Fase de Transição (*Transition*)

OpenUP

Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte
Diretoria de Educação e Tecnologia da Informação
Curso de Tecnologia em Análise e Desenvolvimento de Sistemas

Prof. Fellipe Aleixo (*fellipe.aleixo@ifrn.edu.br*)

Fase de Transição

- Propósitos da fase:
  - Garantir que o software está pronto para ser entregue aos usuários
  - Realizar uma sintonia fina do software (versão "beta" resultante da iteração de construção), em termos de
    - Funcionalidade
    - Usabilidade – clareza e facilidade de acesso
    - Estabilidade
    - Performance
    - e da qualidade geral

Objetivos e Atividades

Objetivos da Fase	Atividades que Endereçam os Objetivos
Realizar um teste “beta” para validar se as expectativas do usuário foram satisfeitas	(i) Tarefas contínuas (ii) Desenvolver um incremento da solução (iii) Testar a solução
Alcançar a concordância dos *stakeholders* que o desenvolvimento está completo	(i) Planejar e gerenciar a iteração (ii) Testar a solução
Melhorar a performance de projetos futuros através das lições aprendidas	(i) Planejar e gerenciar a iteração

Objetivos e Atividades

- Para se atingir a concordância dos *stakeholders* é preciso:
  - Executar vários níveis de teste de aceitação, incluindo testes formais, informais e “beta”
- A validação da versão “beta” envolve:
  - Correção de bugs
  - Fazer melhoramentos na performance e usabilidade
- Para melhorar os projetos futuros:
  - As lições aprendidas precisam estar documentadas
  - Deve-se mapear os impactos no processo e nas ferramentas utilizadas

Considerações Chave

- A fase de transição pode incluir
  - Execução de sistemas antigos em paralelo
  - Migração de dados
  - Treinamento de usuários
  - Ajustes de processos de negócio
- Cada uma das ações listadas acima deve ser precedida de uma estratégia bem definida
  - Bem como a definição de soluções de contorno para possíveis problemas de execução

