Engenharia de Software

Módulo 03

Prof. Daniel Caixeta

Conteúdo programático

01 Processo Unificado

1.1. Conceitos iniciais, breve histórico e evolução.
1.2. Caracterização do Processo Unificado.
    1.2.1. Dirigido por caso de uso/*use case*.
    1.2.2. Iterativo e incremental.
    1.2.3. Focado nos riscos.
1.3. As fases do processo.
1.4. As boas práticas [...].
1.5. Os *workflows* [...].
    1.5.1. A classificação.
1.6. Outros modelos baseados no RUP.
    1.6.1. AUP - *Agile Unified Process*.
    1.6.2. OpenUP - *Open Unified Process*.
    1.6.3. OUM – *Oracle Unified Method.*
    1.6.4. RUP-SE - *Rational Unified Process-Systems Engineering.*

1.1. Conceitos iniciais.

Breve introdução e histórico [...]

1.1. Introdução

- *Rational Unified Process* – RUP (SOMMERVILLE, 2018) é um modelo de processo moderno, derivado de trabalhos sobre a UML e do *Unified Software Development Process* (RUMBAUGH, *et al*., 1999; ARLOW & NEUSTADT, 2005).
- É considerado um processo híbrido que além de reunir elementos de todos os modelos de processos genéricos, ilustra boas práticas na especificação, com apoio na prototipação e entrega incremental.
- O RUP reconhece que os modelos convencionais apresentam uma visão única do processo, que normalmente são descritos em três perspectivas:

    i. Dinâmica, que mostra as fases do modelo ao longo do tempo.
    ii. Estática, que mostra as atividades realizadas no processo.
    iii. Prática, que sugere boas formas a serem usadas durante o processo.

- De acordo com Pressman (2011), o Processo Unificado é uma tentativa de aproveitar os melhores recursos e características dos modelos tradicionais de processo de *software*, mas caracterizando-os de modo a implementar muitos dos melhores princípios do desenvolvimento ágil de *software*.
- Enfatiza a importância da arquitetura de *software* mantendo o foco nas metas estimadas, assim como na compreensibilidade, confiança em mudanças futuras e reutilização. (*ibidem apud* JACOBSON, BOOCH & RUMBAUGH, 1999).
- O PU / UP (Processo Unificado / *Unified Process*) sugere um fluxo interativo e incremental, proporcionando a sensação evolucionária que é essencial no desenvolvimento de *software* moderno, além de atender às necessidades específicas do projeto. (*ibidem*).

Um breve histórico [...]

- Durante o início dos anos 1990, James Rumbaugh, Grady Booch e Ivar Jacobson começaram a trabalhar em um "método unificado" que combinaria as melhores características de cada um de seus métodos individuais de análise e projeto O.O e adotaram características adicionais propostas por outros especialistas em modelagem orientada a objetos. (PRESSMAN, 2011).
- O resultado foi a UML[1] - uma linguagem de modelagem unificada que contém uma notação robusta para a modelagem e o desenvolvimento de sistemas O.O. (*ibidem*).
- No ano de 1997, a UML tornou-se um padrão da indústria de desenvolvimento de *software* orientado a objeto. (*ibidem*).

1. *Unified Modeling Language*.

1.2. As características.

Processo Unificado (PU) / *Unified Process* (UP)

1.2. Caracterização do Processo Unificado/*Unified Process*

- O PU/UP é mais do que um processo:

  É um framework extensível para a concepção de processos, podendo ser adaptado às características específicas de diferentes empresas e projetos. (WAZLAWICK, 2013).

- As principais características são:

