# Engenharia de Software II

Aula 7

http://www.ic.uff.br/~bianca/engsoft2/

# Ementa

- Processos de desenvolvimento de software
- **Estratégias e técnicas de teste de software** (Caps. 13 e 14 do Pressman)
- Métricas para software
- Gestão de projetos de software: conceitos, métricas, estimativas, cronogramação, gestão de risco, gestão de qualidade e gestão de modificações
- Reengenharia e engenharia reversa

# Uma Estratégia Global para Arquiteturas Convencionais

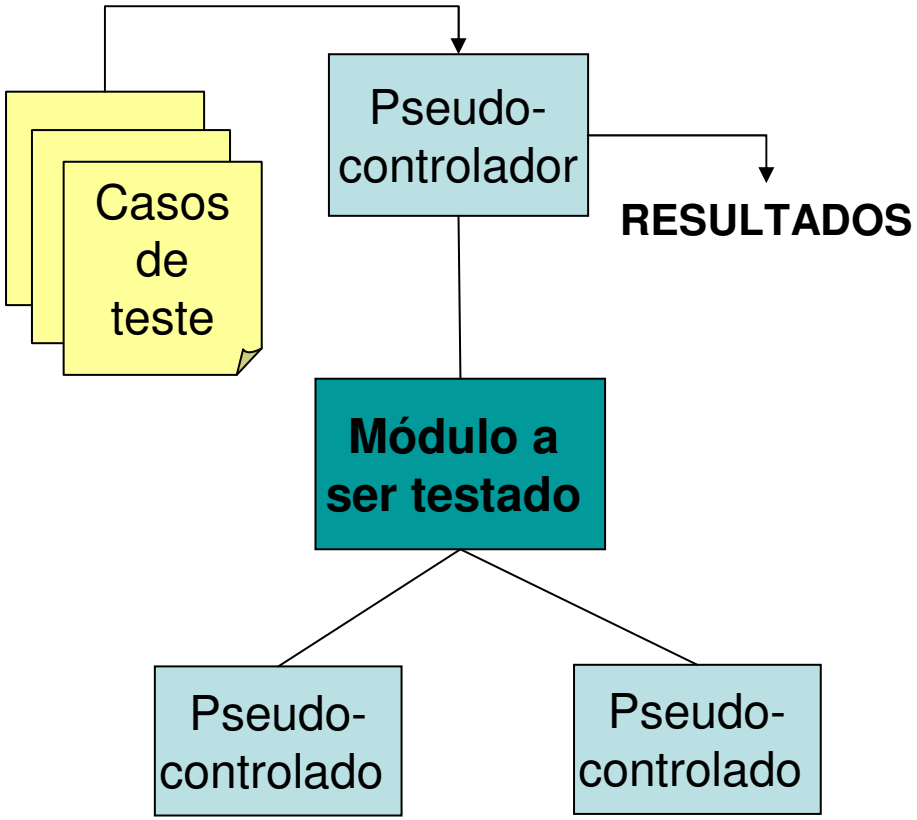
1. Teste de unidade
   - Focaliza cada componente individualmente, garantido que funciona.
   - Faz uso intensivo de técnicas que exercitam caminhos específicos na estrutura de controle.
2. Teste de integração
   - Focaliza o pacote de software completo e trata da verificação do programa como um todo.
   - Faz uso de técnicas de projeto de casos de teste que enfocam as entradas e saídas, além de exercitar caminhos específicos.
3. Teste de validação
   - Critérios de avaliação estabelecidos durante a análise de requisitos são avaliados.
4. Teste de sistema
   - Testa a combinação do software com outros elementos do sistema (como hardware, pessoal e bancos de dados).
   - Verifica se a função/desempenho global do sistema é alcançada.

# Procedimentos de Teste de Unidade

- O projeto de teste pode ser realizado:
  - Antes que o código seja iniciado (abordagem ágil)
  - Depois que o código-fonte tenha sido gerado.
- Uma revisão da informação de projeto fornece diretrizes para o estabelecimento de casos de teste.
- Cada caso de teste deve ser acoplado a um conjunto de resultados esperados.

