



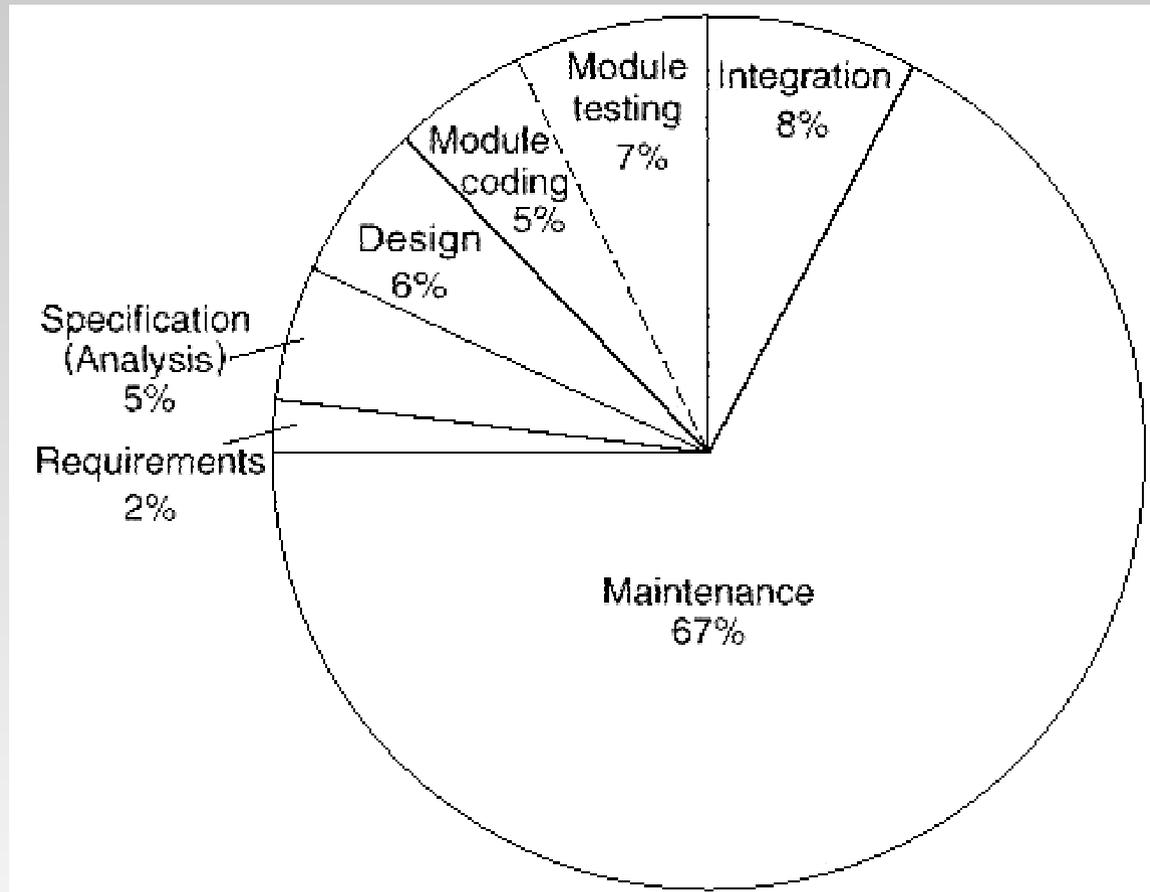
keep the bar green to keep the code clean...



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

- 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

- 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”

- 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 Gamma)
- 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-controller)
 - 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

– 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 descobriam, 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

The screenshot displays the Eclipse IDE interface during a JUnit test run. The top window shows the overall test status: "JUnit (ErpTestsAll)" with a progress bar of 10 red blocks, indicating 100% failure. Below this, the "JUnit (ErpTestsAll)" window shows a summary: "Runs: 53/70", "Errors: 0", and "Failures: 8". The "Failures" tab is active, showing a tree view of test classes. The "erp.tests.InventoryDocumentReportTest" class is selected, and the "testConfirmSale" method is highlighted. The "Failure Trace" window shows the following error details:

```
 junit.framework.AssertionFailedError: Qual...
  at erp.tests.InventoryDocumentReportTe...
  at erp.tests.InventoryDocumentReportTest.testPchDocConfirm(InventoryDocumentReportTest.java:575)
  at erp.tests.InventoryDocumentReportTe...
  at erp.tests.InventoryDocumentReportTe...
  at erp.tests.InventoryDocumentReportTe...
  at sun.reflect.NativeMethodAccessorImpl...
  at sun.reflect.NativeMethodAccessorImpl...
```

The "Package Explorer" on the left shows the project structure, with "erp.tests.InventoryDocumentReportTest" selected. The "Console" window at the bottom right shows the following SQL query:

```
left outer join Tb_Er_Usr_User Usr on (Cln_Usr_Key_Salesrep =
left outer join Tb_Er_Cur_Currency Cur on (Cln_Cur_Key = Cur.
left outer join Tb_Er_Usr_User App on (Cln_Usr_Key_CreditLim
where (Cln_Per_Key = ?)
|TRASH|URL: cmd=erp.FreeCmd jsp=main action=execute dir=
```

The "Code Editor" window shows the source code for the "testConfirmSale" method:

```
1.314 /**
1.315  * Confirm the purchase
1.316  * @param pchKey The key of the purchase to be confirmed
1.317  * @throws Exception
1.318  */
1.319 public void testConfirmSale()
1.320     throws Exception
1.321 {
1.322     testInsertSaleItems();
1.323
1.324     VO clientVO = getSaleClient( salKey );
1.325
1.326     Integer totalOpenPurchases = clientVO.getAsInteger("Cln_Num_
1.327
1.328     Integer totalPurchases = clientVO.getAsInteger("Cln_Num_Buy
1.329
1.330     BigDecimal totalValueOpenPurchases = clientVO.getAsBigDecim
1.331
1.332
1.333
1.334     // Test purchase order confirmation
1.335     cmdRunner.resetConversationalObjects();
1.336
1.337     cmdRequest = cmdRunner.getCmdRequest();
```

JUnit-SE

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

- Adoção formal da metodologia:
 - Documentação (Convenções de Testes)
 - Processo a mais para ser verificado pelo Guardiã
 - 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