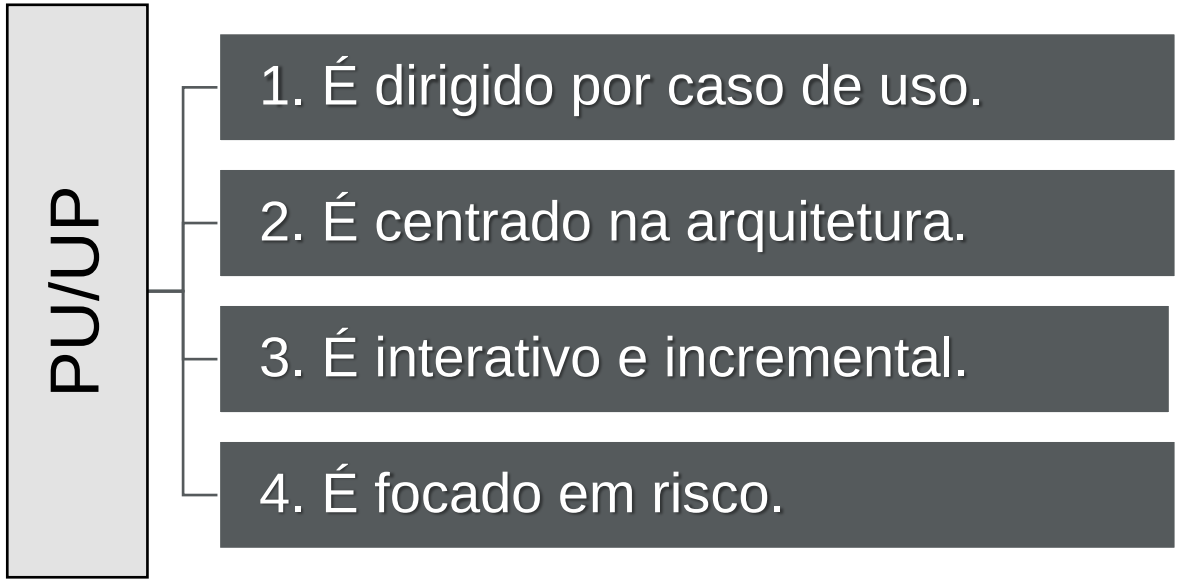


Figura 1. As principais características do Processo Unificado.

1.2.1. Dirigido por caso de uso/*use case*[2]

- Segundo Wazlawick (2013), o UC é um processo compreendido do ponto de vista do usuário. São úteis para várias atividades relacionadas ao desenvolvimento de um sistema, entre elas:

  a) Definição e validação da arquitetura do sistema: Em geral, classes e atributos são obtidos a partir dos textos dos UC expandidos.

  b) Criação dos casos de teste: os UC podem ser vistos como um roteiro para o teste de sistema e de aceitação, em que as funcionalidades são testadas do ponto de vista do cliente.

  c) Planejamento das iterações: São desenvolvidos com base estimativa no número de interações do usuário com o sistema.

  d) Base para a documentação do usuário: Os casos de uso (UC) são descrições de fluxos normais e alternativos de operação de um sistema. Essas descrições são excelentes para iniciar o manual de operação do sistema, pois todas as funcionalidades possíveis estarão descritas aí de forma estruturada e completa.

2. De agora em diante identificaremos o termo Caso de Uso ou Use Case através da sigla UC.

- Porém, a aplicação mais fundamental do UC é a incorporação dos requisitos funcionais de forma organizada.
- Cada passo dos fluxos principais e alternativos corresponde a uma função do sistema.
- Requisitos não funcionais podem ser anotados juntamente com os UC ou anotados em um documento à parte.

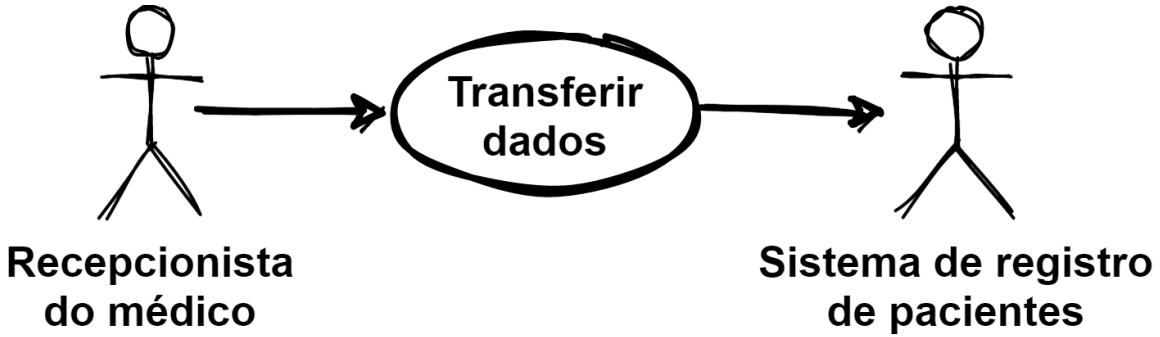


Figura 2. UC de transferência de dados. (SOMMERVILLE, 2018).

