FinTD IV
Finding Technical Debt Experiment

Mário André (IFS/UFBA)
Thiago Souto (IFBA/UFBA)
Ph.D. Student in Computer Science
Prof. Dr. Manoel Mendonça (UFBA/Fraunhofer)
Prof. Dr. Rodrigo Spínola (UFBA/Fraunhofer)

AGENDA

- Dívida Técnica
- Tipos de Dívida
- Gerenciamento da Dívida Técnica
- Identificação da Dívida Técnica
  - Indicadores de Dívida Técnica
  - Code Smells
  - Análise de Comentários
- Experimento FindTD

INTRODUÇÃO

Introdução

- Durante a **evolução**, a **qualidade do software diminui** ao levar em consideração **aspectos** como sua **estrutura interna**, **adesão a normas**, **documentação** e **facilidade de entendimento para futuras manutenções** (LIENTZ *et al.*, 1978) (LEHMAN e BELADY, 1985) (PARNAS, 1994);

- Esse cenário traz impactos na produtividade uma vez que uma modificação em um software de baixa qualidade geralmente é mais difícil de realizar do que em um de alta qualidade (LEHMAN e BELADY, 1985).

Dívida Técnica

A Dívida Técnica (DT) é a **lacuna** entre:

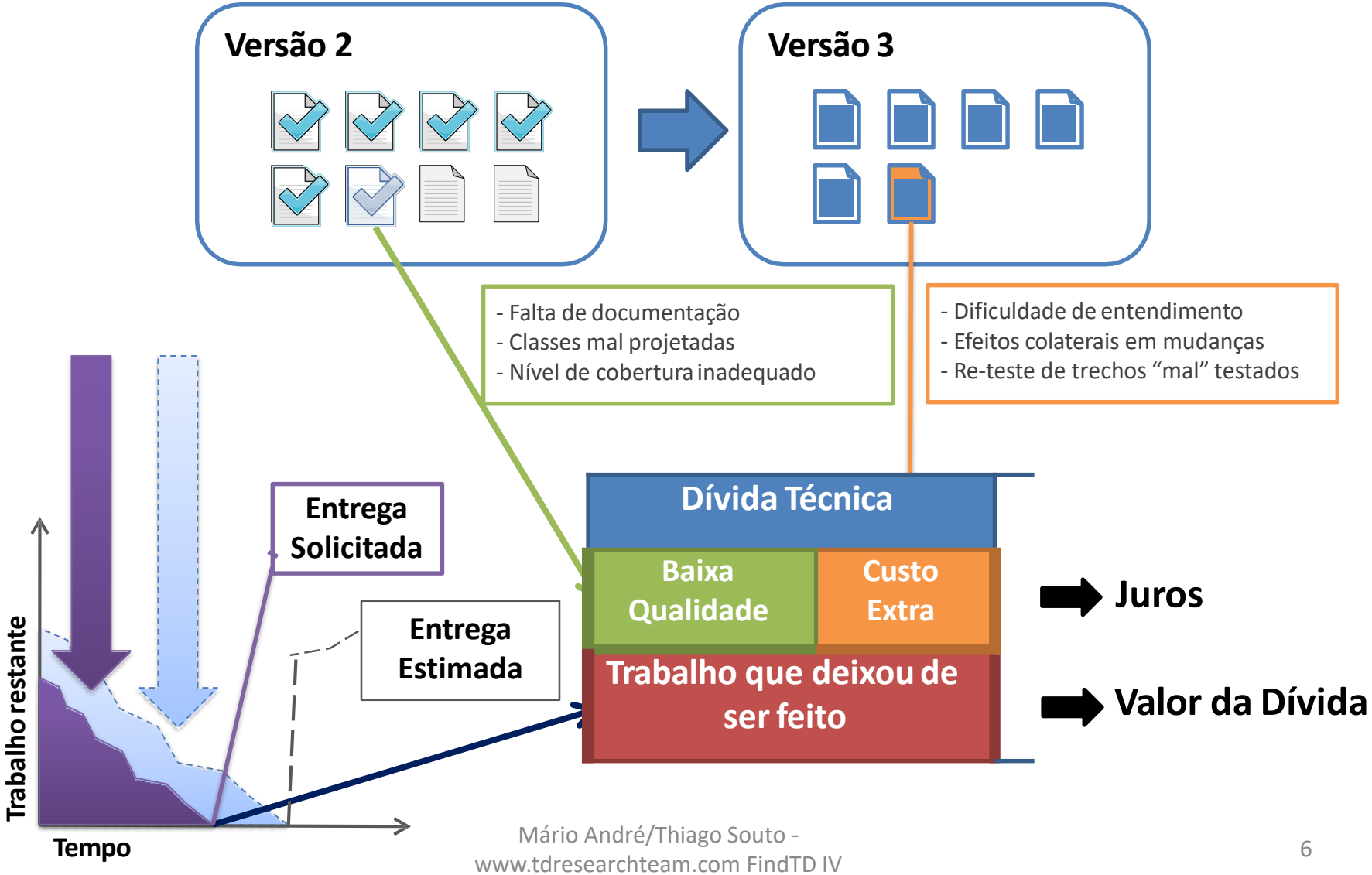
- Desenvolver um sistema de forma **perfeita**
  - Preservando o projeto arquitetural
  - Empregando boas práticas de programação e padrões
  - Atualizando a documentação
  - Testando exaustivamente
- E colocar o sistema para **funcionar**
  - O mais rápido possível
  - Utilizando o mínimo de recursos possível

