INSTITUTO METRÓPOLE DIGITAL

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

# Introdução ao Teste de Software.

## Teste de Integração

**Prof. Eiji Adachi M. Barbosa**

# Objetivos

- Apresentar conceitos e abordagens para testes de integração

# Teste de Unidade – Breve revisão

- O que é Unidade?

- O que é Teste de Unidade?

# Testes de Unidade

- Visa revelar falhas em:
  - Interfaces: parâmetros de entrada e saída
  - Estruturas de dados: integridade dos dados armazenados
  - Condições de limite
  - Condições excepcionais

A qualidade de um sistema depende da qualidade de suas unidades.

A qualidade de um sistema depende da qualidade de suas unidades.

Mas a qualidade das unidades, garante a qualidade do sistema?

# Integração

- Integração refere-se a composição de grupos de unidades para formar módulos ou sub-sistemas

# Teste de Integração

- Fase de Teste em que unidades são combinadas e testadas como um grupo
- Assume que unidades foram testadas isoladamente
- Objetiva identificar problemas de interação e compatibilidade entre unidades

# Teste de Integração

- Teste de Integração de Componentes
  - Testa a integração de componentes internos de um sistema

- Teste de Integração de Sistemas
  - Testa a integração entre diferentes sistemas, ou entre hardware e software

# Falhas de Integração

- Falhas de **interpretação**: ocorrem quando a funcionalidade implementada por uma unidade difere do que é esperado
  - B implementa incorretamente um serviço requerido por A
  - B não implementa um serviço requerido por A
  - B implementa um serviço não requerido por A e que interfere com seu funcionamento

# Falhas de Integração

- Falhas devido a **chamadas incorretas**:
  - B é chamado por A quando não deveria (chamada extra).
  - B é chamado em momento da execução indevido (chamada incorreta).
  - B não é chamado por A quando deveria (chamada ausente).

# Falhas de Integração

- Falhas de **interação**: ocorrem quando o padrão de interação (protocolo) entre duas unidades é violado
  - violação da integridade de arquivos e estruturas de dados globais
  - tratamento de erros (exceções) incorreto
  - problema de configuração / versões
  - falta de recursos para atender a demanda das unidades

# Mais falhas de integração

- Problemas não funcionais: ocorre quando requisitos não funcionais são violados
  - Ex.: Módulo B não tem o tempo de resposta esperado por A
  - Ex.: Módulo B não mantém privacidade dos dados recebidos por A

# Falha Famosa

- Equipe em terra enviou dados no sistema imperial de unidades, enquanto os módulos da sonda trabalhava com dados no sistema métrico


Fonte: https://mars.jpl.nasa.gov/msp98/news/mco990930.html

# Abordagens de Integração

- Não Incremental (Big-Bang)
- Estrutural-Incremental

# Abordagem *Big Bang*

- Integração ocorre apenas quando todos os componentes estão prontos

- Todos os componentes são integrados de uma vez e o comportamento emergente dos componentes integrados é verificado
  - Abordagem “Roda e vê no que dá”

# Abordagem *Big Bang*

- Desvantagens:
  - Defeitos nas interfaces entre componentes são identificados tardiamente
  - Difícil de identificar e isolar causas das falhas observadas

# Abordagem *Big Bang*

- Programas não devem ser integrados assim

- Talvez dê certo para programas pequenos (e só para os pequenos)

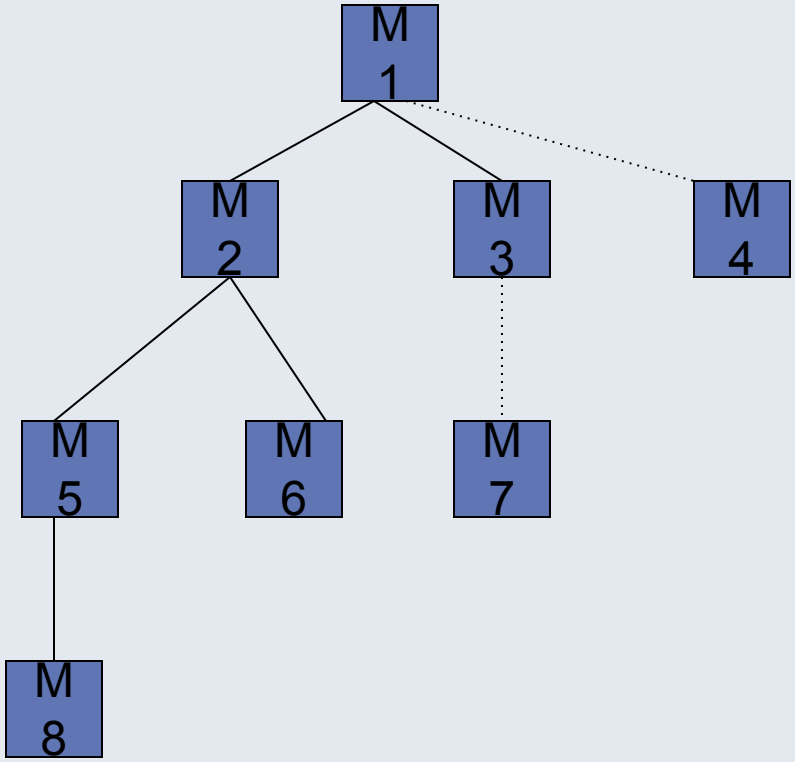
# Como integrar?