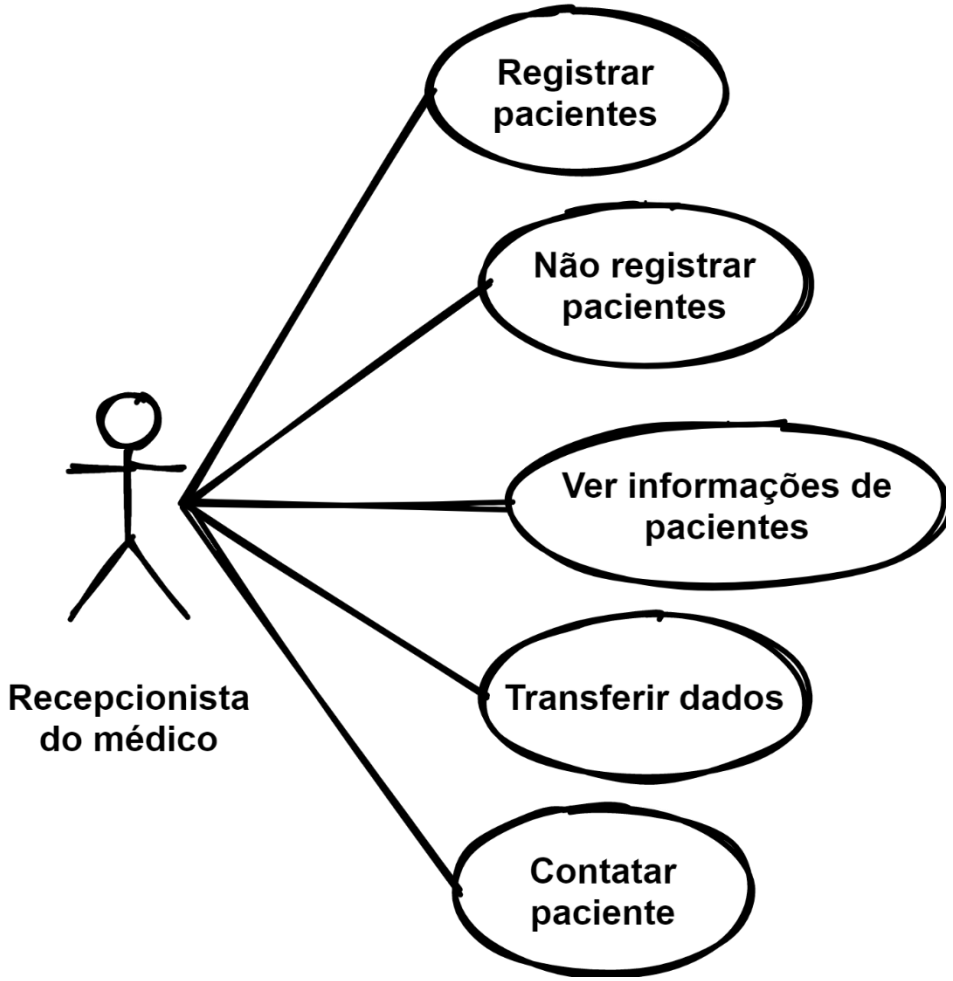


Figura 3. UC envolvendo atividades de uma recepcionista. (SOMMERVILLE, 2018).

1.2.2. Iterativo e incremental

- Para Wazlawick (2013), o PU preconiza o desenvolvimento baseado em ciclos iterativos de duração fixa, em que, a cada iteração, a equipe incorpora à arquitetura as funcionalidades necessárias [...].
- Ressalta-se que cada ciclo iterativo produz um incremento no *design* do sistema, seja produzindo mais conhecimento sobre seus requisitos e arquitetura, seja produzindo um código executável.
- As principais vantagens da integração contínua:
    - Redução de riscos no desenvolvimento;
    - Facilidade nos testes e validação; e
    - Melhora no aprendizado da equipe.

1.2.3. Focado em riscos

- Em função das priorizações dos UC mais críticos nos primeiros ciclos iterativos, dizemos o PU é focado em riscos. (WAZLAWICK, 2013).
- Se esses UC são os que apresentam maiores riscos, então devem ser tratados com prioridade enquanto o custo ainda é baixo e o tempo disponível para lidar com as surpresas ainda é relativamente grande.