# Ambiente de Teste de Unidade

- Um pseudocontrolador (*driver*) deve ser criado para cada unidade.
  - Programa principal que aceita dados do caso de teste, passa os dados para o componente e imprime os resultados.
- O pseudocontrolado (*stub*) substitui um sub-módulos que é chamado pelo módulo sendo testado.
  - Faz o mínimo de manipulação de dados necessário.



# Procedimentos de Teste de Unidade

- Pseudocontroladores e pseudocontrolados são despesas indiretas.
  - Precisam ser escritos mas não são entregues.
  - Há situações em que não há recursos para fazer testes de unidades abrangentes.
    - Nesse caso, deve-se selecionar alguns módulos críticos e mais complexos.

# Teste de Integração

- Mesmo que todos os módulos estejam funcionando individualmente, não se pode garantir que eles funcionarão em conjunto.
  - Dados podem ser perdidos na interface
  - Imprecisão aceitável individualmente pode ser amplificada
  - Estruturas de dados globais podem apresentar problemas
- Teste de integração é uma técnica sistemática para construir a arquitetura do software enquanto se conduz testes para descobrir erros.

# Abordagens de Integração

- Abordagem *big-bang*
  - Todos os componentes são combinados com antecedência.
  - O programa inteiro é testado d uma vez.
  - Usualmente resulta em caos!
    - A correção é difícil, porque fica complicado isolar as causas dos erros.
    - Uma vez corrigidos os erros, novos erros aparecem.