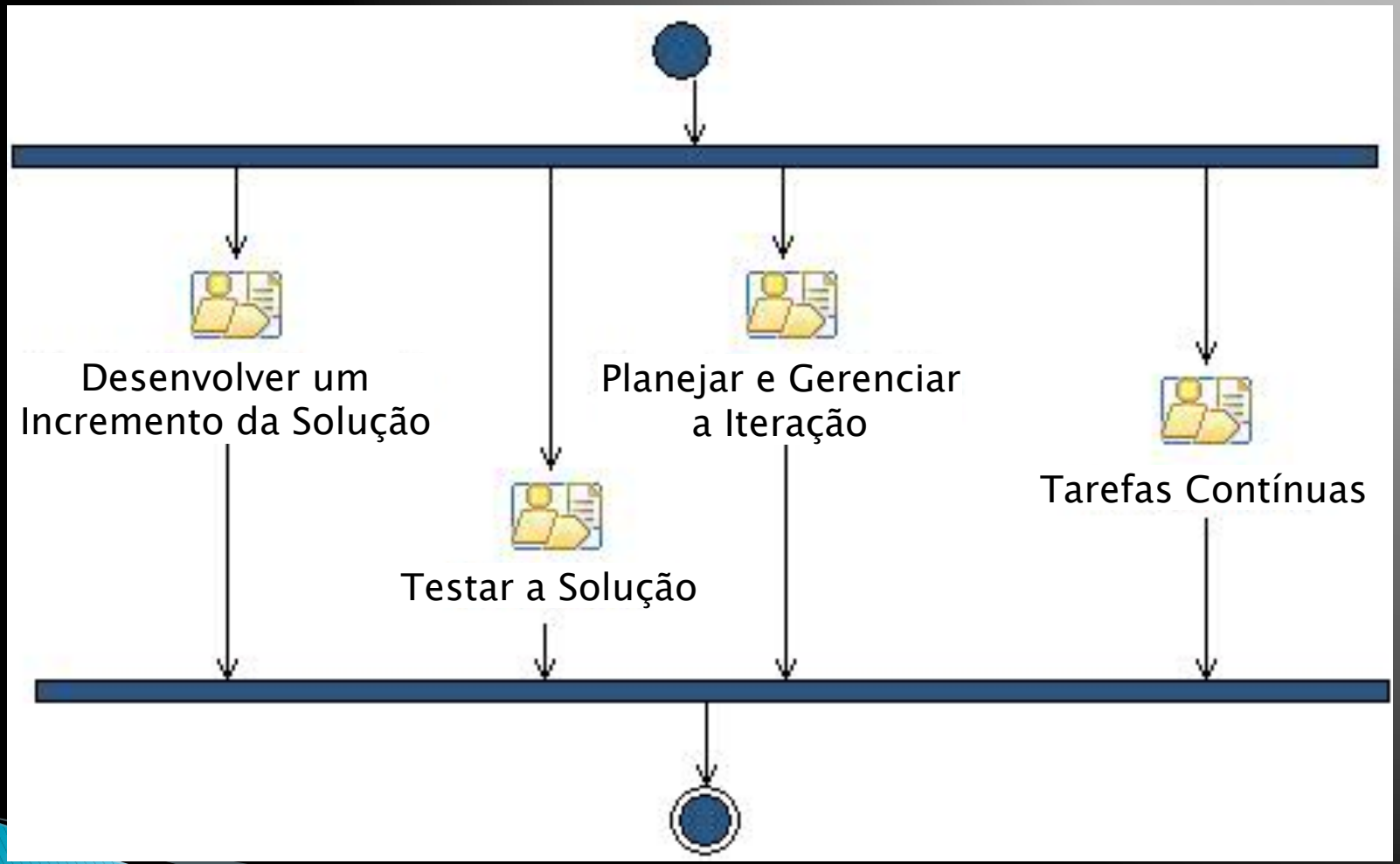
Considerações Chave

- O número de iterações de transição varia de acordo com o cenário em questão:
  - 1 iteração → sistemas simples que podem requerer poucas correções de *bugs*
  - Várias iterações → sistemas mais complexos, envolvendo a adição de novas características, além da realização de um conjunto de ações de migração entre um sistema legado e o novo sistema

Considerações Chave

- Quando os objetivos da fase de transição forem alcançados o projeto pode ser fechado
- Para alguns produtos, o fim do ciclo de vida do desenvolvimento pode coincidir com início de um novo ciclo de vida
  - Manutenção (mesma equipe ou uma nova equipe)
  - Desenvolvimento de uma nova versão

