Laboratório de Engenharia de Software

INF2134 Teste e medição de software
Organização

Arndt von Staa
arndt *at* inf.puc-rio.br
Departamento de Informática
2014/2
site: www.inf.puc-rio.br/~inf2134

Laboratório de Engenharia de Software

More than the act of testing, the act of designing tests is one of the best bug preventers known. [B. Beizer]

Objetivo da disciplina

- Avaliar, criticar, pesquisar e desenvolver assuntos relacionados com:
  - Engenharia de Software Experimental
    - condução de experimentos
    - medição de software
  - Controle da qualidade de software
    - aspectos formais leves voltados para testes
    - auto-verificação
    - testes

Motivação

- Independentemente de quão bem trabalhemos
  - software conterá defeitos
    - geram erros endógenos
  - estará sujeito a erros provocados por
    - outros artefatos, ex.
      - geram erros exógenos
      - software de terceiros, ex. bibliotecas
      - plataforma → hardware, sistema operacional, ...
      - transmissões via rede
      - sensores, atuadores
    - humanos
      - cometem erros de uso fortuitos
      - falta de atenção, ignorância, negligência
      - induzidos por interfaces humanas ruins
    - agressões
      - uso indevido intencional
      - fraude

Motivação

- Desejamos desenvolver e manter software de qualidade
  - qualidade satisfatória
- Para isso precisamos
  - Processos e práticas de desenvolvimento e manutenção
    - reduzir, por construção, a injeção de defeitos
  - Técnicas de controle da qualidade
    - ao desenvolver
    - ao manter
    - ao executar
  - Técnicas de especificação e modelagem
  - Técnicas de arquitetura e projeto
- Para avaliar sistematicamente as ferramentas, técnicas, métodos e processos propostos precisamos realizar
  - Engenharia de software experimental

Avaliação sistemática

- Engenharia de Software Experimental
  - avaliar através de experimentos sistemáticos as características do software, a eficácia e a eficiência dos testes e das ferramentas, práticas, técnicas, e processos de desenvolvimento em uso ou propostos
  - exemplos de formas de avaliar
    - estudos (*surveys*) sistemáticos da literatura
    - pesquisa de opinião (*surveys*) sistemáticos
    - provas de conceito
    - estudos de caso
    - experimentos controlados

Avaliação sistemática

- Medição
  - desenvolver, avaliar e selecionar métricas relevantes
    - medição de artefatos
    - medição de processos
  - Perguntas básicas
    - por que e para que medir?
    - o que medir?
    - como medir de forma confiável?
    - quando medir?
    - quem deve medir?
    - como divulgar o que foi medido e concluído a partir dessas medições?

Testes

- Testes são
  - experimentos controlados
    - verificam a corretude do resultado obtido por meio de um oráculo
  - ineficientes
    - caros, realizados repetidas vezes
      - precisam ser automatizados para tornarem-se econômicos
    - encontram poucos problemas por rodada
  - ineficazes
    - encontram somente uma parcela de todos os problemas existentes
  - requerem criatividade
    - geração de casos de teste
    - criação de armaduras ("*scaffolds*") de teste
  - apesar disso são necessários

Testes

Laboratório de Engenharia de Software

- Como tornar testes
  - mais eficientes?
  - mais eficazes?
  - menos aborrecidos?
  - menos custosos?
  - menos . . .
- Oximoro: como testar sem realizar (criar) testes? (hem?)

Testes

Laboratório de Engenharia de Software

- Como tornar programas amigáveis aos testes?
  - testabilidade de requisitos
  - testabilidade de arquitetura e projetos
  - testabilidade da implementação
  - repetibilidade dos testes
- Como tornar programas amigáveis à correção?
  - diagnosticáveis
  - depuráveis
  - robustos
  - tolerantes a defeitos
- Como tornar programas auto-verificantes?
  - os programas que verificam continuamente a si próprios
  - mesmo quando em uso produtivo

Bibliografia da disciplina, livros

Beck, K.; *Test-Driven Development by Example*; Addison-Wesley; 2003

Borba, P.; Cavalcanti, A.; Sampaio, A.; Woodcock, J.; eds.; *Testing Techniques in Software Engineering*; LNCS 6153; Berlin: Springer, Lecture Notes in Computer Science; 2010

Cockburn, A.; *Escrevendo Casos de Uso Eficazes - Um Guia para Desenvolvedores de Software*; São Paulo, SP: Bookman; 2005

