GERAÇÃO DE TESTES DE ACEITAÇÃO EM FIT A PARTIR DE ESPECIFICAÇÕES EM B

IV WDRA

SBQS'2010

Thiago C. de Sousa (EP-USP) , Claudia de O. Melo (IME-USP)
thiago.carvalho@poli.usp.br, claudia@ime.usp.br

11/06/2010

Agenda

- Motivação
- Conceitos
  - Testes de aceitação e FIT
  - Métodos formais, B e ProB
- Proposta
- Trabalhos relacionados
- Solução B2Fit e seus mapeamentos
- Discussão e trabalhos futuros

Motivação

- Uma das tarefas de um projeto ágil é definir um conjunto de testes de aceitação em conjunto com o cliente.
  - isso pode ser difícil em sistemas críticos - muitas restrições/muitos casos de teste.
- Como ajudar equipes ágeis a lidar com sistemas críticos?

Conceitos::Testes de aceitação

- Testes de aceitação são testes formais conduzidos pelo cliente para determinar se o sistema satisfaz ou não seus critérios de aceitação, determinando também se o sistema deve ou não ser aceito [IEEE 1986].

Conceitos::Testes de aceitação

- Automatizar testes de aceitação pode ser uma abordagem para espeficar requisitos de forma executável e legível [Reppert 2004].
- A comunidade ágil incentiva a criação de testes de aceitação automatizados durante todo o desenvolvimento.

Conceitos::Fit

- **F**ramework for **I**ntegrated **T**ests.
- Ferramenta para especificação de testes de aceitação automatizados em forma tabular.
  - Clientes podem escrever os testes em tabelas HTML, Excel etc.

Conceitos::Fit – Tabelas Column

- *Column: cada cenário de teste em uma linha (entradas e saídas esperadas).*

eg.Division		
numerator	denominator	quotient()
1000	10	100.0000
-1000	10	-100.0000
1000	7	142.85715
1000	.00001	100000000
4195835	3145729	1.3338196

Conceitos::Fit – Tabelas Row

- *Cada linha é, em geral, um dos objetos resultantes de uma consulta.*

eg.music.Display					
title	artist	album	year	time()	track()
Handy Man	James Taylor	JT	1977	3.30	7 of 12
Sailing To Philadelphia	James Taylor	October Rose	2001	5.47	3 of 3
Ananas	James Taylor	Hourglass	1997	5.73	5 of 13
Another Grey Morning	James Taylor	JT	1977	2.73	4 of 12
Copperline	James Taylor	New Moon Shine	1991	4.37	1 of 12

Conceitos::Fit – Tabelas Action

- *Cada linha é uma ação, em geral voltada para interfaces de usuário* (ex: ENTER, CHECK).

fit.ActionFixture		
enter	select	1
check	title	Akila
check	artist	Toure Kunda
enter	select	2
check	title	American Tango
check	artist	Weather Report
check	album	Mysterious Traveller
check	year	1974
check	time	3.70
check	track	2 of 7