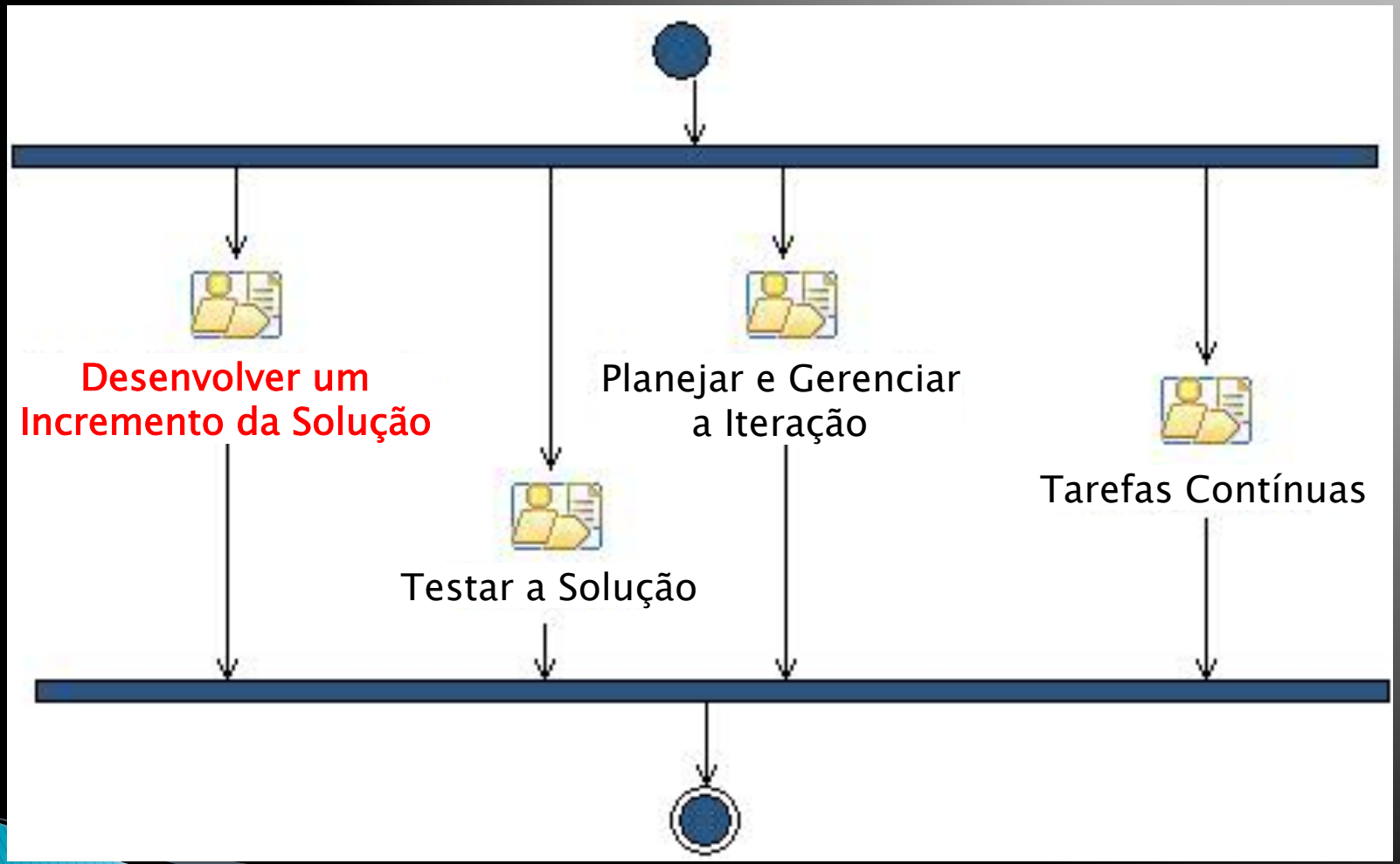
Estrutura de Divisão de Trabalho



Atividades

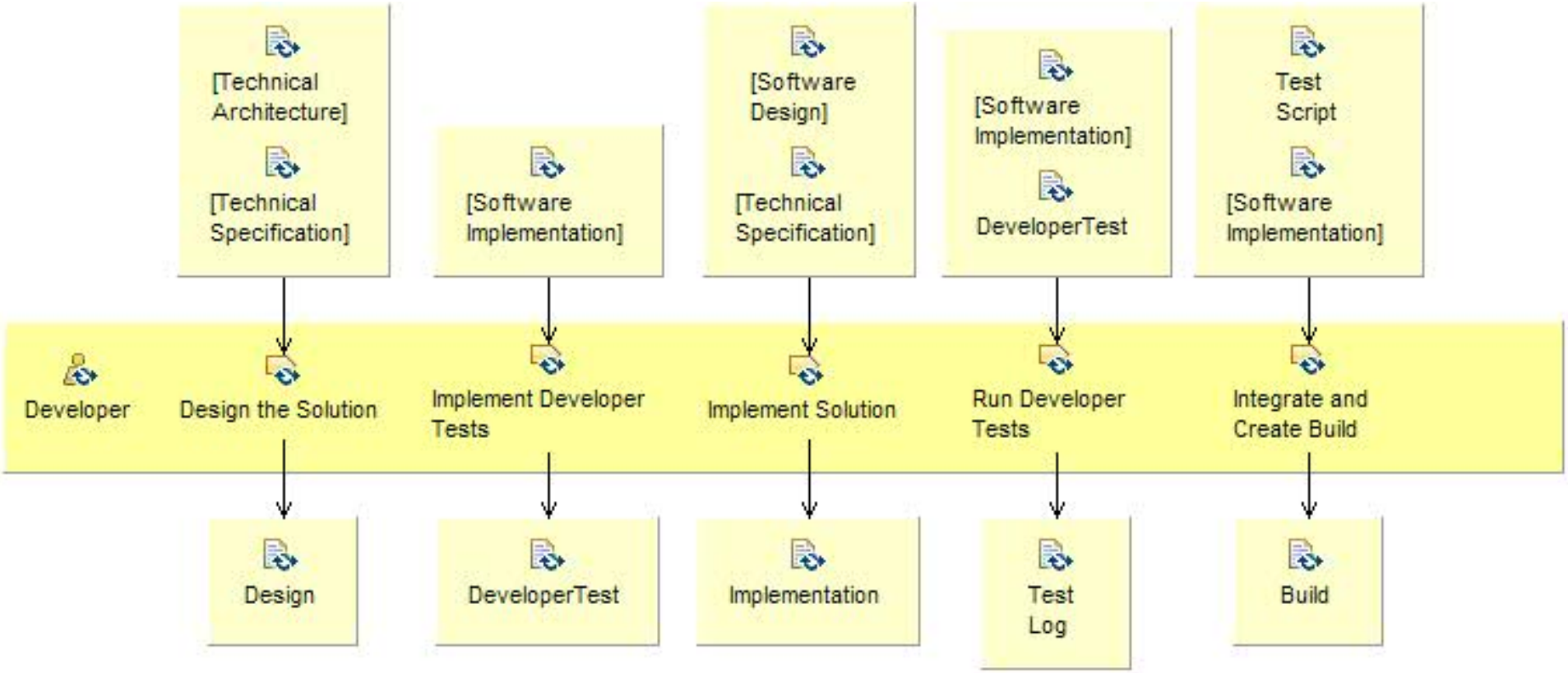
- Descrição das atividades, suas tarefas e os respectivos passos

Estrutura de Divisão de Trabalho



Desenvolver um Incremento da Solução

- Aplicada na (i) correção de *bugs*, (ii) melhoramentos (performance ou usabilidade) ou (iii) adição de novas funcionalidades



Projeto

- Não é necessário fazer o projeto para todos os casos de uso sendo trabalhados
  - Para o caso de novas funcionalidades: pode ser tomado como base a arquitetura e os projetos de outros casos de uso já implementados
  - Nesse estágio, fazer o projeto pode comprometer o tempo a ser usado na correção de outros *bugs*
  - Se o projeto for vital a opção é por um projeto ágil
- Para os casos de correção de *bug* e melhorias é importante se consultar o projeto original