- Se a abordagem Big Bang é ruim, então como devemos integrar um programa?

## Abordagens Incrementais

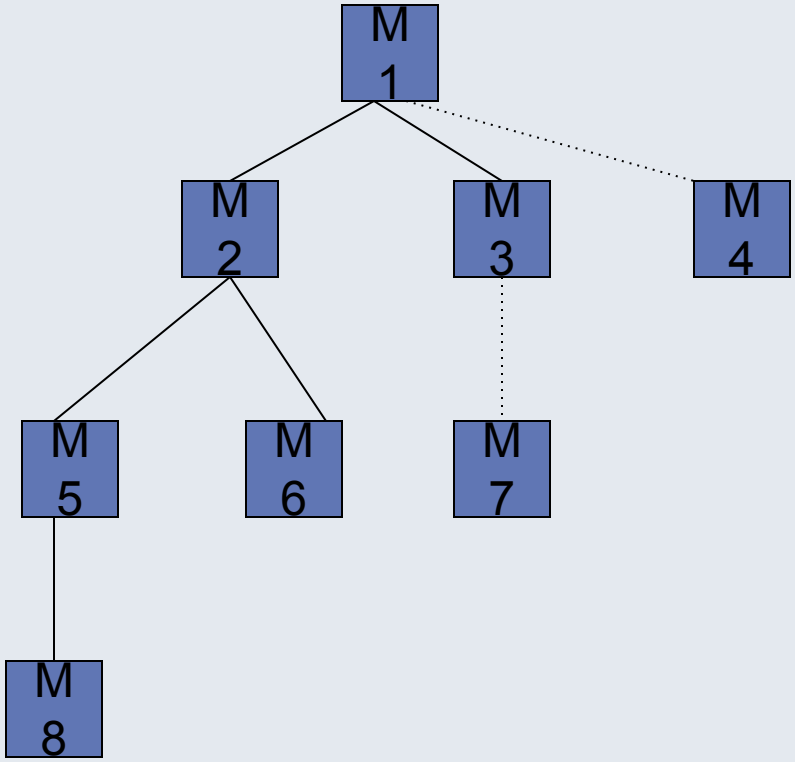
# Terminologia

- Driver
  - Dado um módulo sob teste chamado M, um *test driver* é um módulo “falso” que passa dados de teste para exercitar M



# Terminologia

- Stub
  - Dado um módulo sob teste M, um *test stub* é um módulo “falso” que simula o comportamento de uma dependência de M
    - Não confundir com o *stub* que é um tipo de Test Double