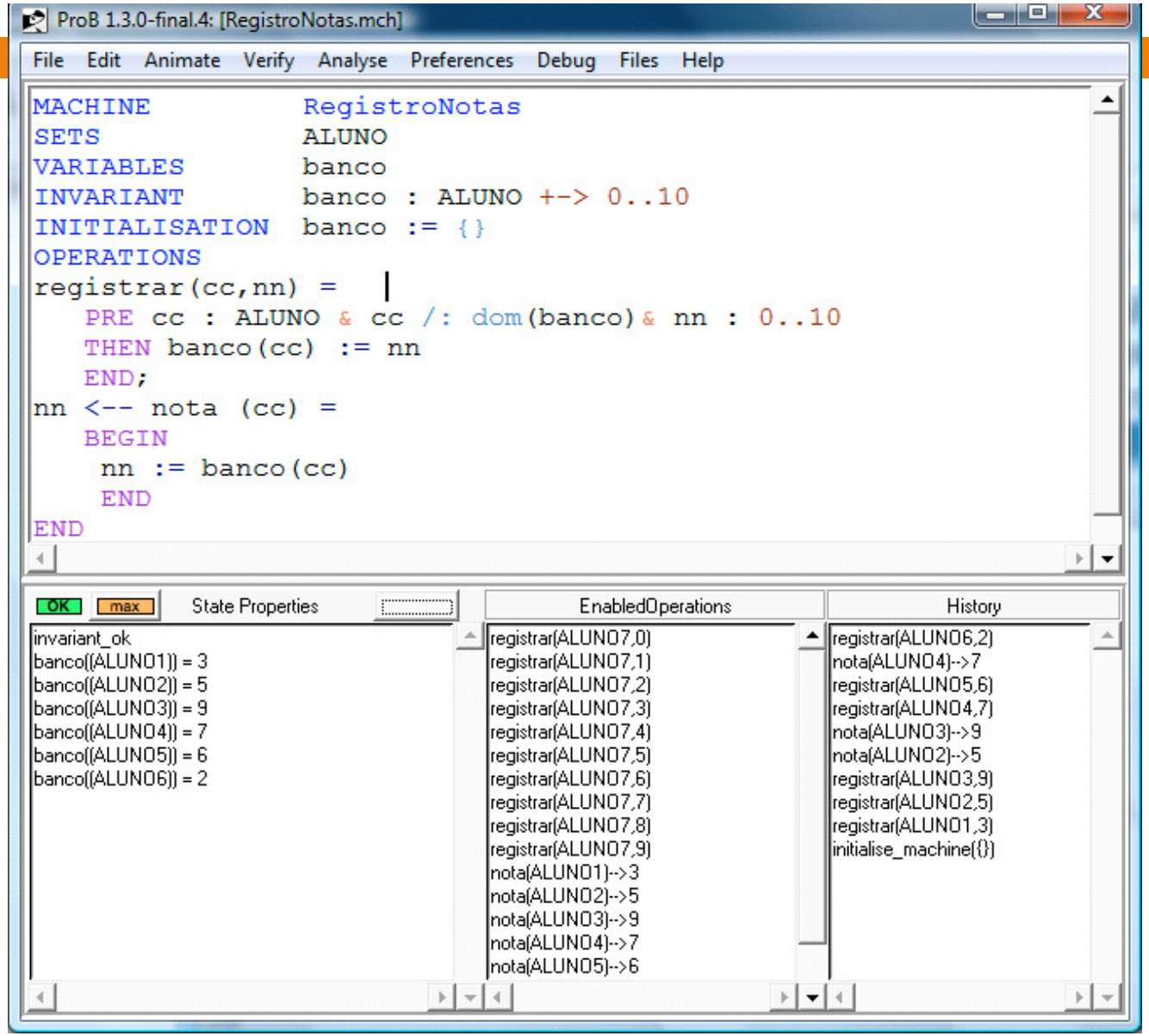
Conceitos::Métodos formais

- Técnicas matemáticas para especificação, desenvolvimento e verificação de software.
- Componentes de um método formal:
  - Linguagem de especificação/modelagem formal;
  - Sistema de raciocínio formal (para verificação/análise formal).

Conceitos::Método B

- Desenvolvido na década de 90 em Oxford para requisitos funcionais de sistemas criticos.
- Baseado em lógica de predicado, teoria dos conjuntos, aritmética de inteiros e substituições generalizadas.
- Forte aplicação industrial, especialmente sistemas metro-ferroviários.