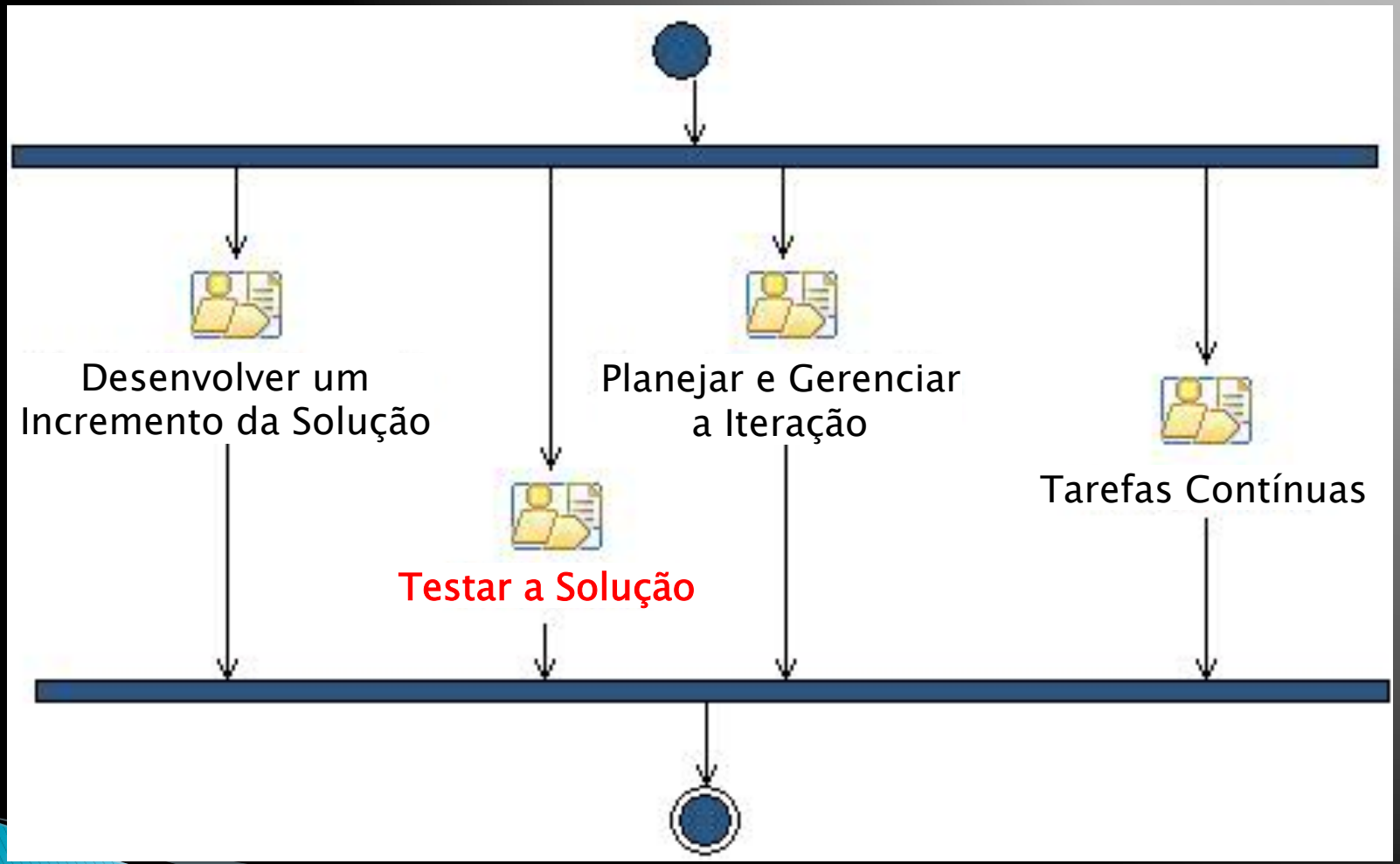
Testes de Desenvolvedor

- Os testes de desenvolvedor ganham uma importância ainda maior nessa fase
  - Devido a possível ausência do projeto
  - Trazem segurança ao desenvolvedor, que o código inserido não compromete outras partes do sistema
- As modificações no código só serão integradas se todos os testes passarem

Refatoração

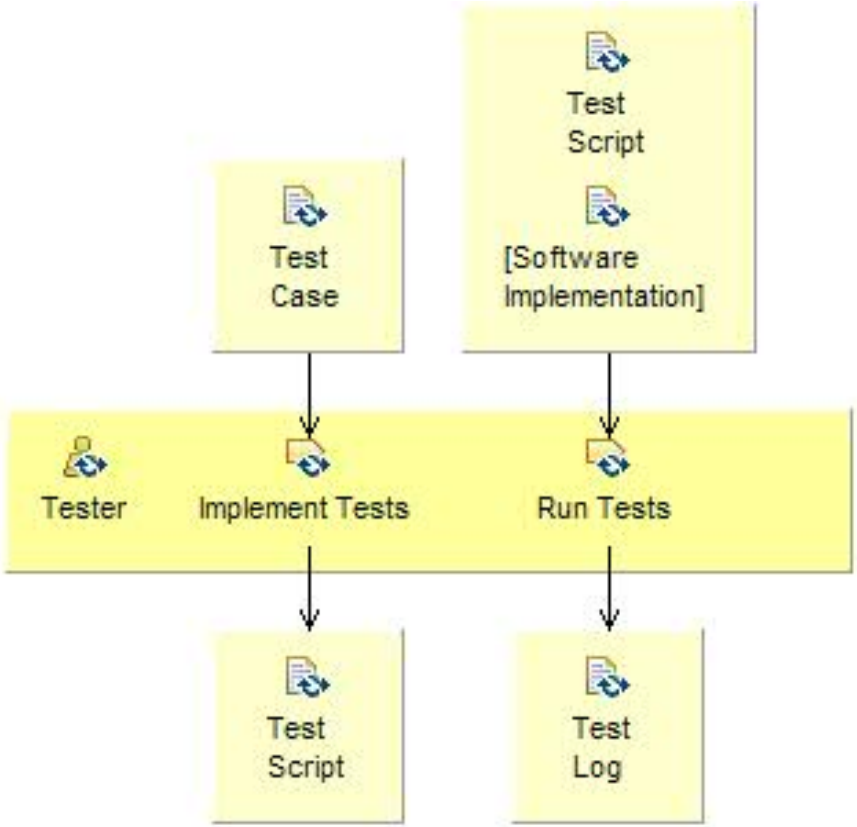
- Durante a implementação é importante se observar oportunidades de refatoração
  - Reestruturar o código sem mudar o seu comportamento no intuito de melhorar o projeto, a performance ou deixar o mesmo mais claro
- Quatro razões para se refatorar o código:
  1. Melhora o projeto do software
  2. Torna o código mais fácil de entender
  3. Ajuda a encontrar *bugs*
  4. Torna o programa mais rápido
- Só é seguro refatorar se existirem os testes de desenvolvedor para validar o resultado