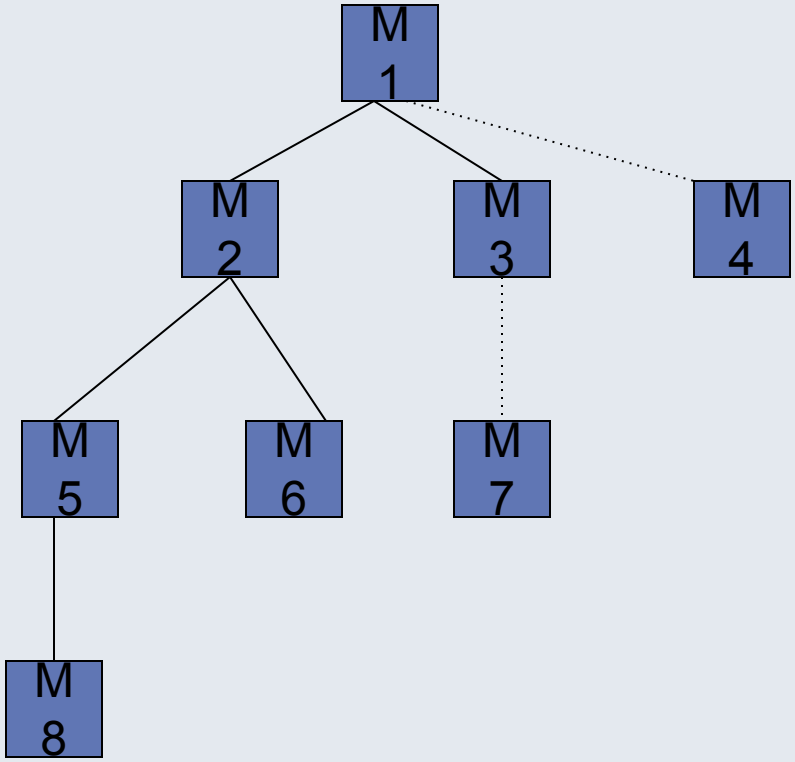


# Teste de Integração

- Abordagens incrementais:
  - *Top Down*
  - *Bottom Up*

# Abordagem *Top-Down*

- Módulos são integrados incrementalmente seguindo uma ordem “de cima para baixo” a partir do módulo *Main (M1)*

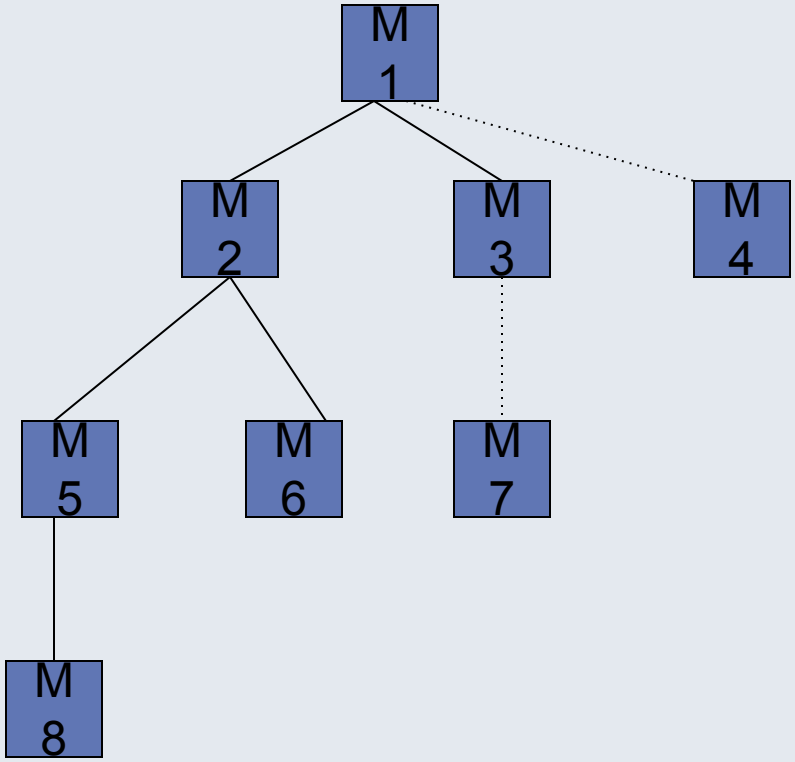


# Abordagem *Top-Down*

- Passos:
  - O próprio módulo *Main* coordena a execução do programa (é o *test driver*)
  - Módulos *Stub* são criados para todos os módulos subordinados ao módulo *Main*
  - Módulos *Stub* são substituídos pelos módulos verdadeiros
    - Estratégias de ordem substituição:
      - Em Largura
      - Em Profundidade
  - Testes são realizados a cada módulo integrado