Delamaro, M.E.; Maldonado, J.C.; Jino, M.; *Introdução ao Teste de Software*; Elsevier / Campus; 2007

Fewster, M.; Graham, D.; *Software Test Automation*; Addison-Wesley; 1999

Glass, R.L.; *Facts and Fallacies of Software Engineering*; Addison-Wesley; 2003

Huizinga, D.; Kolawa, A.; *Automated Defect Prevention: Best Practices in Software Management*; Hoboken, New Jersey: John Wiley & Sons; 2007

Hunt, A.; Thomas, D.; eds.; *Pragmatic Unit Test: in Java with JUnit*; Sebastopol, CA: O'Reilly; 2003

Hunt, A.; Thomas, D.; *O programador pragmático*; Artmed; 2010

Juristo, N.; Moreno, A.M.; *Basics of Software Engineering Experimentation*; Dordrecht: Kluwer; 2001

Kan, S.H.; *Metrics and Models in Software Quality Engineering*; Addison-Wesley; 1995

Bibliografia da disciplina, livros

Kaner, C.; Falk, J.; Nguyen, H.Q.; *Testing Computer Software*; 2nd edition; London: Thomson; 1993

Lewis, W.; *Software Testing and Continuous Quality Improvement*; Auerbach; 2000

McGregor, J.D.; Sykes, D.A.; *A Practical Guide to Testing Object-Oriented Software*; Reading, Massachusetts: Addison-Wesley; 2001

Metzger, R.C.; *Debugging by Thinking: A Multidisciplinary Approach*; Elsevier; 2003

Myers, G.J.; *The Art of Software Testing*, 2nd edition; Hoboken, New Jersey: John Wiley & Sons; 2004

Pezzè, M.; Young, M.; *Teste e Análise de Software*; Porto Alegre, RS: Bookman; 2008

Software Measurement Guidebook, Software Engineering Laboratory, Maryland, 1995

Splaine, S.; Jaskiel, S.P.; *The Web Testing Handbook*; STQE Publishing; 2001

Staa, A.v.; *Programação Modular*; Rio de Janeiro: Campus; 2000

Wohlin, C.; Runeson, P.; Höst, M.; Ohlsson, M.C.; Regnell, B.; Wesslén, A.; *Experimentation in Software Engineering An Introduction*; Kluwer; 2000

Bibliografia

- Periódicos e conferências
  - ACM Software Engineering Notes
  - IEEE Software
  - IEEE Transactions on Software Engineering
  - Journal of Systems and Software
  - Software Testing, Verification and Reliability
- Artigos buscados na rede

Critério de avaliação

- Participação ativa nas aulas
  - interação durante a aula
  - presença
  - pontualidade
- 3 seminários no decorrer do semestre sobre assunto escolhido pelo aluno
  - Apresentados em *powerpoint* (`.ppt` ou `.pptx`)
  - Arquivo enviado para mim antes do meio-dia do dia da apresentação
- Relatório técnico, complementação dos seminários
  - Ideal seria um artigo para o SBES
  - Texto ***word*** (`.doc ou .docx`) em formato monografia do DI
  - A ser entregue até 23 de janeiro de 2015

Exemplos de assuntos

- *Acceptance test driven development*
- *Model driven testing*
- *Metrics driven development* :)
- Avaliação do uso de técnicas formais leves
- Instrumentação do software, *Recovery oriented software*
- Teste de software distribuído
- Processo de controle da qualidade interdependente com o de desenvolvimento
- Automação da inspeção: verificação e medição estática
- Geração automatizada das suítes de teste (coletâneas de casos de teste)
- Automação da execução de testes
  - para GUI – desktop, laptop
  - avaliação de ferramentas através de exemplos
    - evitem estudar coisas já bastante exploradas e conhecidas: JUnit e similares
- Apoio ao diagnóstico e à depuração
- Estudo de fatores de risco e *risk driven testing*
- Testes não funcionais: teste de invasão; teste distribuído; teste de carga, **...**
- **...**

Seminário 1

- Apresentações de cerca de 15 minutos para cada um
  - apresentação + perguntas
- Consta de um esboço da proposta do trabalho:
  - O assunto: por que é um assunto interessante?
  - Motivação:
    - O que você pretende fazer ou aprender? Exemplos: estudo de caso, experimentos, resenha da literatura, observação do estado da prática, estudo teórico, modelagem
    - Qual é a experiência profissional que você tem no assunto?
    - O que já leu a respeito?
  - Roteiro de trabalho ou estudo até o segundo seminário
  - Bibliografia preliminar proposta
- OBSERVAÇÃO: Procurem seguir, no que der, o padrão de proposta de tese que está na página da disciplina.