Estrutura de Divisão de Trabalho



Testar a Solução

- A prioridade são os testes de aceitação executados pelos usuários finais
- Implementar os testes seria preparar o cenário para os testes de aceitação
- O usuário executa todos os casos de uso e "aceita" ou não



Tarefa: Implementar os Teste

- Passos:
  1. Selecionar os casos de teste a serem implementados
  2. Projetar o roteiro de testes
  3. Implementar um roteiro de testes executável
  4. Definir dados de teste específicos
  5. Organizar os roteiros de teste em “suítes”
  6. Verificar a implementação dos testes
  7. Compartilhar e avaliar os roteiros de teste

Tarefa: Executar os Testes

- Passos:
  1. Rever os itens de trabalho completados no *build*
  2. Selecionar os roteiros de testes
  3. Executar os roteiros de testes no *build*
  4. Analisar e comunicar os resultados dos testes
  5. Prover um feedback para a equipe, identificando:
     - *Bugs*
     - Possíveis melhorias (funcionalidade, interface ou performance)
     - Possíveis novas funcionalidades (a se discutir)

Teste de Requisitos Qualitativos

- Nesse momento é importante observar outros aspectos a serem testados
  - Integridade (resistência à falhas)
  - Habilidade de instalar e usar em várias plataformas
  - Habilidade de manipular muitas solicitações simultâneas

Teste de Requisitos Qualitativos

- Tipos de teste de funcionalidade:
  - Teste funcional → foca em validar as funções, conforme foram especificadas (casos de uso)
  - Teste de segurança → foca em garantir que os dados são acessíveis apenas atores "autorizados"
  - Teste de volume → foca em verificar a habilidade do sistema em manipular um grande volume de dados (entrada, saída ou no banco de dados)

Teste de Requisitos Qualitativos

- Tipos de teste de usabilidade:
  - Teste de usabilidade → foca em fatores humanos, estética, consistência dos dados, ajuda sensível ao contexto, assistentes e agentes, documentação do usuário, material de treinamento
  - Teste de integridade → foca em avaliar a robustez (resistência à falha), fidelidade a linguagem, sintaxe e uso dos recursos
  - Teste de estrutura → foca em avaliar a aderência ao projeto de "fluxo de navegação" (p.ex.: *links* de páginas web corretor e não quebrados)

Teste de Requisitos Qualitativos

- Tipos de teste de confiabilidade:
  - Teste de *stress* → foca em avaliar como o sistema responde em condições fora do normal (carga de trabalho extrema, memória insuficiente, serviços e hardware indisponível, ou recursos compartilhados limitados)
  - Teste de *benchmark* → foca em comparar a performance do novo sistema com algum sistema de referência
  - Teste de contenção → foca em validar a habilidade de manipular, de forma aceitável, múltiplas demandas para um mesmo recurso