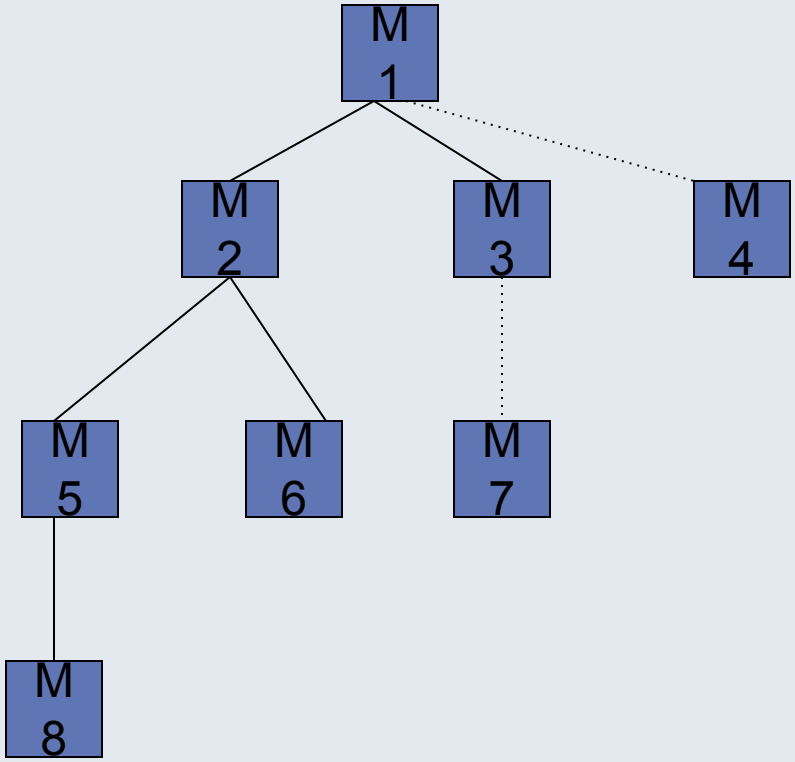
# Abordagem *Top-Down*

- Estratégia em Largura
  - M1 testado com *stubs* para M2, M3 e M4
  - M1-M2 testado com *stubs* para M3, M4, M5, e M6
  - M1-M2-M3 testado com *stubs* para M4, M5, M6, e M7
  - …



# Abordagem *Top-Down*

- Estratégia em Profundidade
  - M1 testado com *stubs* para M2, M3 e M4
  - M1-M2 testado com *stubs* para M3, M4, M5, e M6
  - M1-M2-M5 testado com *stubs* para M3, M4, M6, M7 e M8
  - …



# Abordagem *Top-Down*

- Não requer a implementação de *test drivers*
- Requer a implementação de múltiplos *stubs*

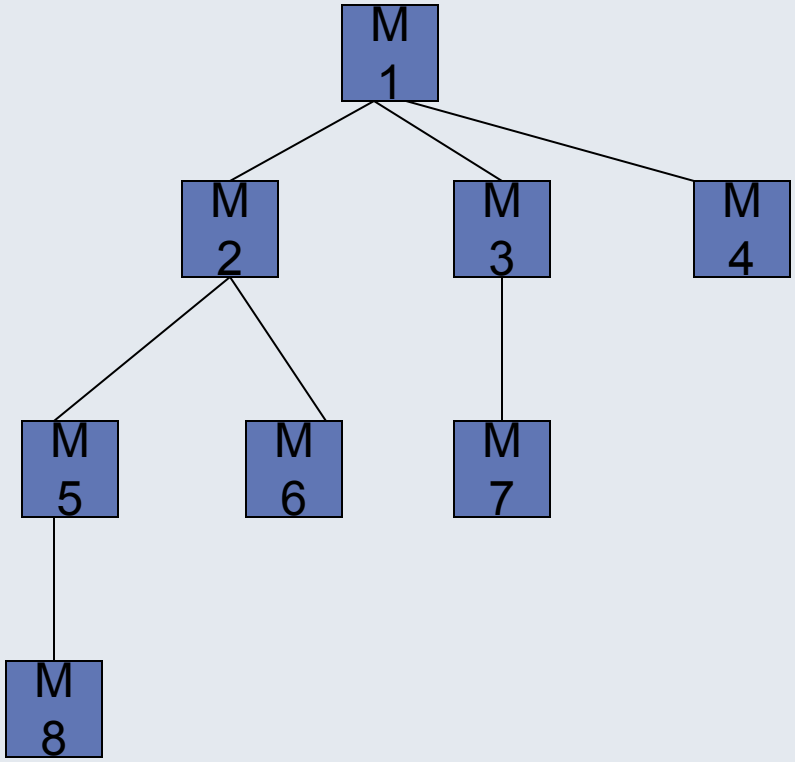
# Abordagem *Bottom-Up*

- Passos:
  - Módulos inferiores são combinados em grupos
  - Um *Driver* é produzido para coordenar as entradas e saídas dos módulos inferiores
  - O grupo de módulos é testado
  - Driver é removido e outro módulo é incorporado ao grupo

# Abordagem *Bottom-Up*

*Bottom Up:*

- M8 testado com Driver
- M5-M8 testado com Driver
- M6 testado com Driver
- M5-M8-M2 testado com Driver
- …



# Abordagem *Bottom-Up*

- Não requer implementação de *Stubs*
- Inicia construção e teste com módulos atômicos (nós folha)
- Grupos de módulos podem ser testados em paralelo

INSTITUTO METRÓPOLE DIGITAL

UFRN UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE

# Introdução ao Teste de Software.

## Teste de Integração

**Prof. Eiji Adachi M. Barbosa**