Dívida Técnica

Dívida Técnica representa os efeitos de **artefatos imaturos** na evolução do software que **trazem benefícios a curto prazo** em termos de aumento da produtividade e redução de custos, mas que tem que ser **ajustado mais tarde com juros**.

[Seaman and Guo, 2011; Kruchten *et al.*, 2012; Izurieta *et al.*, 2012]

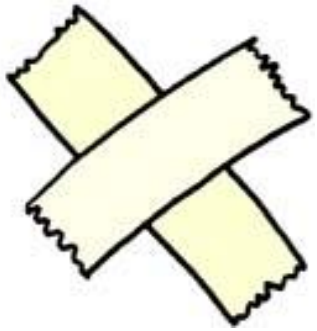
Como a Dívida Técnica Ocorre



Technical Debt – Summarizes

Definição:

- **Artefato incompleto**, imaturo ou inadequado no ciclo de vida de desenvolvimento do software;
- Aspectos do software que **sabemos que estão errados**, mas não temos tempo para ajustá-los agora;
- Tarefas que deixaram **de ser feitas**, mas que correm um risco de causar problemas futuros se não forem efetuadas;



Benefícios:

- Maior **produtividade** no curto prazo;
- Menor custo da versão atual;

Custos:

- "**Juros**" – aumento dos custos de evolução;
- **Risco** que a dívida fuja do controle.

Tipos da Dívida Técnica

Tipos de Dívida Técnica

Dívida de Arquitetura	Dívida de Código	Dívida de Serviço
Dívida de Defeito	Dívida de Construção (Build)	Dívida de Usabilidade
Dívida de Projeto	Dívida de Infraestrutura	Dívida de automação de teste
Débito de Documentação	Dívida de Pessoal	Dívida de Teste
Dívida de Processo	Dívida de Requisito	Dívida de Versionamento

Gerenciamento da Dívida Técnica

Gerenciamento de Dívida Técnica

- É comum que a equipe incorra em dívidas uma vez que elas podem aumentar a produtividade do time;