Conceitos::ProB



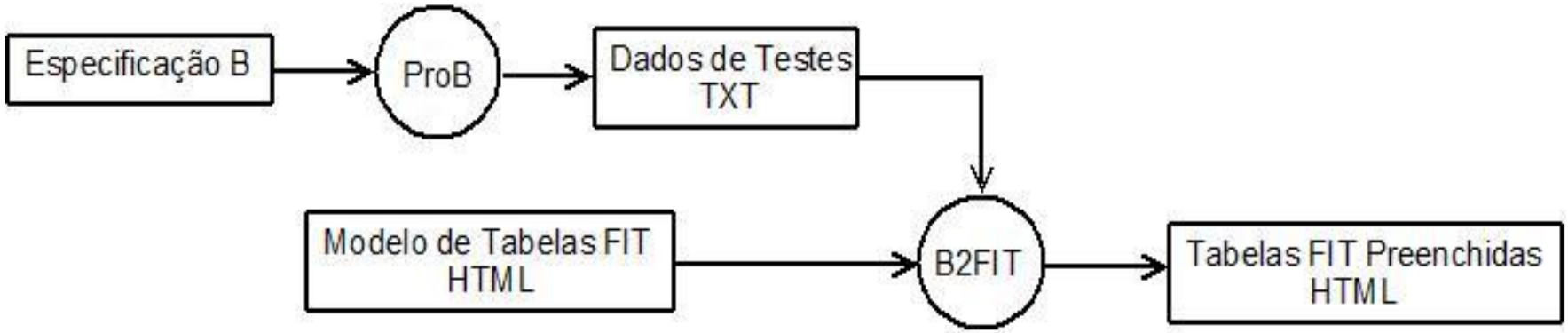
Proposta

- Utilizar especificações B como ferramenta auxiliar para a geração de testes de aceitação Fit.
- Aplicabilidade: apoiar o desenvolvimento de sistemas críticos por equipes ágeis.

Alguns trabalhos relacionados

- Métodos formais e testes são técnicas complementares para a redução de erros em sistemas [Bowen et al. 2002].
- Ferramentas de geração de testes podem melhorar trade-off entre restrições de tempo e qualidade em um projeto [Bacchelli et al. 2008].
- Método para geração de casos de testes a partir de especificações em B e a ferramenta ProB [Gupta e Bhatia 2010]. Não tem integração com ferramentas de testes de aceitação usadas por equipes ágeis.

