "Comentários" e "Violações" apresentam valores medianos (30,05%) para distribuição da DT.

Do conjunto de projetos avaliados observa-se que apenas sete deles apresentaram valores insatisfatórios em relação à dívida, demonstrando que boa parte dos projetos encontra-se em um nível da dívida que pode ser considerado aceitável. Contudo, a análise dos dados dos eixos de qualidade, especialmente da média dos projetos relacionada à "Complexidade" e à "Cobertura" sugerem a necessidade de maior atenção às questões arquiteturais relacionadas à complexidade e definição de casos de testes que possibilitem melhor cobertura do código fonte.

O estudo conduzido por [Eisenberg 2012], citado na Seção 2, estabelece limites aceitáveis para cada eixo da DT, contudo ele utiliza um cálculo diferente do empregado pelo *Plugin Technical Debt*, restringindo a confrontação dos seus dados com os resultados deste trabalho, prevista para trabalho futuro.

7. Considerações Finais e Trabalhos Futuros

Com a avaliação dos projetos realizada pela plataforma Sonar, utilizando o *Plugin Technical Debt*, é possível analisar a situação da dívida técnica de PCA, bem como verificar quais eixos de qualidade são mais representativos na composição da DT. Por exemplo, pode-se sugerir que há uma lacuna nas questões arquiteturais e de cobertura de testes entre esse conjunto de projetos da Comunidade de Software Livre, justificada pela

proporção dos eixos “Complexidade” e “Cobertura” na composição da repartição média da DT entre os projetos avaliados.

A avaliação realizada neste trabalho permite considerar que os projetos avaliados apresentam, em sua maioria, valores aceitáveis e pouco elevados em relação à proporção da dívida técnica. Também foi possível verificar os eixos de qualidade de maior e menor representatividade na composição da dívida. Considera-se que essa avaliação auxilia em aspectos relacionados à gestão da dívida técnica, mantendo-a em níveis aceitáveis, favorecendo ainda a evolução e qualidade desses sistemas.

Como proposta para trabalhos futuros é importante buscar a ampliação da base de dados, bem como sua representatividade no conjunto de projetos na Comunidade de Software Livre. Torna-se interessante reforçar a análise estatística da composição da DT para verificar a correlação entre seus componentes e observar tendências. Uma abordagem importante seria avaliar o histórico da dívida para versões distintas dos PCA ou sua evolução num determinado intervalo de tempo. Outra possibilidade é analisar os resultados da dívida técnica dos projetos sob outras perspectivas relacionadas à sua própria gestão, qualidade de software e sustentabilidade da arquitetura desses sistemas, sendo este trabalho um estudo inicial para essas outras pesquisas em andamento.

8. Referências Bibliográficas

[Cunningham 1992] Cunningham, W. (1992) “The WyCash Portfolio Management System”. OOPSLA’92. Experience Report.

[Eisenberg 2012] Eisenberg, R. J. (2012) “A Threshold Based Approach to Technical Debt”. ACM SIGSOFT Software Engineering Notes, vol. 37, n. 2, 2012, pp. 01-06.

[Kruchten et al. 2012] Kruchten, P. et al. (2012) “Technical Debt: From Metaphor to Theory and Practice”, IEEE Software, vol. 29, n. 6, 2012, pp. 18-21.

[Lim et al. 2012] Lim, E. et al. (2012) “A Balancing Act: What Software Practitioners Have to Say about Technical Debt”, IEEE Software, vol. 29, n. 06, 2012, pp. 22-27.

[Siebra et al. 2012] Siebra, C. A. et al. (2012) “Managing Technical Debt in Practice: An Industrial Report”, Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, pp. 247-250.

[Sommerville 2011] Sommerville I. (2011). Engenharia de Software, 9 ed. Addison-Wesley.

[CheckStyle 2012] Home page CheckStyle. Disponível: http://checkstyle.sourceforge.net.

[FindBugs 2012] Home page FindBugs. Disponível: http://findbugs.sourceforge.net/.

[Maven 2012] Home page Maven. Disponível: http://maven.apache.org/.

[Nemo Sonar 2013] Home page Nemo Sonar. Disponível: http://nemo.sonarsource.org/.

[PMD 2012] Home page PMD. Disponível: http://pmd.sourceforge.net/.

[Sonar 2012] Home page Sonar. Disponível: http://www.sonarsource.org/.

[Technical Debt 2012] Plugin Technical Debt. Disponível: http://docs.codehaus.org/display/SONAR/Technical+Debt+Plugin.