

# Um Modelo para Predição da Confiabilidade baseado em Métricas de Software

Vagner Clementino

Departamento de Ciência da Computação  
Universidade Federal de Minas Gerais(UFMG)  
Software Quality and Measurement - 2015-1

# Agenda

---

Contexto

Problema

Objetivo

Modelo Proposto

Avaliação

Resultados

Trabalhos Relacionados

Ameaças à Validade

Conclusões e Trabalhos Futuros

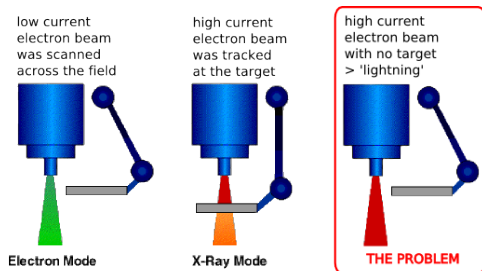
# Contexto

---

- ▶ Um sistema computacional é um artefato humano, e suas falhas, em última instância, representam a nossa incapacidade de compreender integralmente o comportamento de um sistema[6].
- ▶ Restrições de custo e tempo impendem a validação completa de um sistema.
  - ▶ Necessário a definição de metodologias, ferramentas e processos que visem priorizar as partes mais críticas de um software.

# Falhas são Perigosas

- ▶ Therac 25, uma máquina de terapia por radiação, que no ano de 1986 provocou graves lesões e mortes em pacientes devido a uma falha do seu software [5].




tray including the target, a flattening filter, the collimator jaws and an ion chamber was moved OUT for "electron" mode, and IN for "photon" mode.

# Falhas são Caras

---

- ▶ Manter ou desenvolver novas funcionalidades é bem mais caro que o custo de criação[8].

$$C_i^M = \gamma_0^i + \gamma_1^i M_i + \frac{1}{2} \gamma_2^i M_i^2 + c_3^i M_i \sum_{j=0}^{i-1} M_j.$$


Os três termos representam o custo de manter ou desenvolver novas características.

# Falhas podem dormir

- ▶ Um erro pode levar algum tempo até se manifestar

The screenshot shows a web browser window with the address bar displaying `time.com/3906908/skype-crash/`. The browser's address bar and tabs are visible at the top. Below the browser window, the Time magazine website is shown. The main article is titled "This Single Message Can Crash Skype" by Daniel Roberts for Fortune, dated June 3, 2015. The article's sub-headline is "And it's only 8 characters long". The text of the article begins with "If you're sending someone a link in Skype, be careful — a simple typo could crash the program for them." and continues with "These eight characters — <http://> — when used in any message, can crash Skype for the recipient and,". A large image of the Skype logo is visible on the right side of the article. The left sidebar of the website features a "MENU" button and a list of other articles, including "California Wildflowers Suffering From California Drought", "See the First Photo From Independence Day Resurgence", "5 Ways to Work and Live Abroad", "What You Need to Know About Negotiating Your First Salary", and "Gmail Now Officially Lets You 'Undo Send' Those Really Embarrassing E-mails". The bottom of the browser window shows a taskbar with open files: "01 - Introducao.pdf" and "Therac25.png".

TIME INC. NETWORK: TIME PEOPLE FORTUNE SI SPORTS ILLUSTRATED EW ENTERTAINMENT WEEKLY M MONEY G GOLF MORE

**TIME** Subscribe

LATEST MAGAZINE VIDEOS

**TECH SKYPE**

## This Single Message Can Crash Skype

Daniel Roberts / Fortune | June 3, 2015

**And it's only 8 characters long**

If you're sending someone a link in Skype, be careful — a simple typo could crash the program for them.

These eight characters — <http://> — when used in any message, can crash Skype for the recipient and,

01 - Introducao.pdf Therac25.png

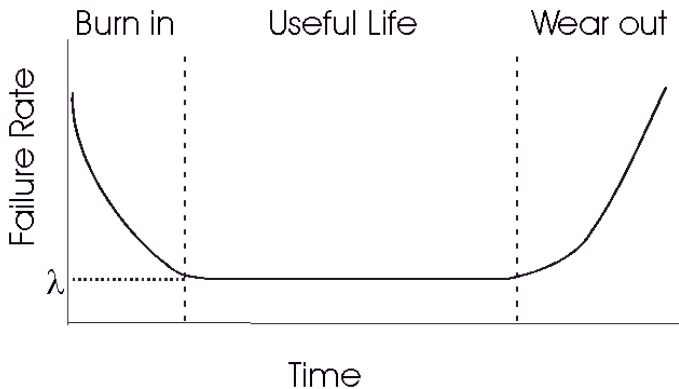
Mostrar todos os downloads...