Solução proposta:: B2Fit

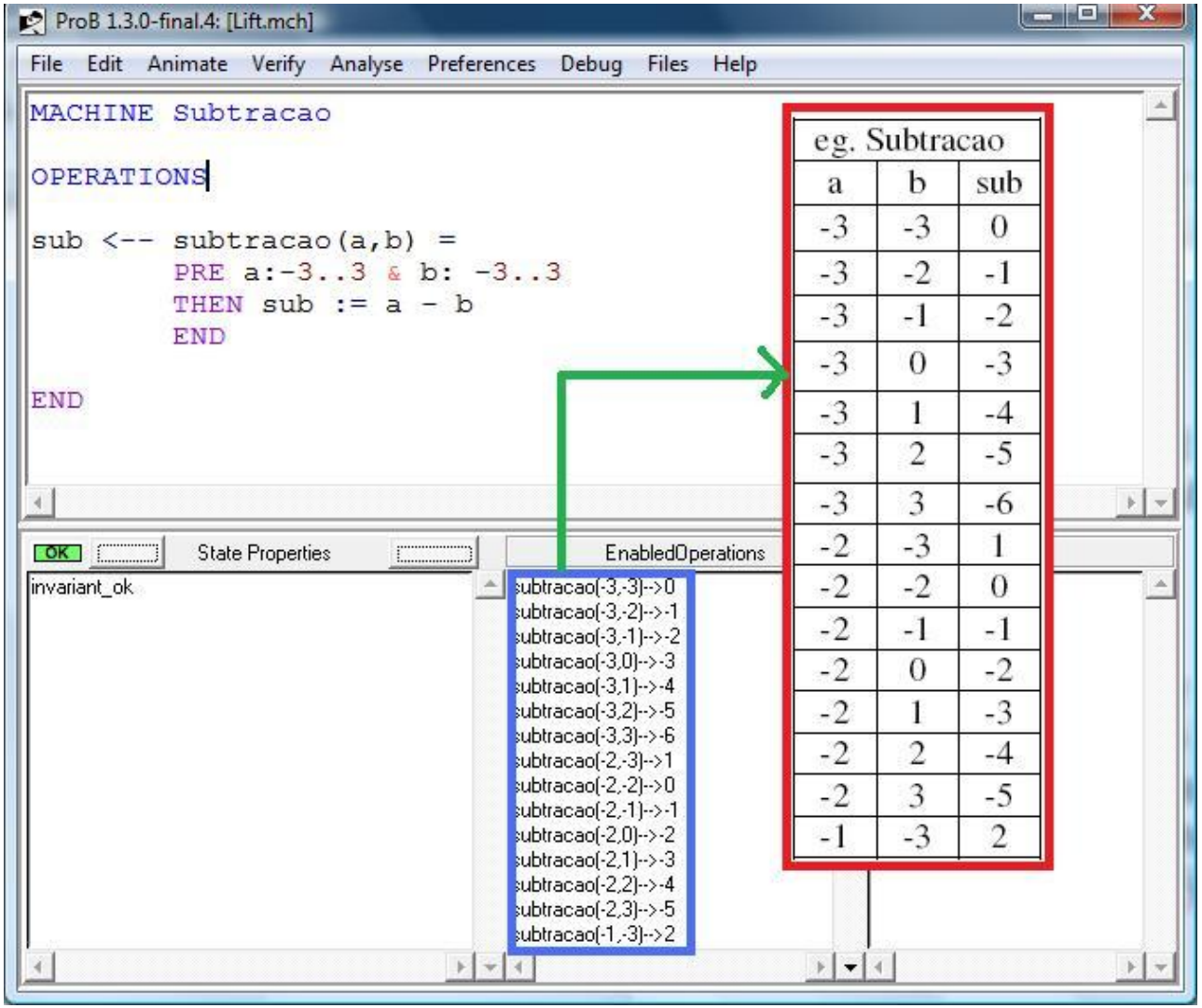


Padrão para *ColumnFixture*

- Usada quando se deseja verificar se uma dada função produz corretamente uma saída de acordo com as entradas fornecidas.

```
MACHINE OPFUN1