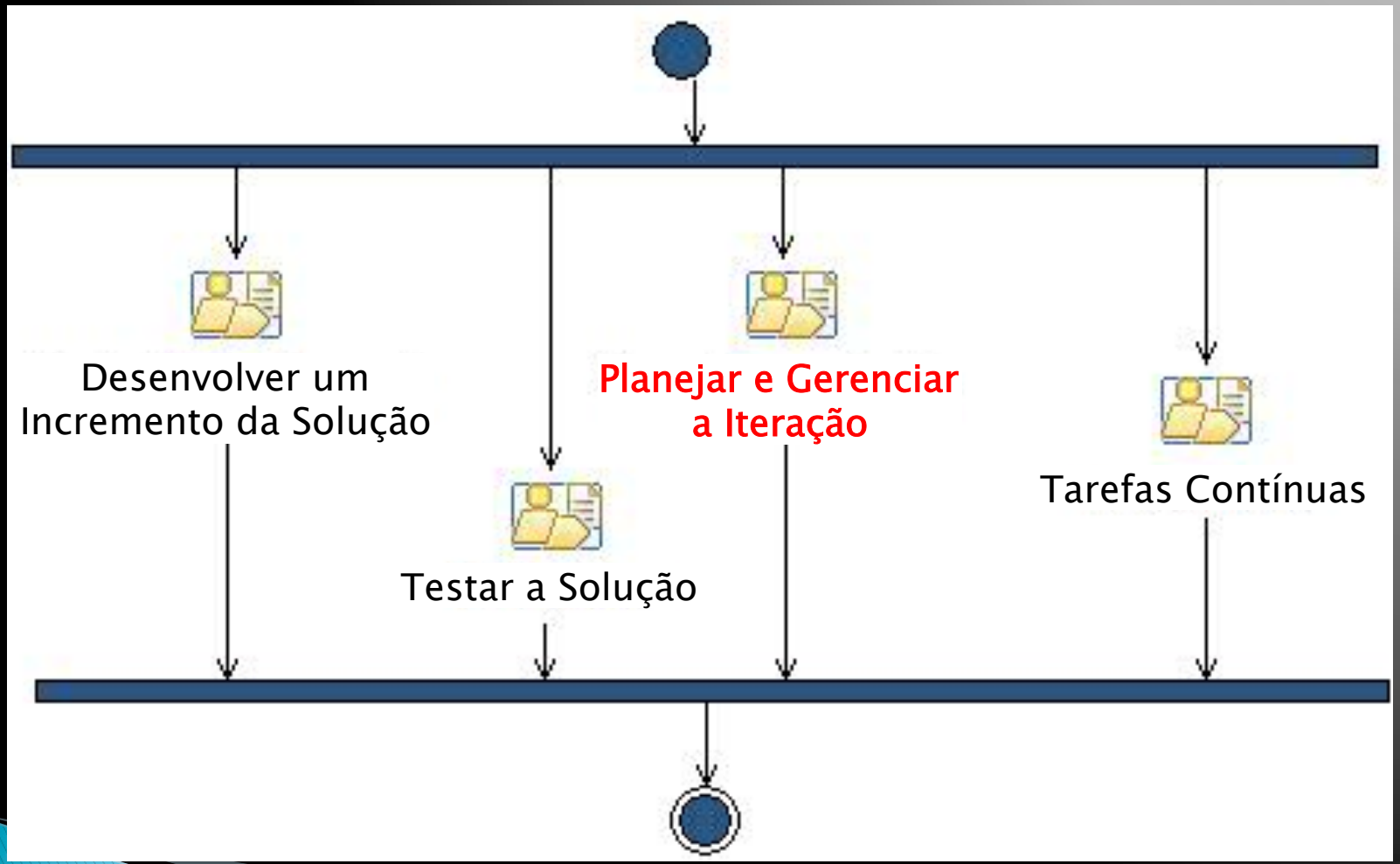
Teste de Requisitos Qualitativos

- Tipos de teste de performance:
  - Teste de carga → usado para validar e avaliar a aceitabilidade dos limites operacionais do sistema dada a variação de carga de trabalho a que o mesmo é submetido
  - Perfilamento de performance → teste no qual o perfil de tempo é monitorado, incluindo o fluxo de execução, acesso a dados e chamadas ao sistema
  - Teste de configuração → foca em garantir a perfeita funcionalidade do sistema sob diferentes configurações de hardware e software

Teste de Requisitos Qualitativos

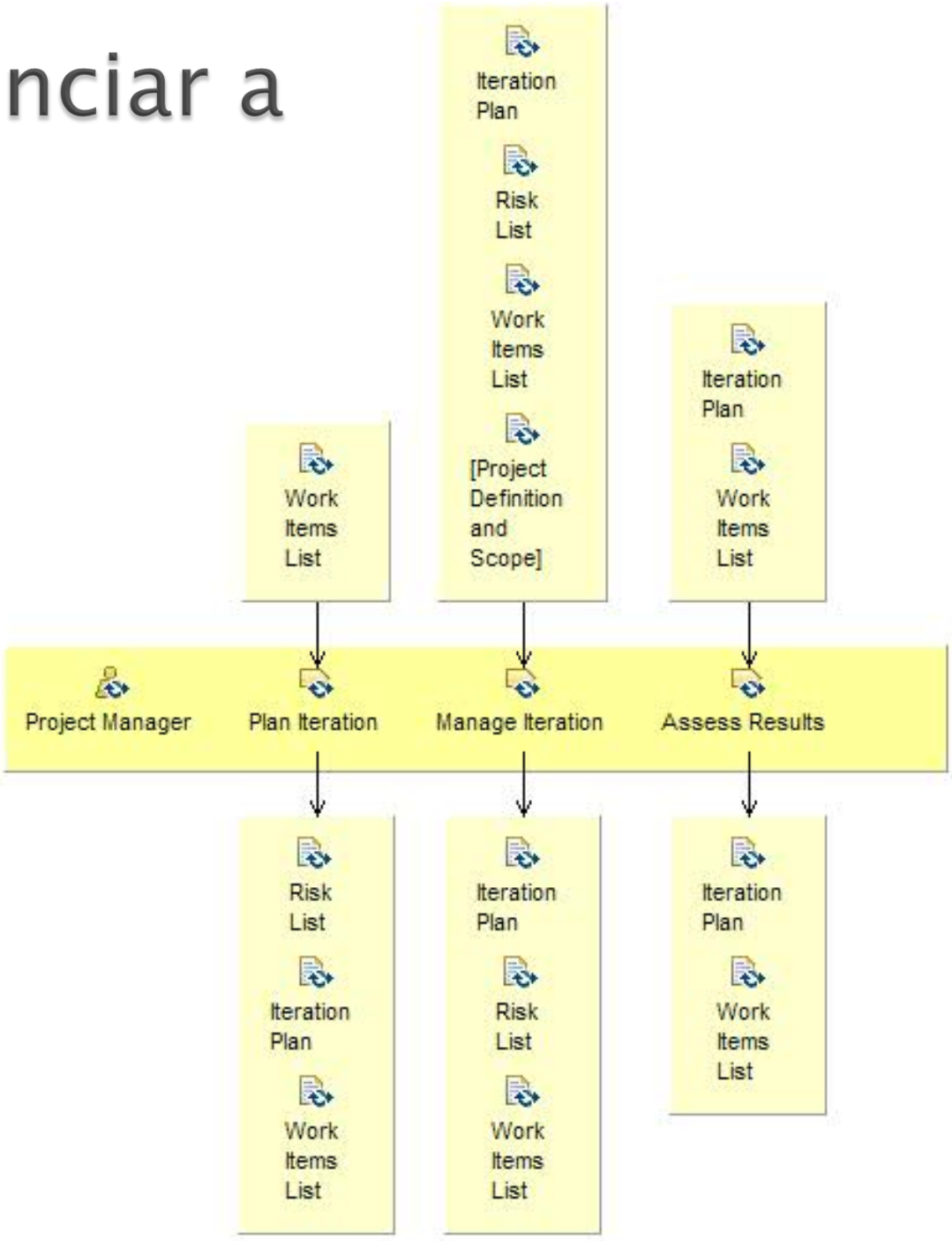
- Tipos de teste de suportabilidade:
  - Teste de instalação → foca em garantir que o sistema instala como previsto em diferentes configurações de hardware e software