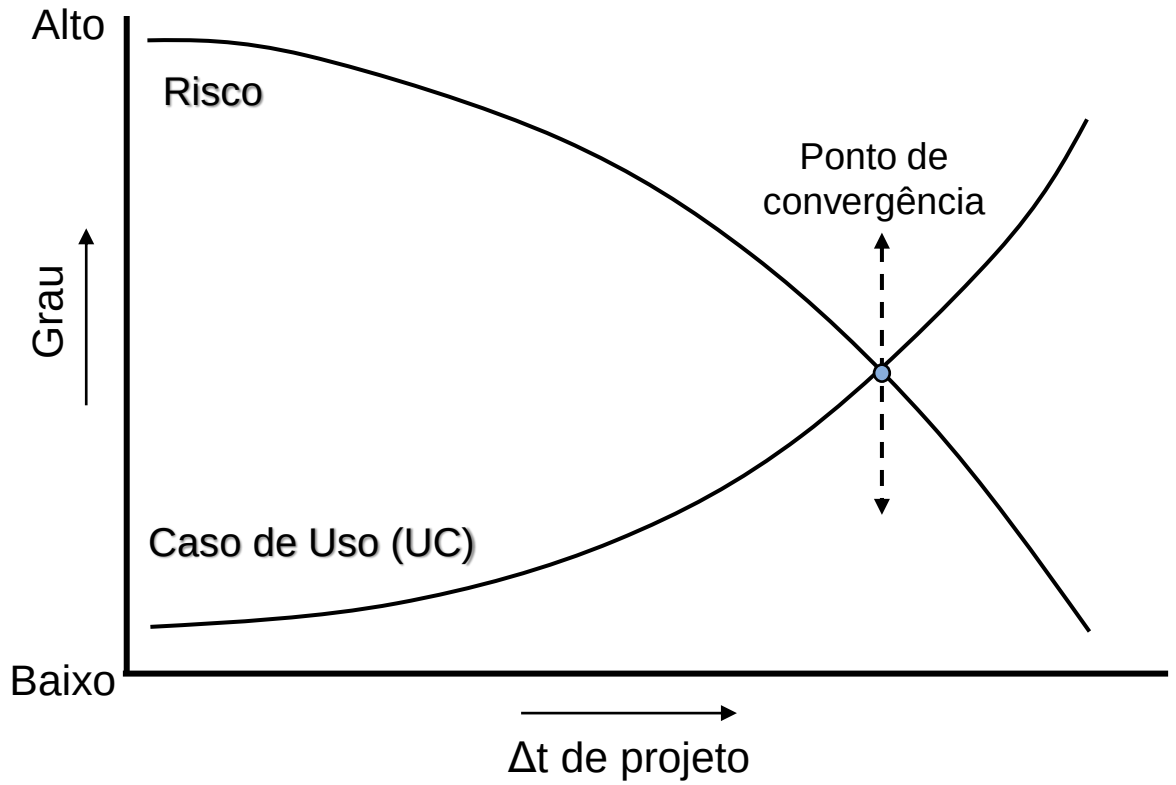


Figura 4. Fator de risco de um UC. (*adaptado* CAIXETA, 2020).

1.3. As fases do processo.

Concepção, elaboração, construção e transição.

1.3. As fases [...]

- No RUP são identificadas quatro fases no processo de desenvolvimento de *software*. (SOMMERVILLE, 2018). São elas:
  - Concepção: Estabelece um *business case* para o sistema. Identifica todas as entidades externas (pessoas e sistemas) que vão interagir com o sistema e definir as interações.
  - Elaboração: As metas desta fase são desenvolver uma compreensão do problema dominante, estabelecer um *framework* da arquitetura para o sistema, desenvolver o plano do projeto e identificar seus maiores riscos. No fim dessa fase, é apresentado um modelo de requisitos que pode ser um conjunto de casos de uso da UML, uma descrição da arquitetura ou um plano de desenvolvimento do *software*.

- Construção: Envolve a programação e testes do sistema. Durante essa fase, as partes são desenvolvidas em paralelo e integradas. Na conclusão, um sistema de *software* já deve estar funcionando, bem como a documentação pronta para ser entregue aos usuários.
- Transição: A fase final do RUP implica em transferência do sistema de *software* da equipe de desenvolvimento para a comunidade de usuários e em seu funcionamento em um ambiente real. [...]. Na conclusão dessa fase, o sistema deve estar funcionando corretamente em seu ambiente operacional e documentado.

cont. [...]

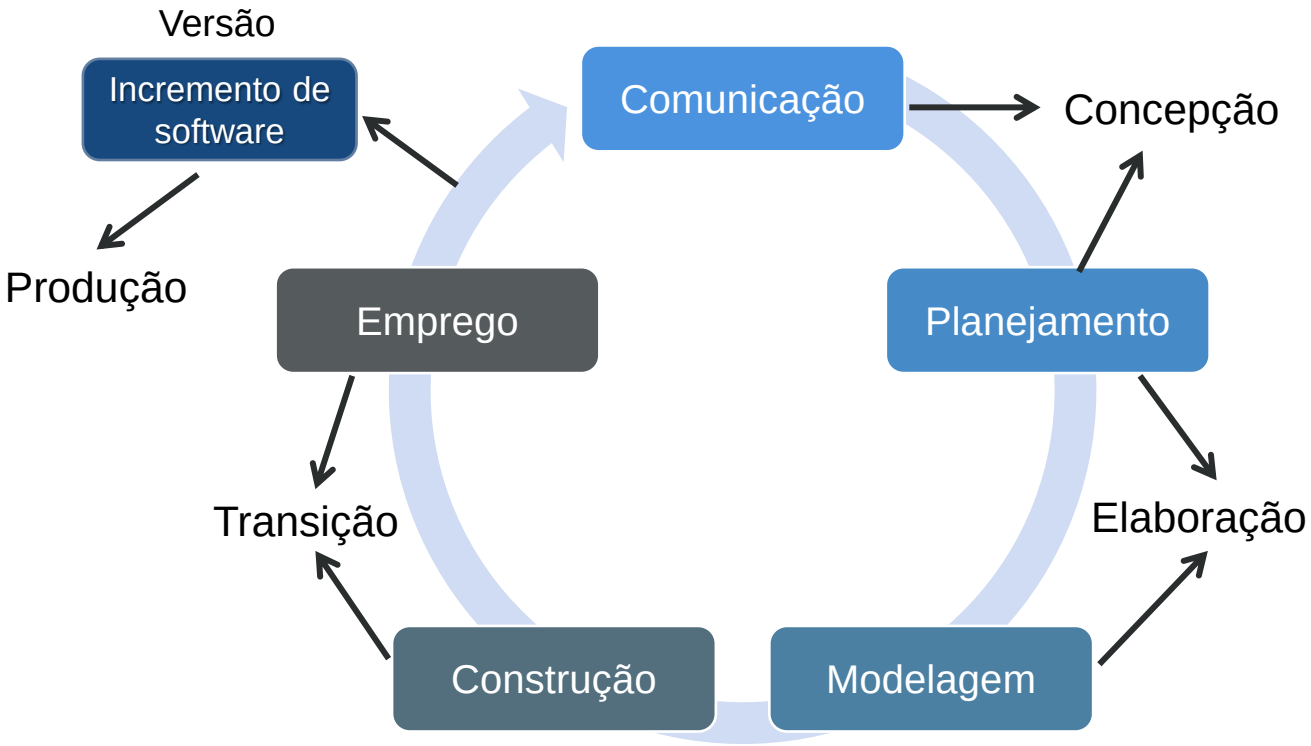


Figura 5. Fluxo genérico das fases RUP. (PRESSMAN, 2011).