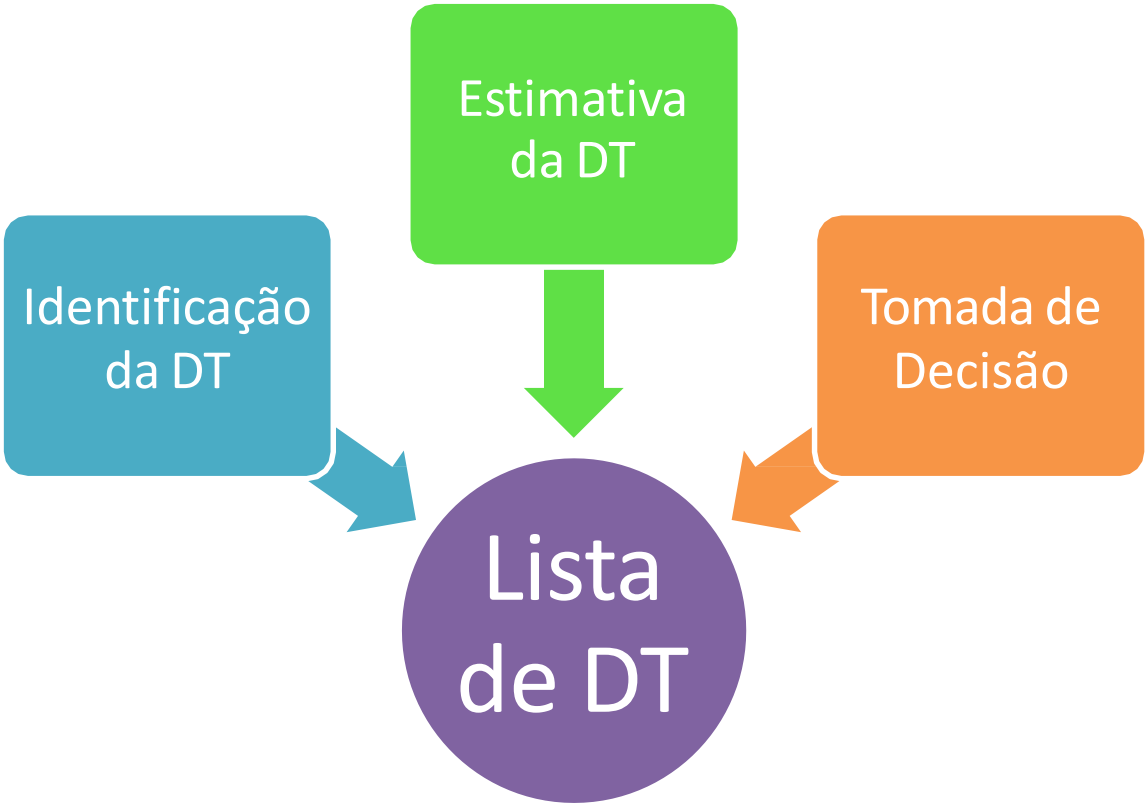
- Se uma equipe **não gerencia a DT**, ela pode **causar problemas significativos a longo prazo**, como custos de evolução aumentados e atrasos inesperados no projeto [Brown et al., 2010];

- A gestão da DT é fundamental para alcançar e manter a qualidade do software e compreende **ações** de **identificação**, **monitoramento** e **pagamento da dívida** ao longo do ciclo de vida de um software [Guo et al., 2014].

Exemplo de Item de DT

ID	37
Data	31/04/2016 (Versão3.2)
Responsável	Henrique
Tipo	Projeto
Localização	Método **m** no Módulo **X**
Descrição	Na última versão, o método **m** foi adicionado rapidamente e não é seguro do ponto de vista de uso de threads.
Valor base da dívida estimado	Médio (nível médio de esforço para modificar **m**)
Quantidade de juros estimada:	Alto (se precisarmos esperar para modificar **m**, poderão existir mais módulos dependentes dele que precisarão ser modificados)
Probabilidade estimada dos juros:	Baixa (não é provável que sejam adicionadas chamadas simultâneas ao método **m**)

Gerenciamento de Dívida Técnica



[Guo *et al.*, 2014]

O objetivo do gerenciamento de DT é….

Eliminar todos os itens da dívida técnica

Determinar quando:

- A quantidade de juros evitado justifica seu pagamento ou o custo de pagar a dívida!

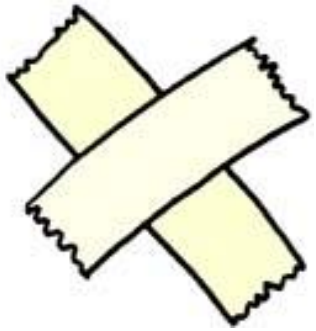
Identificação da Dívida Técnica

Sinais da presença de DT

Violação de modulariodade
Problema de performance
Código desnecessário

Problemas no código fonte
Práticas inadequadas de codificação
Defeitos conhecidos e não conhecidos

Código complexo
Problemas com difíceis soluções
Violação de princípios de padrões de orientação a objetos (Classes enormes, alto níveis de acoplamento, god class e smells)



Problemas na documentação do projeto (documentação inadequada, inexistente ou incompleta)
Problemas relacioanado a testes

Identificação de DT

INDICADORES DOS TIPOS DE DT (Adaptado) [1] [2]
→ Themas de Dívida Técnica;

Indicadores	Tipo de DT
Violação de modularidade	*Dívida de Arquitetura*
Problemas na arquitetura do software	
Betweenness Centrality	
Complexidade Arquitetural	
Dependências estruturais	*Dívida de Arquitetura / Dívida de Construção (build)*
Structural Analysis	*Dívida de Arquitetura / Dívida de Projeto (Design)*