Estrutura de Divisão de Trabalho



Planejar e Gerenciar a Iteração

- Distribuição das tarefas de desenvolvimento entre os membros da equipe
- Monitorar e informar o status para os *stakeholders*
- Identificar e gerenciar as exceções e problemas



Planejar e Gerenciar a Iteração

- Esta atividade é realizada durante todo o ciclo de vida do projeto
- O objetivo principal dessa atividade é:
  - a identificação antecipada de riscos e problemas, para que os mesmos possam ser mitigados
  - Estabelecer metas claras e compartilhadas para a iteração (buscando o comprometimento de todos)
  - Dar suporte à equipe de desenvolvimento para que alcancem as metas definidas

Planejar e Gerenciar a Iteração

- Orientações gerais para gerentes e equipe:
  1. Deve ser realizada uma priorização do trabalho atribuído para a iteração
  2. Todos devem concordar com o que será desenvolvido na iteração
  3. Um membro da equipe pode assumir a responsabilidade por "itens de trabalho"
  4. Um membro da equipe quebra um "item de trabalho" em tarefas de desenvolvimento e estima o seu esforço (estimação mais precisa e realista do tempo que se irá gastar)

Planejar e Gerenciar a Iteração

- A medida que a iteração vai acontecendo, a equipe regularmente informa:
  1. O trabalho que foi completado
  2. O trabalho que está sendo realizado
  3. O próximo trabalho a ser realizado
  4. Problemas que estão bloqueando o progresso
- Essas informações podem ser passadas em uma reunião diária (rápida e objetiva)
- Em uma reunião diária a equipe pode discutir:
  - O *status* do trabalho
  - Apontar problemas e riscos (a serem atacados pelo gerente)
  - Propor ajustes ou correções para garantir que os objetivos da iteração sejam atingidos

Planejar e Gerenciar a Iteração

- Durante a avaliação da iteração, a chave para o sucesso é a demonstração que a funcionalidade planejada foi devidamente implementada
- Se o final da iteração coincidir com o final de uma fase, devem também ser observados os objetivos da fase
- Lições aprendidas devem ser devidamente discutidas e incorporadas ao processo, para o melhoramento do mesmo

Tarefa: Planejar a Iteração

- Planejar o escopo e responsabilidades para uma dada iteração
- Passos:
  1. Priorizar a lista de itens de trabalho
  2. Definir os objetivos da iteração
  3. Consignar trabalho à iteração
  4. Identificar e rever os riscos
  5. Definir um critério de avaliação
  6. Refinar a definição e escopo do projeto