OPERATIONS OPFUN1

END
```

ColumnFixture:: Exemplo



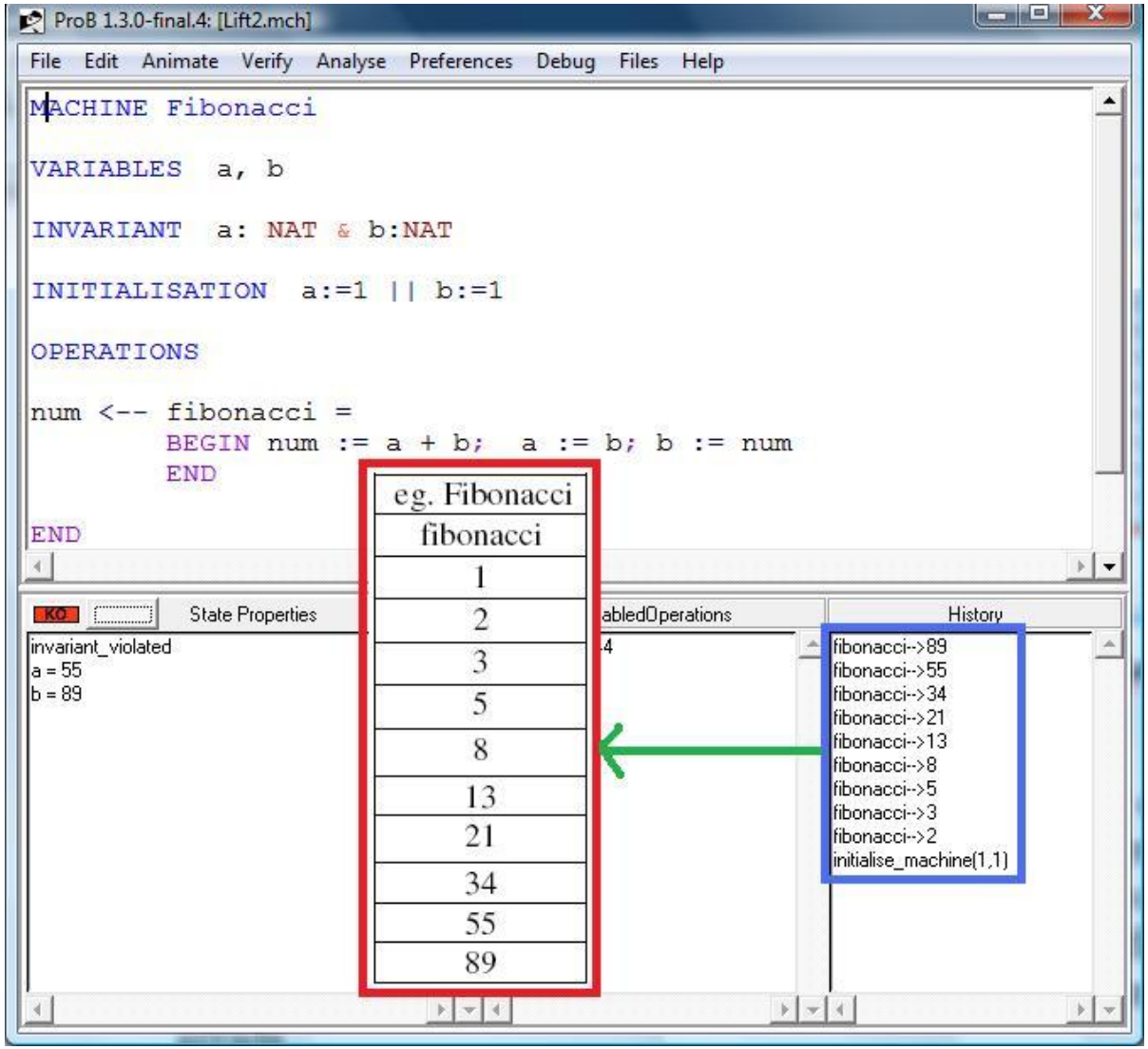
eg. Subtracao		
a	b	sub
-3	-3	0
-3	-2	-1
-3	-1	-2
-3	0	-3
-3	1	-4
-3	2	-5
-3	3	-6
-2	-3	1
-2	-2	0
-2	-1	-1
-2	0	-2
-2	1	-3
-2	2	-4
-2	3	-5
-1	-3	2

