1 - Estratégias de Testes de Software

Ao longo do processo de desenvolvimento o software precisa ser testado em diferentes fases. A Figura 1 mostra a sequência de realização dos testes e a seguir tem-se o que é testado em cada fase de teste:

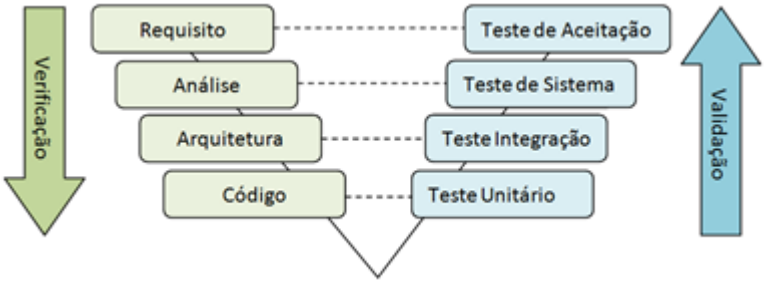

Figura 1 – Modelo V (fases do desenvolvimento x fases dos testes).
Fonte: http://www.devmedia.com.br/teste-de-integracao-na-pratica/31877

- Teste de unidade:
  - Tem por objetivo testar individualmente cada unidade programada, como exemplo, uma classe;
  - Os testes de unidade não testam somente as entradas e saídas da unidade, mas também podem testar caminhos específicos na estrutura de controle do código (teste de caixa branca);
  - Esses testes são realizados pelo próprio programador.
- Teste de integração:
  - Enquanto o teste de unidade se preocupa somente com o funcionamento de uma unidade por vez, o teste de integração se preocupa com o funcionamento integrado das unidades;
  - Unidades funcionam bem de modo isolado, porém quando colocadas juntas, situações inesperadas podem ocorrer. Os testes de integração buscam descobrir erros de interface entre os módulos/classes e de dependências entre os componentes da aplicação;
  - Faz uso de técnicas de projeto de casos de teste que enfocam as entradas e saídas, além de exercitar caminhos específicos (usando diagramas e grafos);
  - Esses testes podem ser realizados pelo próprio desenvolvedor e pela equipe independente de teste.
- Teste de sistema:
  - Testa a combinação do software com outros elementos do sistema, tais como, hardwares, bancos de dados e aplicativos de terceiros;
  - O objetivo é exercitar o sistema por completo;
  - Verifica se a função/desempenho global do sistema é alcançado, ou seja, não se limita a checar somente os requisitos funcionais, mas também os requisitos não funcionais, tais como, uso dos recursos computacionais, tempo de resposta, capacidade de recuperação, estresse, segurança e instalação;
  - Esses testes são realizados pela equipe independente de teste.
- Teste de aceitação:
  - Verifica se o software cumpriu os requisitos especificados;
  - Esses testes são realizados pela equipe independente de teste e podem incluir usuários finais para conferir se os requisitos foram atingidos satisfatoriamente.

2 - Teste de Integração

É a fase do teste de software em que unidades são combinadas e testadas em grupo. Essa fase sucede o teste de unidade, em que as unidades são testadas individualmente, e antecede o teste de sistema, em que o sistema completo (integrado) é testado num ambiente que simula o ambiente de operação.

O teste de integração é alimentado pelos unidades previamente testadas individualmente pelo teste de unidade, agrupando-os assim em módulos, como estipulado no plano de teste, e resulta num sistema integrado e preparado para o teste de sistema.

Um bom método de teste de integração deve possuir algumas características importantes que o diferencia dos outros, tais como:

- Identificar casos de teste cedo, o que pode ser feito através de modelos de análise e design, ajudando a entender e expressar melhor os requisitos do sistema;
- Identificar falhas o quanto antes, ainda no processo de desenvolvimento, economizando tempo, custo e esforço;
- Possuir um potencial para automação, podendo ser através de especificações formais, podendo ter mais perspectivas de ser aproveitado na prática;
- Possuir alguma forma de representação, que sirva para entender melhor os relacionamentos existentes entre os componentes da aplicação, e gerenciá-los facilmente;
- Definir, de forma efetiva, a ordem de integração e realização de testes (Fonte: http://docs.computacao.ufcg.edu.br/posgraduacao/dissertacoes/2004/Dissertacao_CidinhaCostaGouveia.pdf).