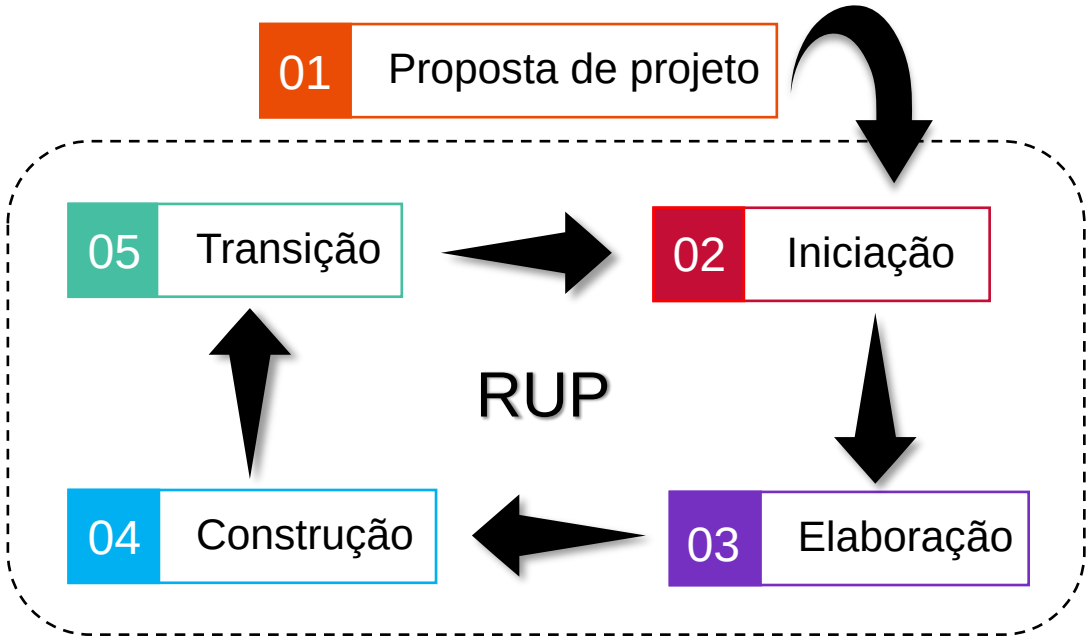


Figura 6. Fluxo comum das fases RUP.

1.4. As boas práticas [...].

Como deve ser feito [...].

1.4. As boas práticas no(a) RUP [...]

- Na perspectiva do(a) RUP, sobre boas práticas na engenharia de software, recomenda-se seis consideradas fundamentais. São elas:
    1. Desenvolver *software* iterativamente: Planejar os incrementos do sistema com base nas prioridades do cliente e desenvolver os recursos de alta prioridade no início do processo de desenvolvimento.
    2. Gerenciar os requisitos: Documentar os requisitos do cliente e acompanhar suas mudanças. Analisar o impacto destas mudanças no sistema antes de aceitá-las.
    3. Adotar arquiteturas baseadas em componentes: Estruturar a arquitetura do sistema em componentes.
    4. Modelar o *software* visualmente: Usar modelos gráficos da UML para apresentar visões estáticas e dinâmicas do *software*.

5. Verificar a qualidade do *software*: Assegurar que o *software* atenda aos padrões de qualidade organizacional e das solicitações do cliente.
6. Controlar as mudanças do *software*: Gerenciar as mudanças, usando sistema, procedimentos e ferramentas de gerenciamento de configurações e versões.

- É bom ressaltar que o(a) RUP não é um processo adequado para todos os tipos de desenvolvimento, *e.g.*, *software* embutido. No entanto, ele representa uma abordagem que potencialmente combina os três modelos de processo genéricos (cascata, incremental e orientada a reuso). (SOMMERVILLE, 2018).
- As inovações mais importantes são a separação de fases e *workflows* e o reconhecimento de que a implantação de *software* em um ambiente do usuário é parte do processo [...]. (*ibidem*).

1.5. Os *workflows* do(a) RUP.

Os centrais e os de apoio.

1.5. Os *workflows* [...]

- Os *workflows* segundo Wazlawick (2013) definem:

  > Um conjunto de atividades e um conjunto de papéis responsáveis por uma atividade. Além disso, indica as dependências entre as diferentes atividades, e quais dependem logicamente de outras atividades para poderem ser executadas.