Padrão para *RowFixture*

- Utilizado quando se deseja testar consultas que devolvem um conjunto com um número exato de elementos.

```
MACHINE OPQUERY1

VARIABLES   XX

INVARIANT   YY

INITIALISATION   ZZ

OPERATIONS OPQUERY1

END
```

RowFixture:: Exemplo

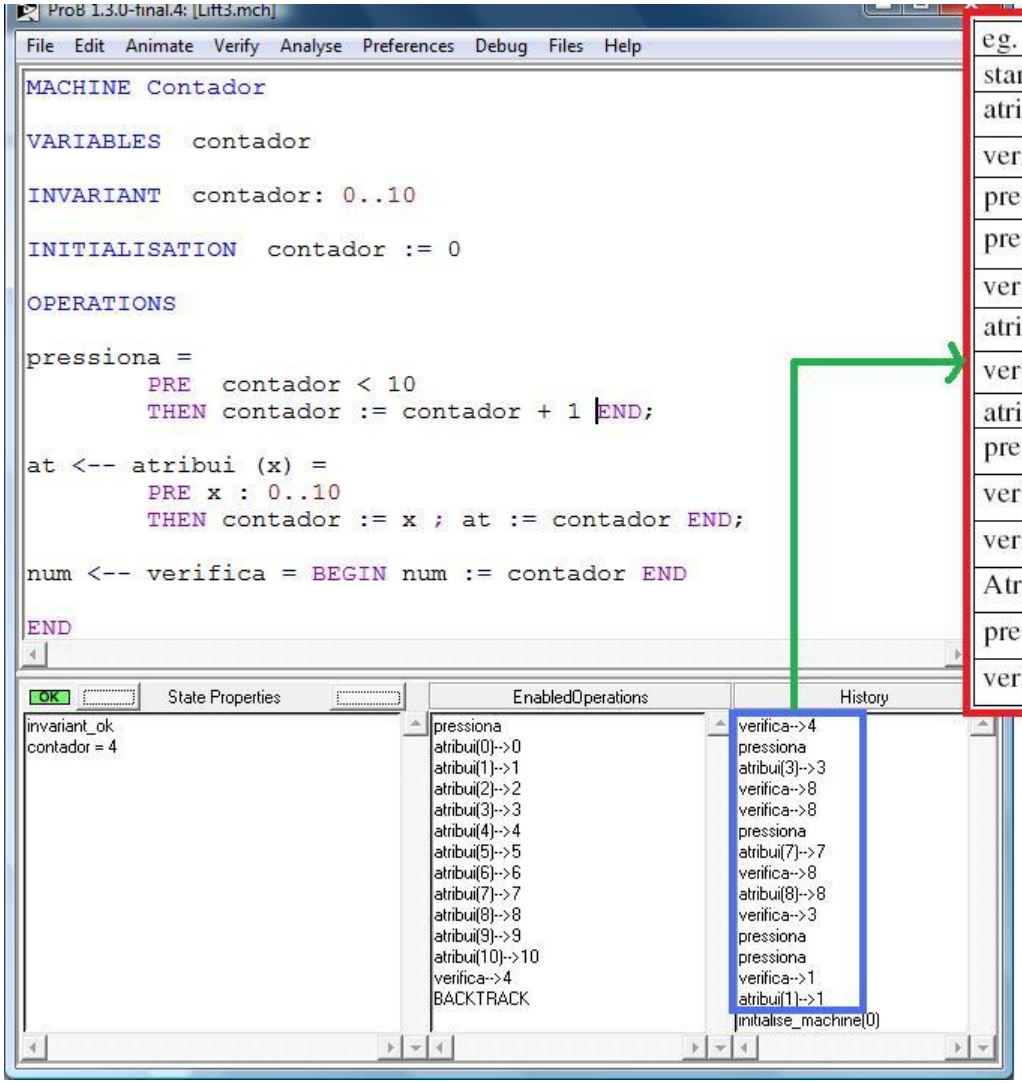


Padrão para *ActionFixture*

- Usada quando se precisa simular eventos ocasionados pelos usuários na interface do sistema.

```
MACHINE OPQUERY1

VARIABLES   XX

INVARIANT   YY

INITIALISATION   ZZ

OPERATIONS OPEVENT1 END; OPEVENT2 END; .....OPEVENTN END

END
```

ActionFixture:: Exemplo



eg. Contador		
start	contador	
atribui	at	1
verifica	num	1
pressiona		
pressiona		
verifica	num	3
atribui	at	8
verifica	num	8
atribui	at	7
pressiona		
verifica	num	8
verifica	num	8
Atribui	at	3
pressiona		
verifica	num	4

Discussão e trabalhos futuros

- Integrar métodos ágeis e métodos formais é uma ideia recente e promissora [Cohen e Money 2008][Black et al. 2009].
- Vantagens
  - Aumenta a possibilidade de aplicação de métodos ágeis em desenvolvimento de sistemas críticos;
  - Redução do tempo de especificação dos testes de aceitação;
  - Redução do tempo de aprendizado da notação B (mapeamento).

Discussão e trabalhos futuros

- O projeto está em fase inicial
  - Protótipo em desenvolvimento.
- Trabalhos futuros
  - Realizar um estudo de caso na indústria – empresa de sistemas críticos;
  - Inclusão de particionamento de equivalência e análise do valor limite;
  - Integração com outras ferramentas (ex: Fitnesse).

Referências

- Cohen, S. J. and Money, W. H. (2008) Bridge Methods: Complementary Steps Integrating Agile Development Tools and Methods with Formal Process Methodologies. In *Proc. of the 41st Annual Hawaii international Conference on System Science*. HICSS. IEEE Computer Society, Washington, DC, 460.
- Black, S., Boca, P. P., Bowen, J. P., Gorman, J., and Hinchey, M. (2009) Formal Versus Agile: Survival of the Fittest. *Computer* 42, 9, 37-45.
- IEEE Std 1012 (1986) IEEE Standard for Software Verification and Validation Plans.
- Reppert, T. (2004) "Don't Just Break Software, Make Software: How Story-Test-Driven-Development is Changing the Way QA, Customers, and Developers Work". *Better Software*, 6(6): 18–23.
- *Stotts, P. D., Lindsey, M., and Antley, A. (2002) An Informal Formal Method for Systematic JUnit Test Case Generation. In Proc of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods - Xp/Agile Universe 2002. Lecture Notes In Computer Science, vol. 2418. Springer-Verlag, London, 131-143.*
- *Gupta, A. and Bhatia, R. (2010) Testing functional requirements using B model specifications. SIGSOFT Software Engineering Notes 35, 2 , 1-7.*
- Bacchelli, A., Ciancarini, P., and Rossi, D. (2008) On the Effectiveness of Manual and Automatic Unit Test Generation. In Proceedings of the 2008 the Third international Conference on Software Engineering Advances. ICSEA. IEEE Computer Society, Washington, DC, 252-257.