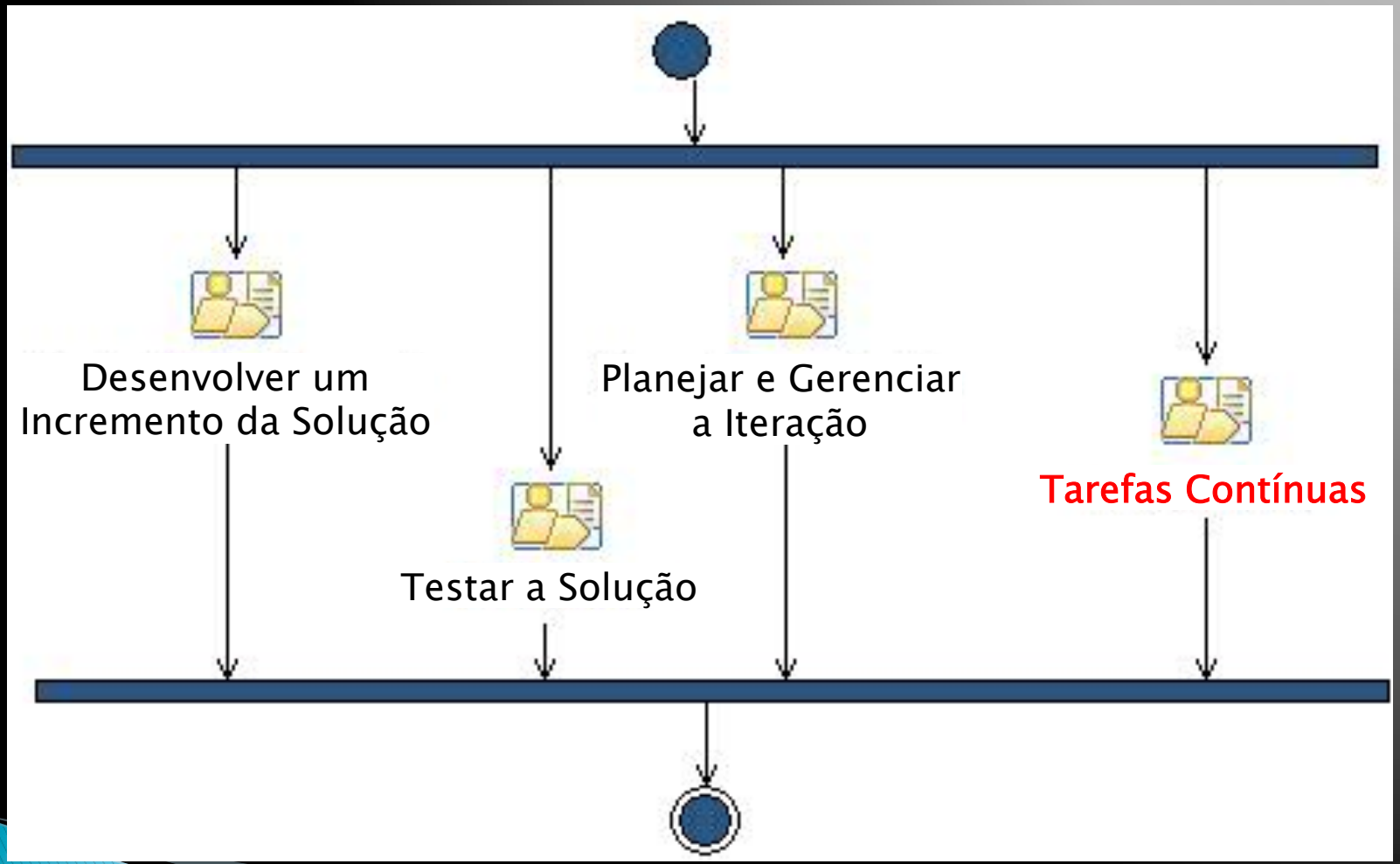
Tarefa: Gerenciar a Iteração

- Avaliar o status do projeto, identificando obstáculos e oportunidades
- Identificar e gerenciar as exceções, problemas e riscos
- Passos:
  1. Rastrear o progresso da iteração corrente
  2. Capturar e noticiar o status do projeto
  3. Gerenciar exceções e problemas
  4. Identificar e gerenciar os riscos
  5. Gerenciar os objetivos

Tarefa: Avaliar os Resultados

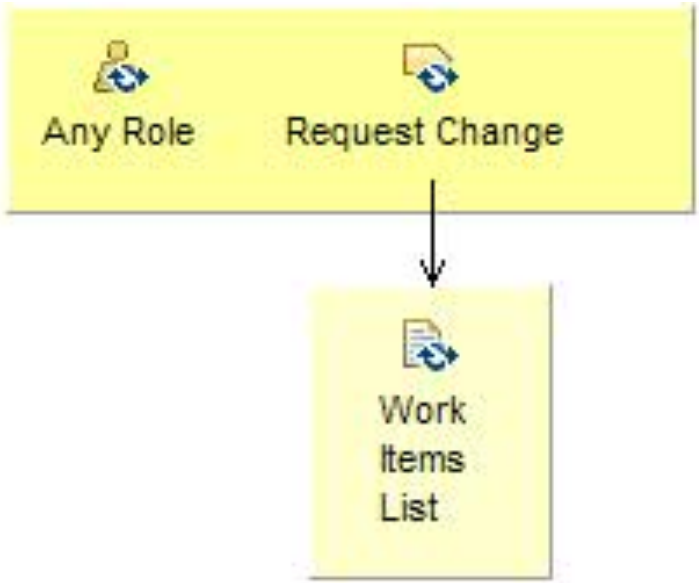
- Determinar o sucesso ou falha da iteração
- Aplicar as lições aprendidas para modificar o projeto e aperfeiçoar o processo
- Passos:
  1. Preparar para a avaliação da iteração
  2. Demonstrar o valor e obter feedback
  3. Realizar uma retrospectiva
  4. Realizar uma retrospectiva (final da fase)
  5. Reunião de finalização do projeto (última iteração)

Estrutura de Divisão de Trabalho



Registrar Correções ou Mudanças

- Atividade que não faz parte do cronograma
- A tarefa de **requisitar mudança** pode ocorrer durante o ciclo de vida em resposta à observação de um defeito, um melhoramento desejado, ou uma solicitação de mudança



Tarefa: Solicitação de Mudança

- Qualquer membro da equipe pode ser responsável por registrar uma solicitação de mudança, os mais comuns são
  - *Stakeholders* – melhoramentos ou mudanças
  - Testadores – relato de defeitos
- Capturar e registrar requisições de mudança
- Passos:
  1. Obter informações sobre a solicitação de mudança
  2. Atualizar a lista de itens de trabalho