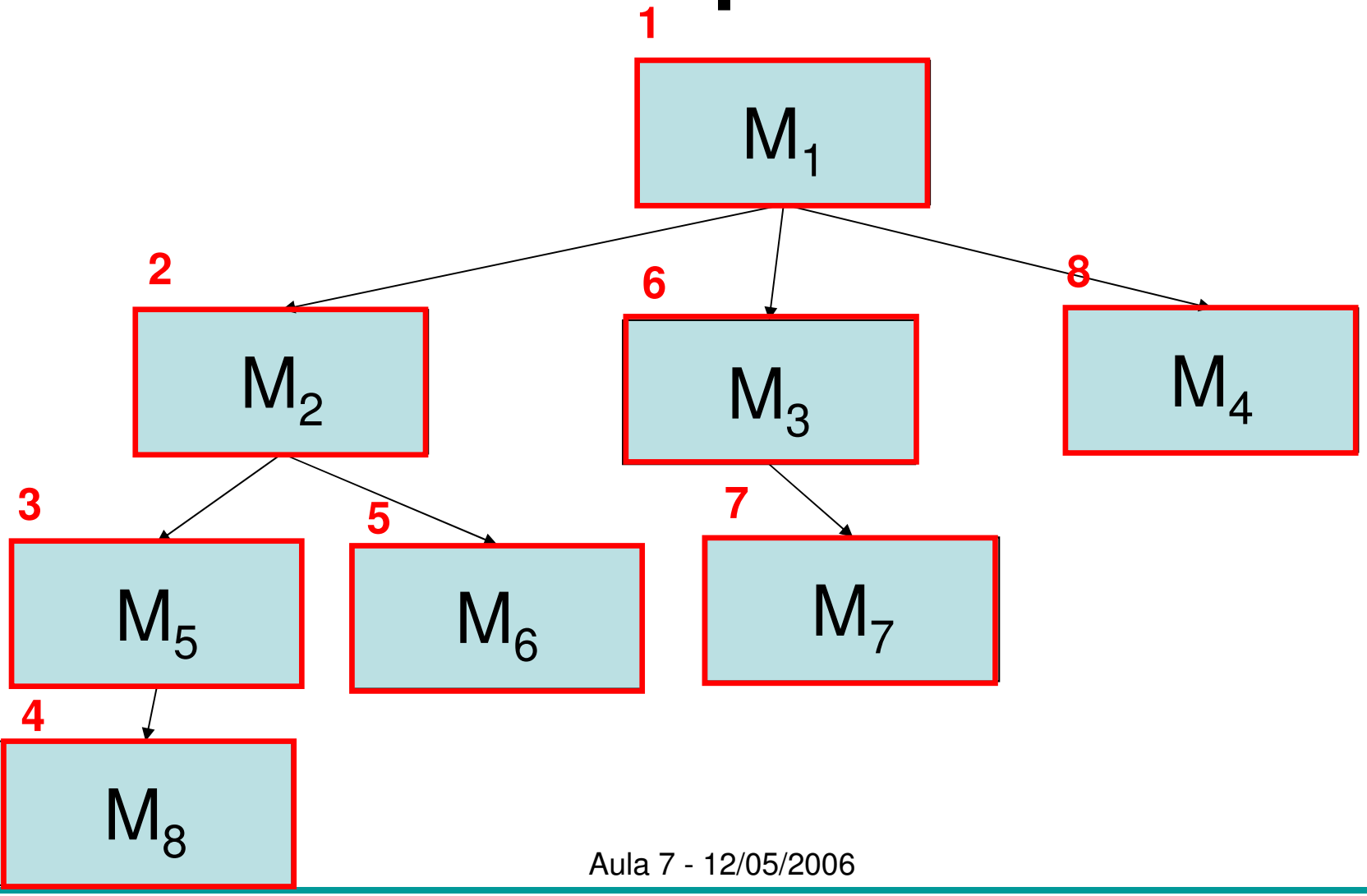
# Abordagens de Integração

- Integração incremental
  - É a antítese da abordagem big-bang.
  - O programa é construído e testado em pequenos incrementos.
  - Erros são mais fáceis de isolar e corrigir.
  - Pode ser aplicada uma interface sistemática de testes.
  - Há várias estratégias incrementais de integração.
    - Integração descendente
    - Integração ascendente
    - Teste de regressão
    - Teste fumaça

