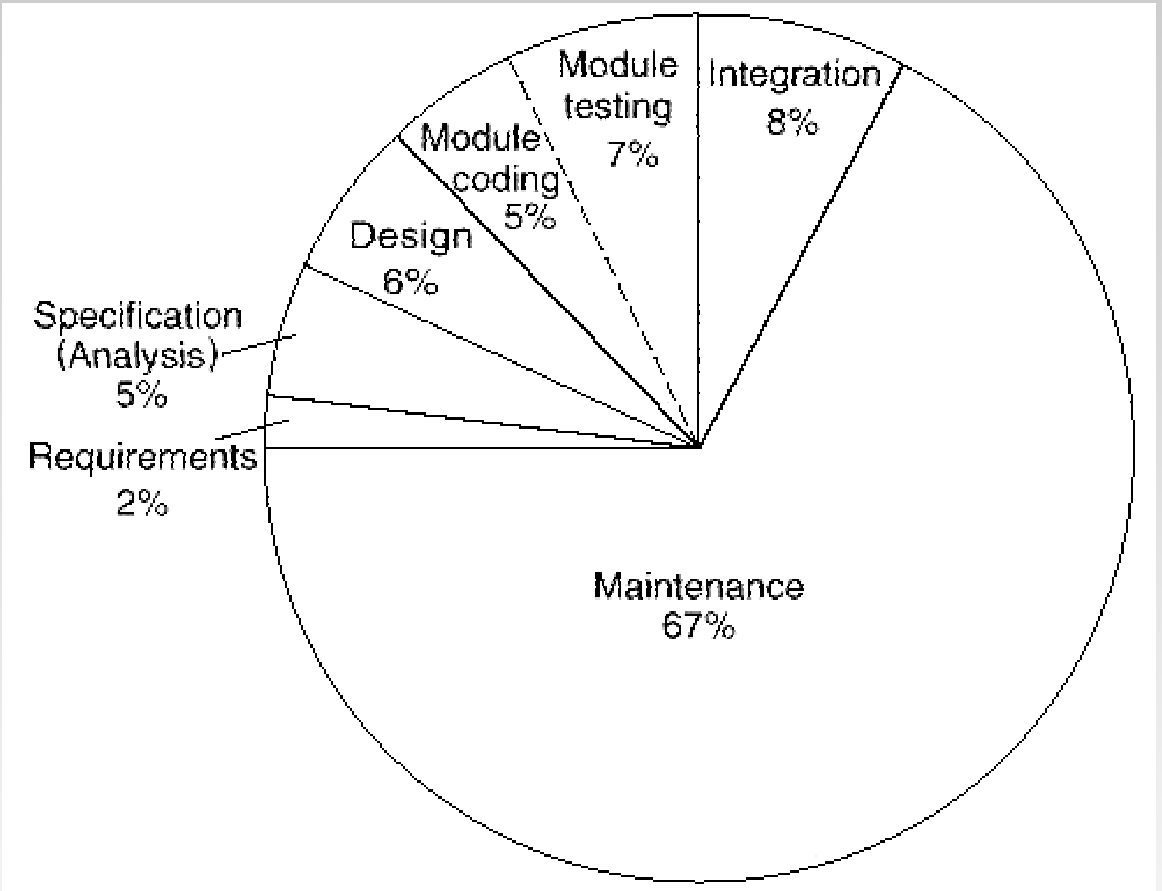
JUnit-SE

Fortaleza
4/9/2003

Motivação



Stephen Schach,
"Classical and Object-oriented Software Engineering",
Mc Graw Hill, 1999

Tipos de Testes

- Interface
  - Java Script
- Escalabilidade
  - Teste de Carga
- Funcionalidade
  - Conformidades
  - Falhas de Modelagem
- Configuração
  - Novas versões
  - Múltiplos Ambientes

Escalabilidade

- Estabilidade com o aumento de carga
  - Funciona com 1 e funciona com 1000
  - Arquitetura robusta
  - Testes requerem paralelismo
  - Testes devem envolver diversos clientes
  - WebApplication Stress Tool
  - Carga demasiada da Base de Dados

Funcionalidade

- Erros mais perigosos
  - Muitas vezes invisíveis e somente percebidos tardiamente
  - Podem gerar inconsistências irrecuperáveis
- Aderência às especificações
- Base para elaboração e validação de testes
  - Contrato de Requisitos
  - Diagramas UML
- Rotinas compartilhadas
  - alterações em um ponto afetam o funcionamento do sistema em outros

"Síndrome do Lençol Curto"

Mundo Ideal

- Testar novamente todas as rotinas e com todas as possibilidades sempre que qualquer alteração for realizada no sistema
- Nenhum erro deve ser detectado pelo usuário final

Testes Manuais:

- Processo consumidor de tempo e enfadonho
- Concentramos os testes onde “achamos” que nossas alterações irão ter impacto
- Processo não cumulativo