Seminário 2

- Apresentações de cerca de 20 minutos
  - complementar ao 1o. seminário
- Versão em formato de proposta de trabalho
  - Introdução, motivação ➔ deve ser bem curto
  - Proposta
    - o que será realizado
  - Discussão
    - confronto com o estado da arte ou da prática, o que existe?
    - por que vale a pena o estudo / trabalho
  - Plano de ação para o restante do semestre
  - Esboço (*outline*) do documento final
  - Bibliografia: espera-se que o aluno tenha lido uma boa parte dela
- OBSERVAÇÃO: Sigam o padrão de proposta de tese que está na página da disciplina. Se for o caso, acrescentem críticas a esse documento.

Seminário 3

- Duração de cerca de 40 minutos mais 5 para perguntas
- Apresentação similar a uma apresentação de trabalho em um congresso ou simpósio
  - identificação: título, autor, data
  - motivação
    - assunto
    - problema específico abordado dentro deste assunto
    - estado da arte / estado da prática
  - solução / elaboração
    - suficientemente detalhada para que um não especialista entenda
  - avaliação da solução / elaboração
    - pontos fortes
    - pontos fracos
  - conclusão
    - principais resultados / contribuições / o que foi aprendido
    - pontos em aberto que merecem ser examinados no futuro
  - bibliografia em formato padrão de dissertação ou tese da PUC-Rio
  - folha de encerramento: autor, e-mail, título da apresentação

Relatório final

- O relatório final (ensaio, monografia, artigo) deve ser redigido no formato padrão de monografia do DI (entre **7500 e 8500 palavras**, não contam pré-texto – título, resumo, abstract, tabelas de conteúdo – nem referências bibliográficas)
  - deve obedecer a regras de "não plágio", algumas referências:
    - http://www-cse.ucsd.edu/users/baden/Honesty.html
    - Collberg, C.; Kobourov, S.; "Self-Plagiarism in Computer Science"; *ACM Communications* 48(4); New York, NY: ACM Association for Computing Machinery; 2005; pags 88-94
    - http://www.netshaq.com/cgiproxy/nph-proxy.cgi/011100A/http/www.acm.org/pubs/plagiarism%20policy.html
  - Deve ser entregue até 23/janeiro/2015

Relatório final

- Será avaliado como se fosse um "paper" submetido a um congresso:
  - contribuição, originalidade (não precisam ser grandes ☺)
    - no caso de resenha, interessam a forma e a abrangência
    - deve deixar claro que foi feito um estudo → existe crítica, contraposição
    - não basta narrar ou fazer uma colagem a partir de outros textos, é necessário mostrar que entendeu o assunto abordado tanto do ponto de vista técnico (ex. uso de ferramenta) como do ponto de vista conceitual (ex. conceitos que governam a ferramenta)
    - não basta relatar como funciona a prática, precisa contrapor com o que é relatado na literatura: por que foi escolhida esta prática ou ferramenta? quais as alternativas que existem? o que foi observado ao usá-la e como se compara a benchmarks ou outros relatos
  - abrangência do estado da arte e do domínio do problema
  - qualidade técnica
  - qualidade da organização do texto
  - qualidade do texto
    - ortografia, sintaxe, clareza de exposição, formatação ...
  - referências bibliográficas
    - no formato exigido pela PUC
    - em um volume razoável
    - citações contidas no texto

Referências

- Boehm, B.W.; Basili, V.R.; "Software Defect Reduction Top 10 List"; *IEEE Computer* 34(1); Los Alamitos, CA: IEEE Computer Society; 2001; pags 135-137
- Fox, A.; "Toward Recovery-Oriented Computing"; Invited talk; VLDB 2002; Buscado em: Março 2006; *URL*: www.cs.ust.hk/vldb2002/VLDB2002-proceedings/papers/S25P01.pdf
- Glass, R.L.; *Facts and Fallacies of Software Engineering*; New York, NY: Addison-Wesley; 2003
- Xinlong Bao; *Software Engineering Failures: A Survey*; School of EECS, Oregon State University, Corvallis, OR, U.S.A; apud Huckle, T.; Collection of Software Bugs; http://www5.in.tum.de/~huckle/bugse.html; last update October 5, 2007.

Perguntas?

Arndt von Staa

Departamento de Informática, PUC-Rio

Sala RDC 420, ramal DI 4333

arndt *at* inf.puc-rio.br