Identificação de DT

Structural Analysis	*Dívida de Projeto (Design)*
Problemas de construção	*Dívida de Construção (Build)*
Processo de construção manual	
Código sem padrões	*Dívida de Código*
Algoritmo lento	
Multithread Correctness	
Código duplicado	
Código de baixa qualidade	
Código complexo	
Código ultrapassado (deprecated)	
Code Metrics (not specified)	*Dívida de Projeto (Design)/ Dívida de Código*
Automatic Static Analysis (ASA) Issues	
Code Smells	
Grime	*Dívida de Projeto (Design)*

Identificação de DT

Questões de projeto de software	
Low External / Internal Quality	
Afferent / Efferent Couplings (AC / EC)	
Depth of Inheritance Tree (DIT)	
Métodos e classes complexos	
Defeitos conhecidos não corrigidos	*Dívida de Defeito / Dívida de Teste*
Defeitos/Bugs	*Dívida de Defeito*
Comentários insuficientes no código	*Dívida de Documentação*
Falta de documentação	
Comentários (hack, fixme, is problematic, ...)	
Problemas na documentação	
Documentação desatualizada	
Falta de comentário de código ou comentários desatualizados	
Falta de deployment automático	*Dívida de Infraestrutura*
Tecnologia antiga em uso	
Ferramentas antigas de suportes	

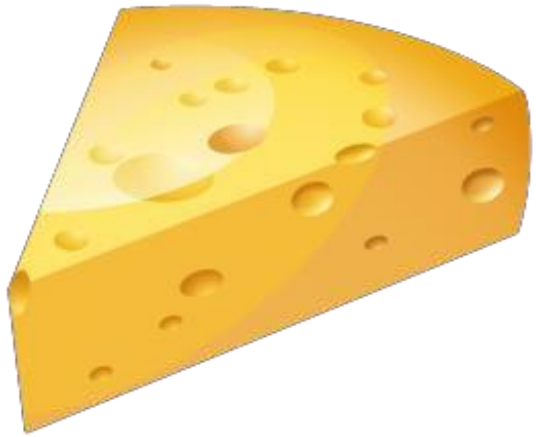
Testes incompletos	*Dívida de Teste*
Correção de defeitos adiada	
Cobertura de código insuficiente	
Falta de documentação de casos de teste	
Falta de planejamento de casos de teste	
Testes caros/custosos	

Identificação de DT

- ASA Issues
- Code Smells
- Grime
- Violação de Modularidade
- Métricas de Código
- Detecção Manual
- Análise de Comentários

Code Smells

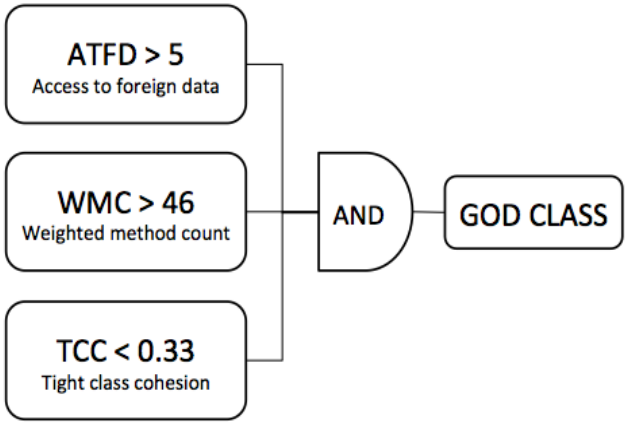
- Métodos e classes que violam os princípios de um bom projeto orientado a objetos
- Exemplos:
  - Encapsulamento
  - Ocultação de Informação
  - Interfaces bem definidas
  - Uso apropriado de herança

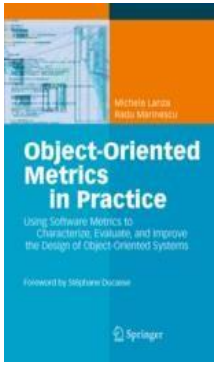


- Code Smells apontam para **potenciais** problemas:
  - Requer investigação e confirmação do desenvolvedor
  - Hoje existem cerca de 20 code smells documentados

Code Smells - God Class

- God Class
  - Classe que possui muitas responsabilidades e está mais interessada em dados de outras classes do que seus próprios dados.