# Confiabilidade de Software

---

- ▶ Conforme a ANSI é definida como a probabilidade de operação de um produto de software livre de falhas por um período de tempo em um ambiente.
- ▶ Possibilidade de classificar os módulos de um sistema a partir de sua **Confiabilidade**
- ▶ **Problema:** Baseado no tempo de execução do software.

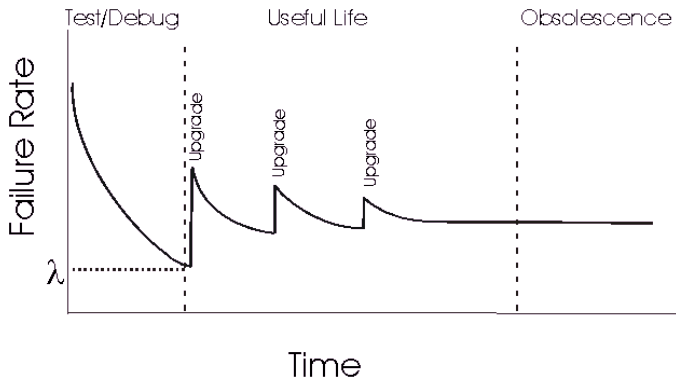
# Confiabilidade em Hardware



**Figura 1 :** Curva Bathtub para a Confiabilidade de Hardware.  
Adaptado de [6]



# Confiabilidade em Software



**Figura 2 :** Curva Bathtub para a Confiabilidade de Software.  
Adaptado de [6]

# Métricas de Software

---

- ▶ Utilizadas com o objetivo de mensurar a qualidade interna do software.
- ▶ Baseadas em Thresholds
- ▶ Fortemente dependente do paradigma e linguagem utilizada no sistema avaliado.

# Problemas

---

- ▶ Modelos de Confiabilidade são dependentes do tempo de execução.
- ▶ Métricas de Software são baseadas em threshold.
- ▶ Inexistência de um **Modelo de Predição da Confiabilidade** que considere valores das **Métricas de Software**

# Objetivo

---

- ▶ Propor **Modelo de Predição da Confiabilidade** que vise definir os módulos com o maior número de falhas utilizando **dados históricos** e **Métricas de Software**.

# Modelo Proposto

---

- ▶ Seja  $X_{vc} = [X_1, X_2, \dots, X_k]$  um vetor de característica de uma versão  $v$  de um componente  $c$  de um determinado sistema.
- ▶ Seja ainda  $\lambda$  a taxa média de falhas do componente  $c$ .
- ▶ Podemos definir uma função de característica  $\varphi_{vc}$  conforme Equação 1.

$$\varphi_{vc} = \sum_{j=1}^k \frac{1}{w_j} \times X_{jc} \quad (1)$$

# Modelo Proposto

---

- ▶ Na Equação 1 o valor de  $k$  representa o tamanho do conjunto de métricas;
- ▶  $w_j$  refere-se ao peso da métrica no cálculo da função de característica;
- ▶  $X_{jc}$  é o valor da métrica  $X_j$  obtida do componente  $c$ .

# Modelo Proposto

---

- ▶ A partir de  $\varphi_{vc}$  é possível calcular o número de erros em determinado módulo com base na Equação 2.

$$\omega = \lambda \times \frac{\varphi_{vc}}{\varphi_{(v-1)c}} \quad (2)$$

# Conjunto de Métricas

---

- ▶ Neste trabalho foi utilizado a suite **CK-Metrics** [2]
- ▶ Conjunto de Métricas largamente utilizado na literatura.
- ▶ Capaz detectar classes mais propensas a falhas [1]



# CK-Metrics

---

- ▶ *Weighted Methods per Class (WMC)*
- ▶ *Depth of Inheritance Tree (DIT)*
- ▶ *Number Of Children(NOC)*
- ▶ *Coupling Between Objects(CBO)*
- ▶ *Response For a Class (RFC)*
- ▶ *Lack of Cohesion on Methods (LCOM)*

# Avaliação do Modelo

---

- ▶ Coletado os dados de bugs de quatro sistemas implementados em Java
- ▶ Desenvolvidos pela Apache Software Foundation<sup>1</sup>
- ▶ Critério de escolha:
  - ▶ Base de usuários abrangente e ativa.
  - ▶ Um grande número de bugs registrados no BST da Apache Foundation
  - ▶ Pertencente a diferentes categorias de software.