3 - Abordagens do Teste de Integração

A integração pode seguir uma abordagem incremental ou não. Enquanto na abordagem não-incremental o sistema é agrupado por completo, na abordagem incremental o sistema é agrupado em etapas, facilitando assim o isolamento do erro, pois são testados pequenos incrementos no código, em que erros são mais fáceis de serem isolados e corrigidos.

Existem duas estratégias clássicas de abordagem incremental, top-down e bottom-up. Como exemplo, na abordagem top-down os módulos de alto nível são testados e integrados primeiro, permitindo encontrar primeiro os erros de lógica e fluxo de dados de alto nível. Por outro lado, a abordagem bottom-up requer o teste e integração dos módulos de baixo nível primeiro. A seguir tem-se a descrição das abordagens de integração:

i. **Top-down**: nessa forma os módulos são testados movendo-se descendentemente na hierarquia de integração, começando com um módulo mais amplo, que é o módulo M1 do exemplo da Figura 2. Nessa hierarquia a integração pode ser realizada de duas formas:

   (a) Por profundidade: utiliza-se uma sequência vertical, por exemplo, M1, M2, M5 e M8, onde o primeiro módulo a ser testado é o M1, na sequência o módulo M2, posteriormente o M5 e por último o M8;

   (b) Por largura: utiliza-se uma sequência horizontal, no exemplo da Figura 2 seriam testados os módulos M2, M3 e M4, pois a união deles formam o módulo M1.

   O processo de integração é feito em cinco passos:

   1) O módulo de controle principal (M1 no exemplo da Figura 2) é utilizado como um pseudocontrolador (stub - simulação) do teste, e pseudocontroladores são substituídos por todos os componentes diretamente ligados ao programa principal. Observações:

- Como alguns componentes podem não estar codificados, então criam-se stubs para representar os componentes;
- Um stub tem como objetivo simular o comportamento real de um componente que ainda não está integrado no sistema.

2) Dependendo da abordagem de integração selecionada (profundidade ou largura), os stubs são substituídos, um por vez, pelos seus componentes reais;
3) Testes são conduzidos à medida que cada componente é integrado;
4) Ao término de cada conjunto de testes, outro stub é substituído por seu componente real;
5) Teste de regressão pode ser realizado para garantir que novos erros não tenham sido introduzidos.

ii. **Bottom-up**: nessa forma os módulos são integrados de baixo para cima.

Stubs não são utilizados, uma vez que o processamento necessário para os componentes subordinados em um determinado nível está sempre disponível.

O processo de integração é feito em quatro passos:

1) Módulos de níveis mais baixos são combinados em grupos que executam funções específicas do módulo principal;
2) É elaborado um driver que coordena a entrada e saída dos casos de teste para cada grupo;
3) O grupo é testado;
4) Drivers são substituídos pelos grupos que passam a interagir com os módulos superiores, na estrutura do programa.

Vantagens e desvantagens das integrações top-down e bottom-up:

- **Top-down**:
  - Vantagem: testar logo no início as funções principais do software;
  - Desvantagem: necessidade de criar stubs.
- **Bottom-up**:
  - Vantagem: não precisa de stubs;
  - Desvantagem: módulo principal não existe enquanto todos os módulos não estiverem testados.

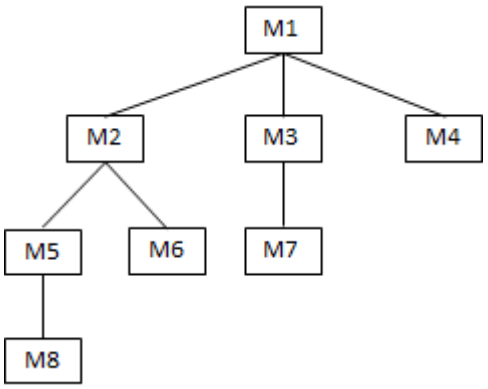

Figura 2 – Representação da estrutura dos módulos de um programa.

Para identificar os erros de interface e de dependências é interessante utilizar alguma representação gráfica que mapeia os relacionamentos entre os métodos/componentes da aplicação. A partir dessa representação gráfica das dependências dos componentes e com o auxílio de outras informações possivelmente fornecidas pelos componentes, são gerados os casos de teste de integração baseados na construção de stubs.

Os diagramas UML e grafos são exemplos de representações gráficas que podem ser utilizadas no processo de mapeamento das interações e dependências entre os componentes.