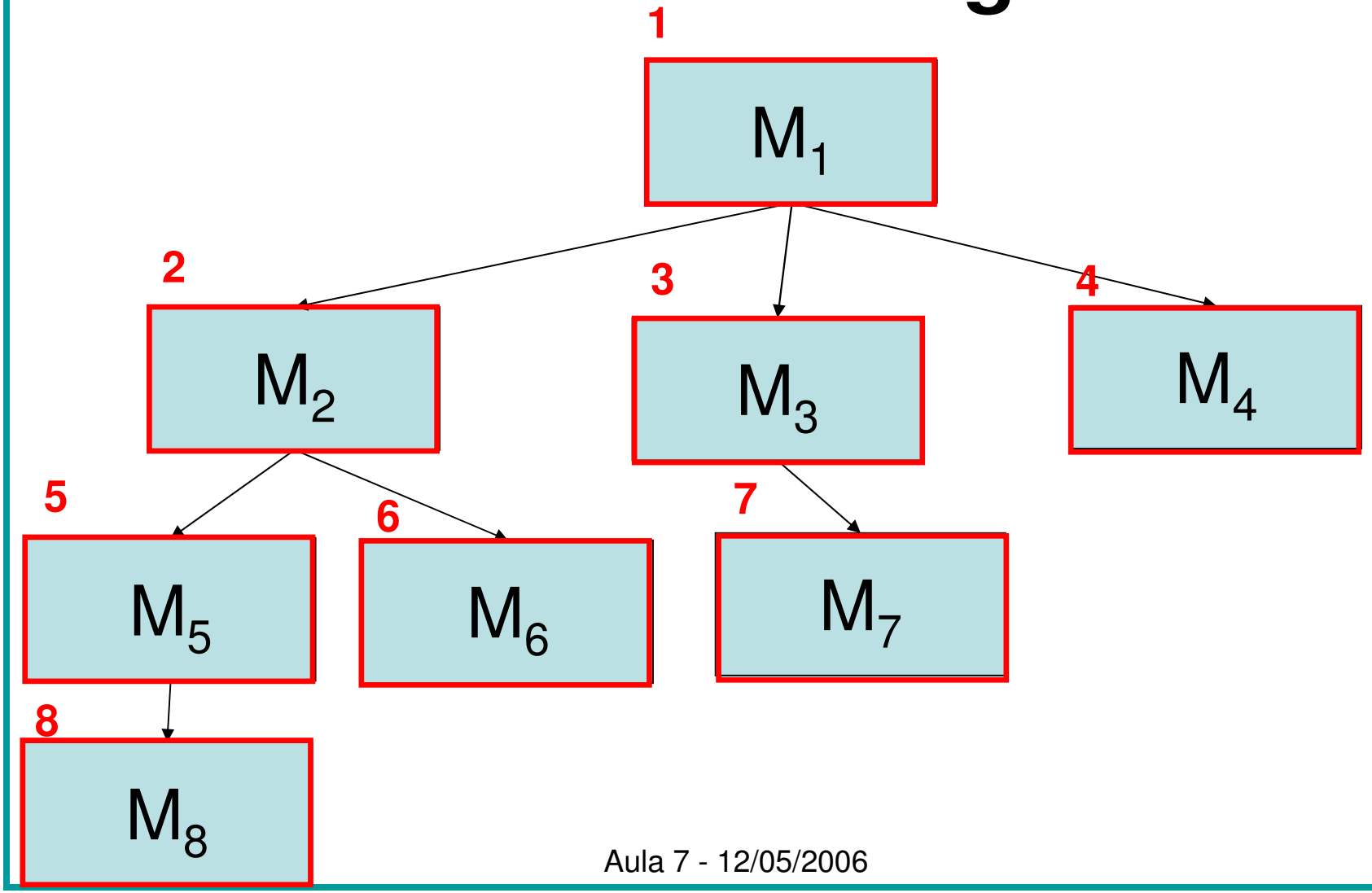
# Integração Descendente

- Na integração descendente (*top-down*) os módulos são integrados movendo-se de cima para baixo na hierarquia de controle.
  - Começa pelo módulo de controle principal.
  - Os módulos subordinados são incorporados à estrutura de uma de duas maneiras:
    - Primeiro-em-profundidade (*depth-first*)
    - Primeiro-em-largura (*breadth-first*)

# Integração Ascendente Primeiro-em-profundidade



# Integração Ascendente
# Primeiro-em-largura



# Passos de Integração Descendente

1. O módulo de controle principal é usado como pseudocontrolador do teste.
   - Pseudocontrolados substituem todos os componentes diretamente subordinados ao principal.
2. Os pseudocontrolados são substituídos, um de cada vez, pelos componentes reais.
3. Testes são conduzidos à medida que cada componente é integrado.
4. Ao término de cada conjunto de testes, outro pseudocontrolado é substituído pelo componente real.
5. O teste de regressão pode ser conduzido para garantir que novos erros não tenham sido introduzidos.

# Desvantagem da Integração Descendente

- O processamento nos níveis baixos da hierarquia pode ser necessário para testar adequadamente os níveis superiores.
  - Como são usados pseudocontrolados, nenhum dado significativo flui para cima na estrutura do programa.
- Três soluções são possíveis:
  - Adiar alguns testes
  - Desenvolver pseudocontrolados mais complexos
  - Integrar o software de baixo para cima: integração ascendente.