---

<sup>1</sup><http://www.apache.org>

# Sistemas Utilizados

---

Produto	Descrição	Categoria	KLOC
<i>Ant</i>	Ferramenta utilizada para automação de compilação de software.	Gerenciamento da Compilação	133
<i>JMeter</i>	Ferramenta utilizada para testes de carga em um servidores, redes ou objetos Java.	Ferramenta de Testes	92
<i>Log4j</i>	API para que o desenvolvedor de software possa fazer log de dados na aplicação.	Biblioteca	15
<i>Tomcat 7</i>	Servidor web Java que implementa as tecnologias Java Servlet e JavaServer Pages.	Servidor Web	196

Tabela 1 : Sistemas utilizados na avaliação

# Dados Coletados

---

<b>Campo</b>	<b>Descrição</b>
<i>ID</i>	Identificador de um bug no ASF Bugzilla
<i>Situação</i>	Identifica o estado atual de um bug.
<i>Produto</i>	O sistema no qual o bug ocorreu
<i>Versão</i>	A versão do sistema em que o bug ocorreu
<i>Componente</i>	Identifica o módulo do sistema em que o bug ocorreu
<i>Hardware</i>	A plataforma de hardware no qual o erro foi observado.
<i>Importância</i>	A importância de um bug é descrita como a combinação de sua prioridade e gravidade.
<i>Target Milestone</i>	O campo Target Milestone identifica em qual versão do sistema o bug deverá estar selecionado.
<i>Atribuído Para</i>	Identifica o responsável pela resolução do bug.
<i>Data do Bug</i>	Contém a data em que o bug foi reportado.
<i>Relatado Por</i>	Identifica o responsável por informar o bug.
<i>Data da Última Alteração</i>	Contém a data da última alteração ocorrida na resolução do bug.
<i>Descrição do Bug</i>	Contém os detalhes do problema reportado.

**Tabela 2 :** Campos recuperados pelo *ASFBugScraper*<sup>2</sup>

---

<sup>2</sup>Sistema criado para realizar um

# Seleção dos Dados

---

- ▶ Considerados apenas os bugs confirmados
- ▶ Os bugs foi divididos em duas categorias CAT-HIST e CAT-LAST.
  - ▶ CAT-HIST - Bugs em versões diferente da última avaliada.
  - ▶ CAT-LAST - Bugs da última versão do sistema analisada.
- ▶ Falhas em CAT-HIST foram utilizados para de calcular a taxa de confiabilidade  $\omega$
- ▶ Os valores de  $\omega$  são comparados com os bugs em CAT-LAST.

# Categorização das versões dos sistemas

---

Sistema	Versões em CAT-HIST	Tamanho da Amostra	Versão em CAT-LAST	Tamanho da Amostra	Total Bugs
<i>Ant</i>	1.0; 1.1; 1.2; 1.3; 1.4.x 1.5.x; 1.6.x; 1.7.x; 1.8.x	3097	1.9.5	98	3195
<i>JMeter</i>	1.5; 1.7.x; 1.8.x; 1.9.x; 2.0.x 2.1.x; 2.2.x; 2.3.x; 2.4.x; 2.5.x; 2.6; 2.7; 2.8	1415	2.9	140	1555
<i>Log4j</i>	1.0; 1.1	224	1.2.18	539	763
<i>Tomcat 7</i>	7.0.0; 7.0.1; 7.0.2; 7.0.3; 7.0.4; 7.0.5; 7.0.6; 7.0.7; 7.0.8; 7.0.9; 7.0.10; 7.0.11; 7.0.12; 7.0.13; 7.0.14; 7.0.15; 7.0.16; 7.0.17; 7.0.18; 7.0.19; 7.0.20; 7.0.21; 7.0.22; 7.0.23; 7.0.24; 7.0.25; 7.0.26; 7.0.27; 7.0.28; 7.0.29; 7.0.30; 7.0.31; 7.0.32; 7.0.33; 7.0.34; 7.0.35; 7.0.36; 7.0.37; 7.0.38; 7.0.39; 7.0.40; 7.0.41; 7.0.42; 7.0.43; 7.0.44; 7.0.45; 7.0.46; 7.0.47; 7.0.48; 7.0.49; 7.0.50; 7.0.51; 7.0.52; 7.0.53; 7.0.54; 7.0.55; 7.0.56; 7.0.57	475	7.0.59	11	486

**Tabela 3 :** Categorização das versões dos sistemas

# Coletando as Métricas

---

- ▶ Métricas coletadas utilizando a ferramenta *ckjm*[4]
- ▶ Código compilado dos sistemas recuperados diretamente do site da Apache Foundation
- ▶ Métricas por classe vs Falhas por “módulo”
  - ▶ Distinção dos módulos pelo nome dos packages
  - ▶ Considerado a média das métricas das classe como o valor de um módulo.