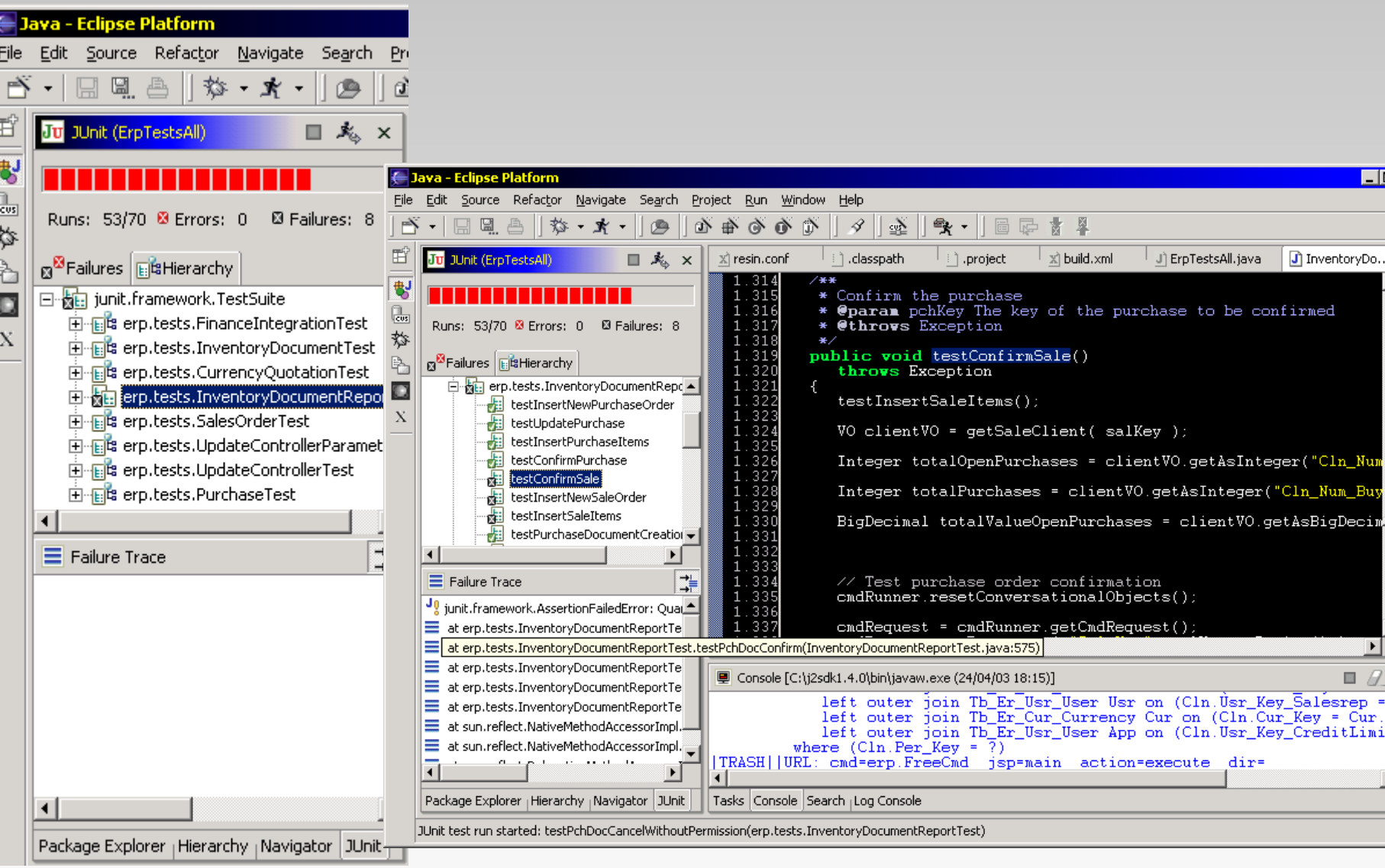
JUnit

- Filosofia de Desenvolvimento criada por um dos gurus da Engenharia de Software (Erich Gama)
- Framework de Testes contemplando biblioteca de classes e módulos de execução de testes
- Rotinas de testes são elaboradas pelo próprio programador: "Eat your own dog food"
- Processos são elaborados tendo em mente a execução de testes automatizados:
  - Interface deve ser separada dos processos (model-view-controler)
  - Mecanismos especiais podem ser necessários para dar suporte às rotinas de testes
  - Rotinas de testes substituem a interação via telas (o retorno do investimento já é possível nessa fase)
  - Testes gerais devem ser realizados com certa frequência para detectar erros de integração

JUnit

- Testes são cumulativos
  - Cada nova rotina codificada passa a fazer parte do projeto e será executada a cada teste geral de integração
- Testes são refinados
  - Sempre que for identificado um bug que as rotinas de teste atuais não descobriram, escrever um teste novo que detecte o problema.
  - Usar essa rotina para consertar o bug.
- Passos Sustentáveis
  - O uso contínuo das rotinas de testes garante que a implementação de funcionalidades novas não irá comprometer outras funcionalidades
  - Maior segurança ao efetuar alterações complexas em rotinas existentes

JUnit + Eclipse



JUnit-SE

- Funcionalidades
  - Classes Bases incorporadas ao Framework
  - Esquema transacional permitindo UNDO
  - Próximos Passos: Gravador de Script
- Interface
  - HttpUnit: simulando um browser
  - Próximos Passos: FW JavaScript
- Carga e Configuração
  - Rotinas ainda sem Framework

JUnit-SE

- Adoção formal da metodologia:
  - Documentação (Convenções de Testes)
  - Processo a mais para ser verificado pelo Guardião
  - Automatização de Testes:
    - Atualiza versões a partir do CVS
    - Recompila o projeto inteiro
    - Executa todas as rotinas de testes para o projeto
    - Envia email reportando resultados dos testes para o coordenador do projeto