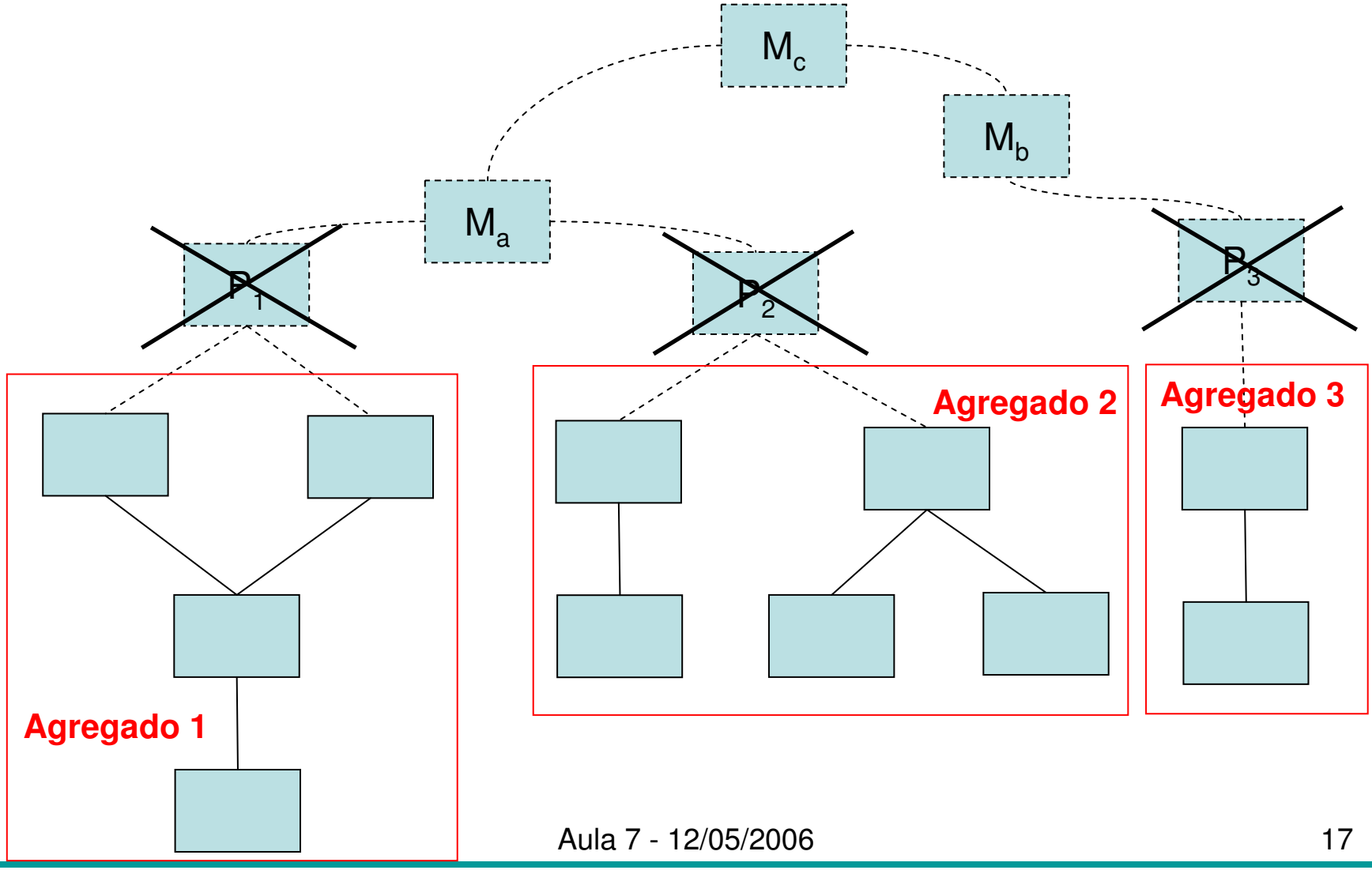
# Integração Ascendente

- Na integração descendente (*bottom-up*) os módulos são integrados movendo-se de baixo para cima na hierarquia de controle.
  - Inicia a construção e teste pelos módulos atômicos.
  - Elimina a necessidade de pseudocontrolados complexos.

# Passos da Integração Ascendente

1. Componentes de baixo nível são combinados em agregados (*clusters*), que realizam uma subfunção específica.
2. Um pseudocontrolador é escrito para coordenar a entrada e a saída do caso de teste.
3. O agregado é testado.
4. Pseudocontroladores são removidos e agregados são combinados movendo-se para cima na estrutura.