- De acordo com a *Rational* (*ibidem*) existem três tipos de *workflows*. São eles:
  1. *Workflow* núcleo: Define a forma geral de condução de uma dada disciplina.
  2. *Workflow* detalhe: Apresenta um refinamento do *workflow* núcleo, indicando atividades em um nível mais detalhado, bem como artefatos de entrada e saída de cada atividade.
  3. Planos de iteração: Consistem em uma instanciação do processo para uma iteração específica. [...].

1.5.1. A classificação dos *workflows* [...]

- De acordo com o/a RUP, os *workflows* são orientados em torno de modelos associados à UML, *e.g.*, modelos de sequência, de objetos, de classe, etc.
- A vantagem de proporcionar visões estáticas e dinâmicas é que as fases de desenvolvimento não estão associadas a *workflows* específicos. A princípio, todos podem ser ativos em todas as fases do processo. (SOMMERVILLE, 2018).
- São classificado em centrais (seis) e de apoio (três).

cont. [...]

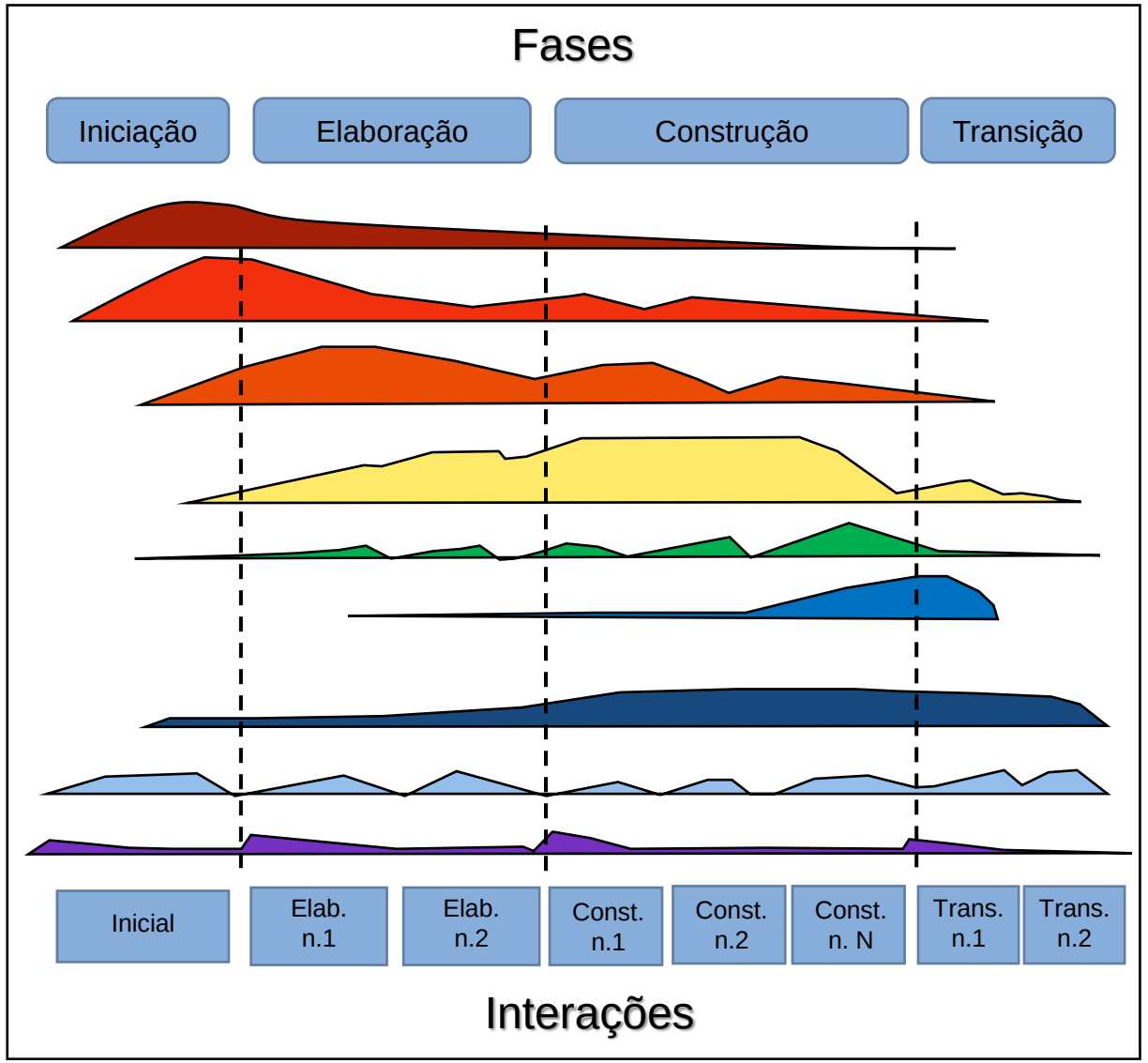


Figura 7. Ciclo de vida do RUP.

cont. [...]