# Resultados

Sistema	Componente	Taxa Média de Falhas	Versão	WMC	DIT	NOC	CBO	RFC	LCOM	$\varphi_{vc}$	$\frac{\varphi_{vc}}{\varphi_{vc}-11c}$	$\omega$
Ant	Core	16,96	1.9.4	8,705	0,521	0,421	7,687	26,515	63,983	107,831	1,027	17,420
			1.9.3	8,695	0,503	0,435	7,691	26,423	61,240	104,985		
	Core tasks	42,87	1.9.4	9,710	0,914	0,247	6,720	27,043	166,968	211,602	1,007	43,177
			1.9.3	9,688	0,914	0,247	6,720	26,989	165,538	210,097		
	Optional Tasks	20,18	1.9.4	11,311	1,410	0,049	4,492	25,754	87,852	130,869	1,000	20,180
			1.9.3	11,311	1,410	0,049	4,492	25,754	87,852	130,869		
Jmeter	Main	12,29	2.9	8,769	0,929	0,380	8,196	21,757	101,673	141,704	0,981	12,055
			2.8	8,900	0,949	0,377	8,383	22,057	103,801	144,467		
	HTTP	26,23	2.9	7,793	0,868	0,245	9,199	22,852	53,926	94,883	1,015	26,618
Log4j	Configurator	13,14	2.8	7,691	0,866	0,246	9,188	22,567	52,942	93,501		
			1.2.17	7,308	1,000	0,000	8,077	28,154	56,308	100,846	1,115	14,654
	Appender	44	1.2.16	6,500	1,000	0,000	8,357	25,714	48,857	90,429		
			1.2.17	10,967	0,600	0,533	6,667	30,367	38,800	87,933	0,976	42,957
Tomcat	Catalina	7,42	1.2.16	10,862	0,621	0,517	7,172	30,069	40,828	90,069		
			7.0.59	9,121	0,121	0,788	3,455	22,152	54,727	90,364	0,947	7,027
			7.0.57	9,581	0,129	0,774	3,484	23,258	58,194	95,419		

Tabela 4 : CK Métricas e o valor de w



# Resultados

---

Sistema	Componente	Média	Total Erros	Erros Encontrados(%)	$\omega$	Erros Previstos (%)
Ant	Core	16,96	7	23,33%	17,42	21,57%
Ant	Core tasks	42,87	20	66,67%	43,177	53,45%
Ant	Optional Tasks	20,18	3	10,00%	20,18	24,98%
Jmeter	HTTP	12,29	27	19,29%	12,055	31,17%
Jmeter	Main	26,23	113	80,71%	26,618	68,83%
Log4j	Appender	13,14	7	87,50%	14,654	25,44%
Log4j	Configurator	44	1	12,50%	42,957	74,56%
Tomcat 7	Catalina	7,42	6	-	7,027	-