  - Abordagens automatizadas têm sido propostas para detecção de code smells baseadas no trabalho de Marinescu. Ex: Java: CodeVizard e Marple

Identificação de DT

- ASA Issues
- Code Smells
- Grime
- Violação de Modularidade
- Métricas de Código
- Detecção Manual
- Análise de Comentários

Temas de comentários

TD types	Themes
Architecture debt	Performance or Robustness problems
	Violation of modularity
	Dependency
	Theme Noun Architecture
Build debt	Unnecessary time or memory consuming
	Dependency
	Theme Noun Build
Code debt	Redundancy
	Temporary solution or workarounds
	Bad coding practices
	Inadequate solution
	To do better (improvements)
	Duplicate code
	Unnecessary or not used code
	It is not working or does not work very well
	Can causes exception
	Deprecated code
	Difficult to be maintained in the future
	Low/bad external/internal quality of properties
	Theme Noun Code
Defect debt	Uncorrected known defects or bugs
	Defects or bugs
	Something to be fixed
	Theme Noun Defect
Design debt	Inadequate location
	Temporary solution or workarounds
	Code smells
	Inadequate solution
	Duplicate code
	Violation of principles of good design
	Theme Noun Design
Documentation debt	Misunderstanding
	Documentation-missing or inadequate or incomplete
	Theme Noun Documentation
Requirement debt	Need to be safe
	Need to implement or it wasn't well implemented
	Not according
	Theme Noun Requirement
Test debt	Tests to do
	Deficiencies in testing activities
	Insufficient code coverage

Alguns indicadores de comentários

“Não se preocupe com a documentação agora”

“A única pessoa que pode mexer nesse código é João.”

“Serve para agora, mas precisaremos refatorar depois”

“ToDo/FixMe: isto precisa ser corrigido antes da release”

“Eu sei que se mexer no código, algo vai deixar de funcionar!”

“Iremos finalizar os testes na próxima release.”

“A data para liberação da release está próxima, então, apenas faça funcionar!”

Alguns Exemplos - JEdit

JEdit			
Id	Comment	Class	Method
371	undocumented hack to allow browser actions to work. XXX - clean up in 4.3	JavaCharStream	arFile
176	This should not be hardcoded	SplashScreen	Eval
2184	XXX: hairy code that is basically just a functional(?) port of some other code i barely understood		Prune
513	TODO: need to add isJavaCastable() test for strictJava (as opposed to isJavaAssignable())	BSHCastExpression	Eval
581	TODO: Note: this trycatch block is copied from BSHMethodInvocation we need to factor out this common functionality and make sure we handle all cases ... (e.g. property style access, etc.) maybe move this to Reflect ?	BSHPrimarySuffix	doName
599	Catch the mismatch and continue to try the next Note: this is innefficient, should have na isAssignableFrom() that doesn't throw TODO: we do now have a way to test assignment in castObject(), use it?	BSHTryStatement	eval

Mais exemplos

Comentários Independentes

* TODO: This needs documentation! Is this really needed? Why?

* TODO: The code implementing this method is from 2003 (see issue 2171) - * mechanically integrated by tfmorris in May 2007. Needs to be * reviewed/updated.

Mais exemplos

Comentários sincronizados com o código

/* This should NOT be looking for a NamedElement,
 * since this is not always about the name of this
 * modelelement alone.*/

```
        /* This should NOT be looking for a NamedElement,
         * since this is not always about the name of this
         * modelelement alone.*/
        if (Model.getFacade().isANamedElement(own)) {
            final NotationName notation = Notation.findNotation(
                    getNotationSettings().getNotationLanguage());
            notationProviderName =
                NotationProviderFactory2.getInstance().getNotationProvider(
                        getNotationProviderType(), own, this,
                        notation);
        }
```

Mais exemplos

Comentários não sincronizados com o código

* TODO: FigEdgePort should be removed from the FigNodeModelElement * hierarchy and so the need for this removed.*/