# Integração Ascendente



# Integração Ascendente vs. Descendente

- Integração Descendente
  - Desvantagem: necessidade de pseudocontrolados.
  - Vantagem: testar logo as principais funções de controle.
- Integração Ascendente
  - Desvantagem: o programa como um todo não é testado até o final.
  - Vantagem: ausência de pseudocontrolados e projeto de casos de teste facilitados.
- A seleção de uma das abordagens depende das características do software e do projeto.
- Uma abordagem combinada pode ser o melhor compromisso.
  - Testes descendentes para níveis mais altos.
  - Testes ascendentes para os níveis subordinados.

# Teste de Regressão

- Toda vez que um novo módulo é adicionado como parte do teste de integração, o software se modifica.
  - Novos caminhos de fluxos de dados são estabelecidos.
  - Nova E/S pode ocorrer.
  - Nova lógica de controle é adicionada
- Essas modificações podem ser causadas problemas com funções que previamente funcionavam corretamente.
- O teste de regressão é a re-execução de algum sub-conjunto de testes que já foi conduzido para garantir que as modificações não introduzam efeitos colaterais indesejáveis.

# Teste de Regressão

- Pode ser conduzido manualmente ou usando ferramentas automatizadas de captação/reexecução.
  - Permitem ao engenheiro de software captar casos de teste e resultados para reexecução e comparação.
- À medida que o teste de integração prossegue, o número de testes de regressão pode crescer significativamente.
  - A suíte de testes de integração deve ser projetada para incluir apenas testes que cuidam das principais funções do programa.

# Módulos Críticos

- Um módulo crítico tem uma ou mais das seguintes características:
  - Aborda vários requisitos do software.
  - Está num alto nível da estrutura de controle.
  - É complexo ou propenso a erro.
  - Tem requisitos de desempenho bem definidos.
- Módulos críticos devem ser testados tão cedo quanto possível.
- Testes de regressão devem ser focados na função de módulos críticos.

# Teste Fumaça

- É uma estratégia de integração constante.
- O software é reconstruído e testado diariamente para dar aos gerentes e desenvolvedores uma avaliação realística do progresso.

# Atividades do Teste Fumaça

1. Componentes de software são integrados em uma "construção".
   - Inclui todos os dados, bibliotecas, módulos reusáveis e componentes que são necessários para implementar uma função do produto.
2. Uma série de testes é projetada para expor erros que impeçam a construção de desempenhar adequadamente a sua função.
   - Propósito principal: descobrir erros "bloqueadores" que tem a maior chance de atrasar o cronograma.
3. A construção é integrada com outras construções e o produto inteiro é testado diariamente.
   - Pode ser usada uma abordagem descendente ou ascendente.

# Benefícios do Teste Fumaça

- O risco de integração é minimizado.
  - Incompatibilidades e outros erros de bloqueio são descobertos logo no início, evitando impactos no cronograma.
- A qualidade do produto final é aperfeiçoada.
  - Descobre tanto erros funcionais quanto defeitos de projeto arquitetural.
- Diagnóstico e correção de erros são simplificados.
  - O software que acabou de ser adicionado às construções é uma causa provável do erro recém-descoberto.
- Progresso é fácil de avaliar.
  - A cada dia que passa, mais é integrado ao software e mais se pode demonstrar que ele funciona.

# Documentação do Teste de Integração

- Documento de especificação de teste
  - Plano de teste global para integração do software
  - Descrição dos testes específicos
- É um produto de trabalho e torna-se parte da configuração de software.

# Teste de Integração no Contexto Orientado a Objetos

- Não há uma estrutura óbvia hierárquica.
  - Estratégias de integração ascendente e descendente perdem o significado.
- Há duas estratégias existentes para o contexto OO:
  - Teste baseado no caminho de execução
    - Integra o conjunto de classes necessárias para responder a uma entrada ou evento do sistema.
  - Teste baseado no uso
    - Começa a construção do sistema testando as classes que usam poucas (ou nenhuma) classes servidoras.
- Pseudocontroladores podem ser usados para testar operações no mais baixo nível e testar grupos inteiros de classes, e para substituir a interface com o usuário.
- Pseudocontrolados podem ser usados em situações nas quais a colaboração entre classes é necessária.