**Tabela 5 :** Valores Coletados x Valores Previstos

# Resultados - Ant

---

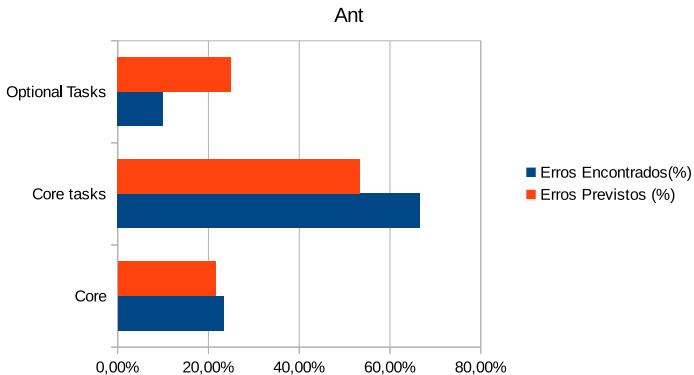


Figura 3 : Resultados Sistema Ant

# Resultados-JMeter

---

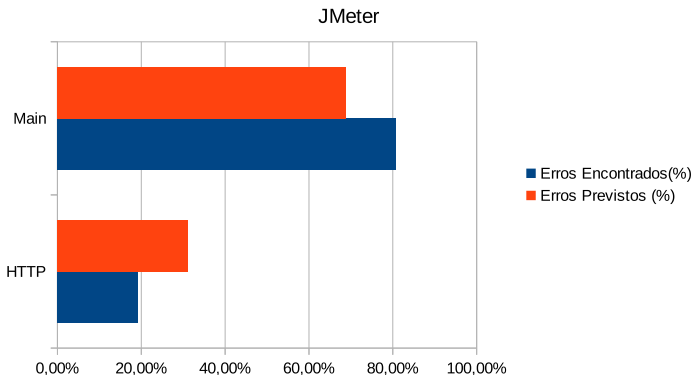


Figura 4 : Resultados Sistema JMeter

# Resultados-Log4J

---

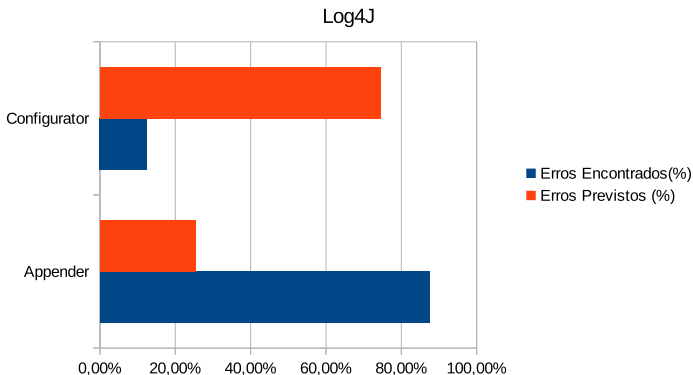


Figura 5 : Resultados Sistema Log4J

# Trabalhos Relacionados

---

- ▶ Em *Chidamber et al.*[2] é proposto um conjunto de métricas para a análise de sistemas orientados a objeto.
- ▶ No trabalho de *Basili et. al.* [1] as métricas proposta por [2] foram utilizadas para buscar classes mais propensas a falhas.

# Trabalhos Relacionados

---

- ▶ Em *Eaddy et al.* [3] é discutido o efeito de Crosscutting Concerns nas falhas dos sistemas com base em dados recuperados de um repositório de software.
- ▶ Em *Salem et al.*[7] foi utilizado a técnica de regressão logística com o objetivo de prever falhas.

# Ameaças à Validade

---

- ▶ Medição da taxa de Confiabilidade em uma granularidade ao nível de módulo.
- ▶ Utilização da média das métricas.
- ▶ Número de sistemas avaliados bem como a linguagem utilizado.
- ▶ Os sistemas foram desenvolvidos pela mesma organização.
- ▶ As falhas foram “aceitas” por diferentes desenvolvedores.

# Trabalhos Futuros

---

- ▶ Este trabalho demonstra a importância do uso de métricas de software na análise da qualidade interna do produto de software.
- ▶ O uso de dados históricos junto com métricas de software podem aumentar a qualidade dos modelos de predição de falhas.



# Trabalhos Futuros

---

- ▶ Refinar o modelo visando uma melhor predição.
- ▶ Coletar maiores informações para testar o modelo ao nível de classe.
- ▶ Aplicar o modelo em softwares de outras linguagens e organizações.

# Dúvidas

---



# References I

---

- [1] V. R. Basili, L. C. Briand, and W. L. Melo, “A validation of object-oriented design metrics as quality indicators,” *Software Engineering, IEEE Transactions on*, vol. 22, no. 10, pp. 751–761, 1996.
- [2] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,” *Software Engineering, IEEE Transactions on*, vol. 20, no. 6, pp. 476–493, 1994.

# References II

---

- [3] M. Eaddy, T. Zimmermann, K. D. Sherwood, V. Garg, G. C. Murphy, N. Nagappan, and A. V. Aho, “Do crosscutting concerns cause defects?” *Software Engineering, IEEE Transactions on*, vol. 34, no. 4, pp. 497–515, 2008.
- [4] M. Jureczko and D. Spinellis, *Using Object-Oriented Design Metrics to Predict Software Defects*, ser. Monographs of System Dependability. Wroclaw, Poland: Oficyna Wydawnicza Politechniki Wroclawskiej, 2010,

# References III

---

vol. Models and Methodology of System Dependability, pp. 69–81.

- [5] N. G. Leveson and C. S. Turner, “An investigation of the therac-25 accidents,” *Computer*, vol. 26, no. 7, pp. 18–41, 1993.
- [6] M. R. Lyu, Ed., *Handbook of Software Reliability Engineering*. Hightstown, NJ, USA: McGraw-Hill, Inc., 1996.

# References IV

---

- [7] A. M. Salem, K. Rekab, and J. A. Whittaker, “Prediction of software failures through logistic regression,” *Information and Software Technology*, vol. 46, no. 12, pp. 781–789, 2004.
- [8] Y. Tan and V. S. Mookerjee, “Comparing uniform and flexible policies for software maintenance and replacement,” *Software Engineering, IEEE Transactions on*, vol. 31, no. 3, pp. 238–255, 2005.