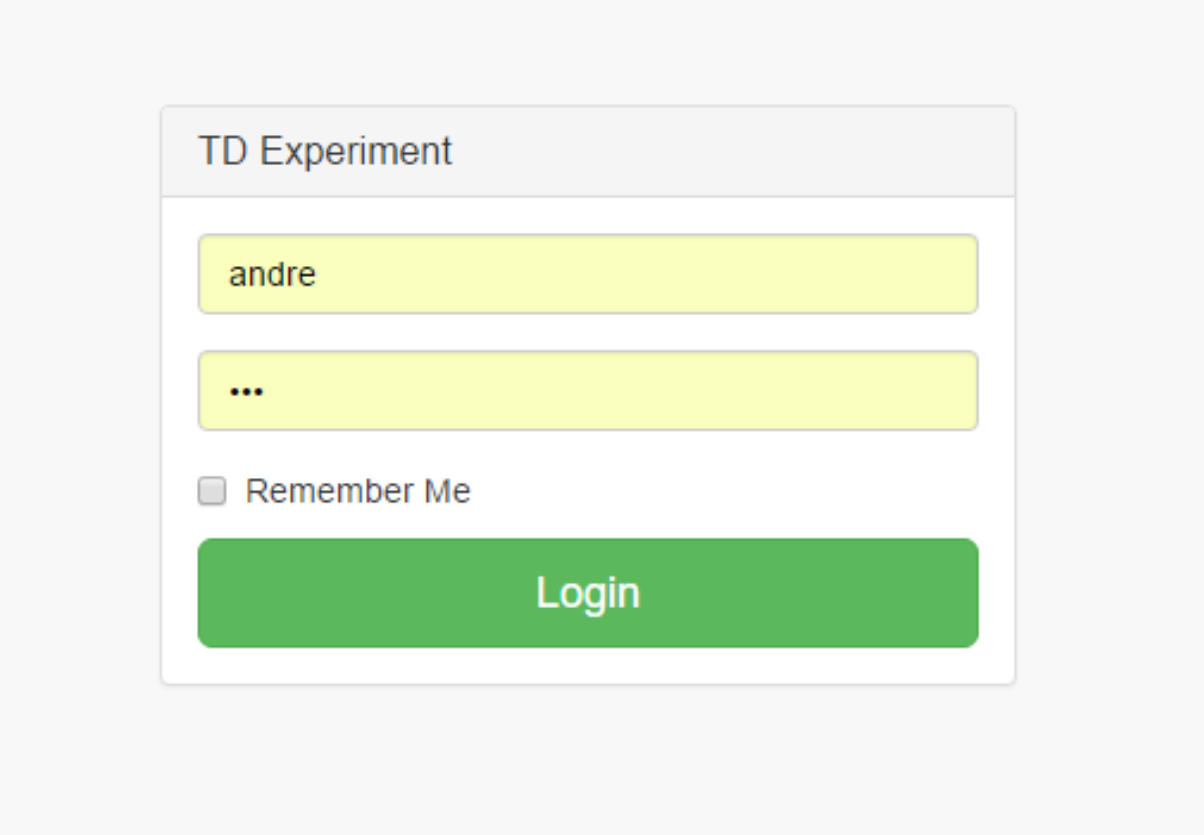
```
private NotationProvider notationProviderName;

/**
 * True if an instance is allowed to be
 * invisible. This is currently only set true by FigEdgePort.
 * TODO: FigEdgePort should be removed from the FigNodeModelElement
 * hierarchy and so the need for this removed.
 */
protected boolean invisibleAllowed = false;
```


// TODO real numbers not yet supported

```
// TODO real numbers not yet supported
//assertEquals(DefaultOclEvaluator.getInstance()
        //.evaluate(null, null, "1.5"), 1.5);
```

Experimento

Estudo



Referências

[1] Seaman, C. Measuring and Monitoring Technical Debt. Presentation at USP, 2013.
[2] W. Cunningham. The WyCash Portfolio Management System. ACM SIGPLAN OOPS Messenger (Vol. 4, No 2). ACM. December, 1992. pp. 29-30.
[3] Lim, Erin, Nitin Taksande, and Carolyn Seaman. "A balancing act: what software practitioners have to say about technical debt." *Software, IEEE* 29.6 (2012): 22-27.
[4] Alves, Nicolli S R and Ribeiro, Leilane F and Caires, Vivyane and Mendes, Thiago S and Spínola, Rodrigo O. Towards an Ontology of Terms on Technical Debt. International Workshop on Managing Techinical Debt. 2014.
[5] Zazworka, Nico and Spínola, Rodrigo O. and Vetro, Antonio and Shull, Forrest and Seaman, Carolyn. A case study on effectively identifying technical debt. Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering – EASE. 2013;

Perguntas

FinTD IV
Finding Technical Debt Experiment

Mário André (IFS/UFBA)
Thiago Souto (IFBA/UFBA)
Ph.D. Student in Computer Science
Prof. Dr. Manoel Mendonça (UFBA/Fraunhofer)
Prof. Dr. Rodrigo Spínola (UFBA/Fraunhofer)