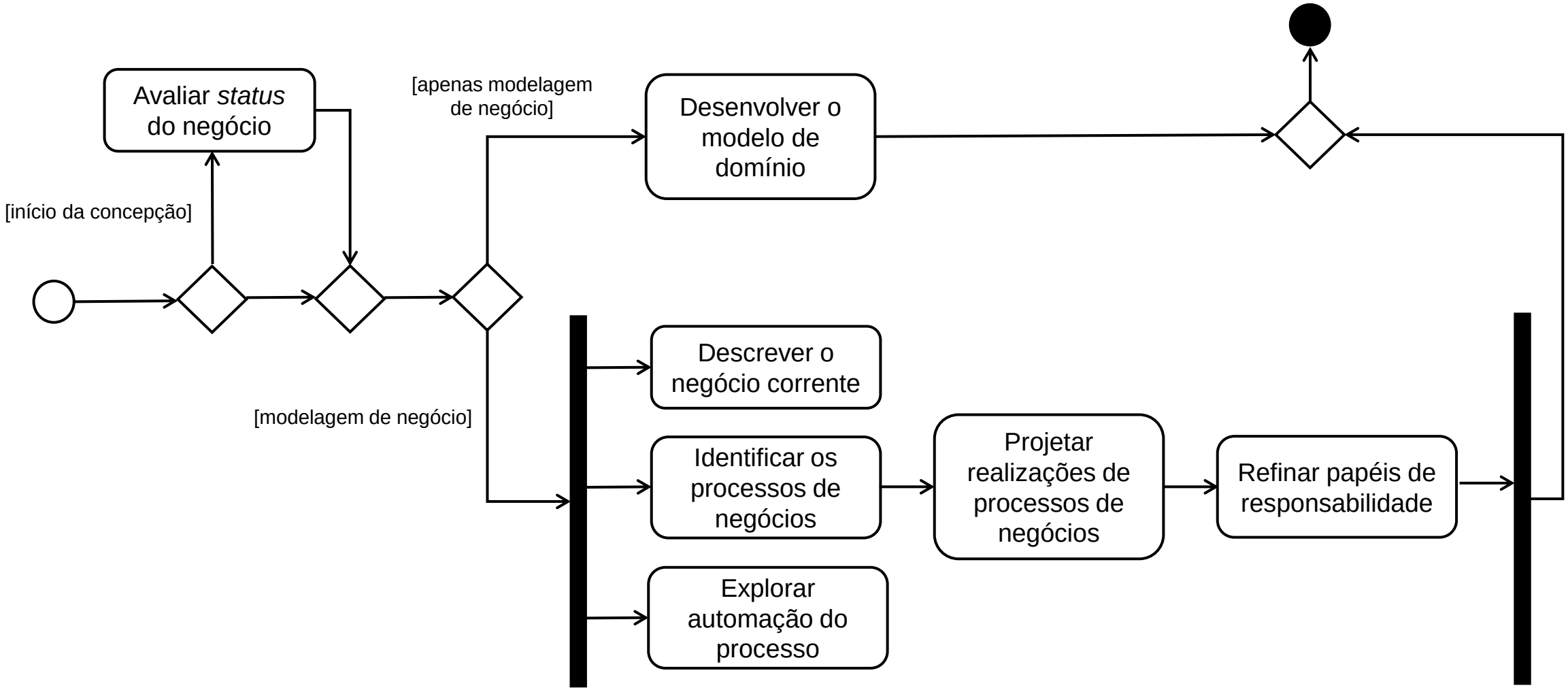


Figura 8. *Workflow* central da disciplina de modelagem de negócio. (WAZLAWICK, 2013).

cont. [...]

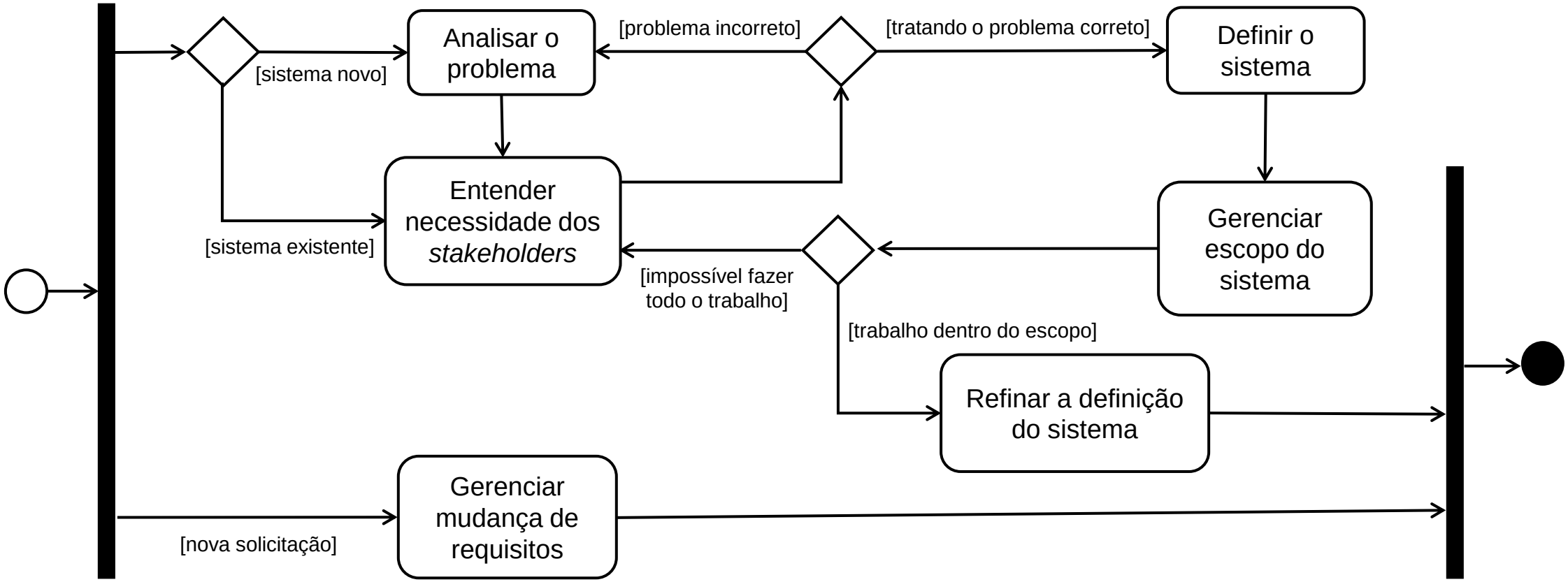


Figura 9. *Workflow* central da disciplina de requisitos. (WAZLAWICK, 2013).

cont. [...]

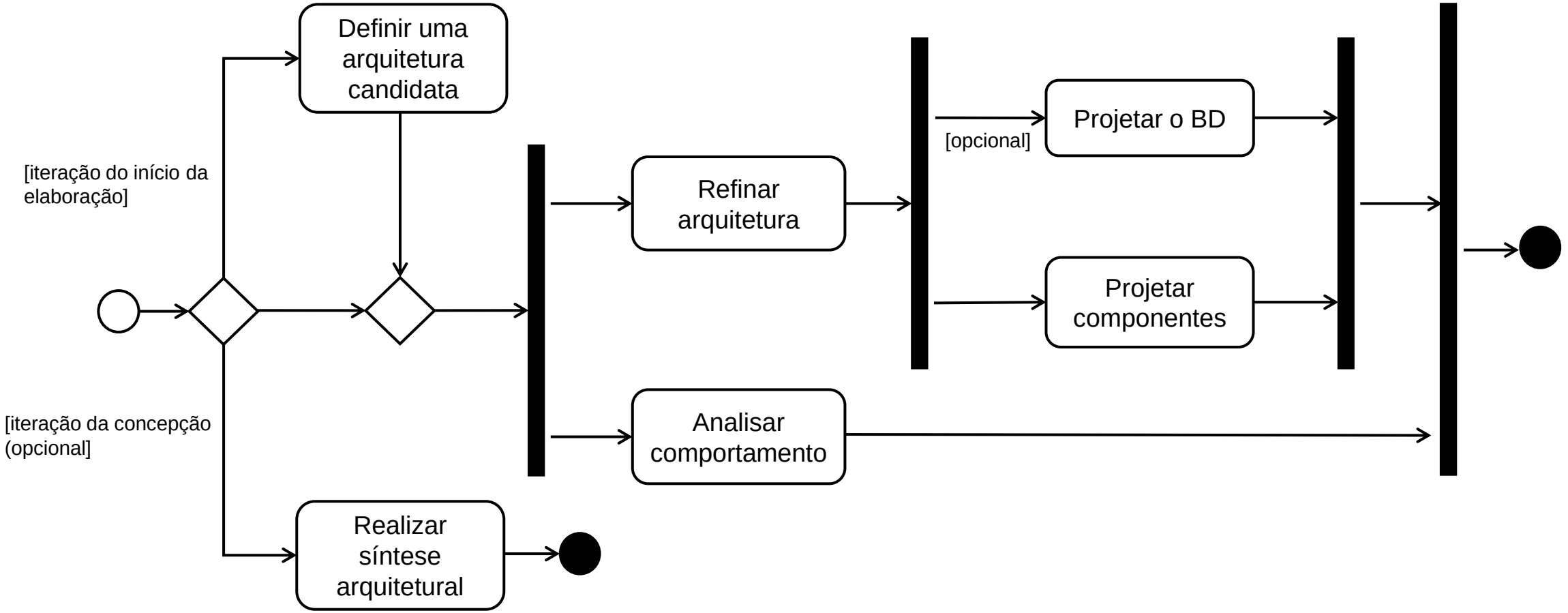


Figura 10. *Workflow* central da disciplina de análise e projeto (*design*). (WAZLAWICK, 2013).

cont. [...]

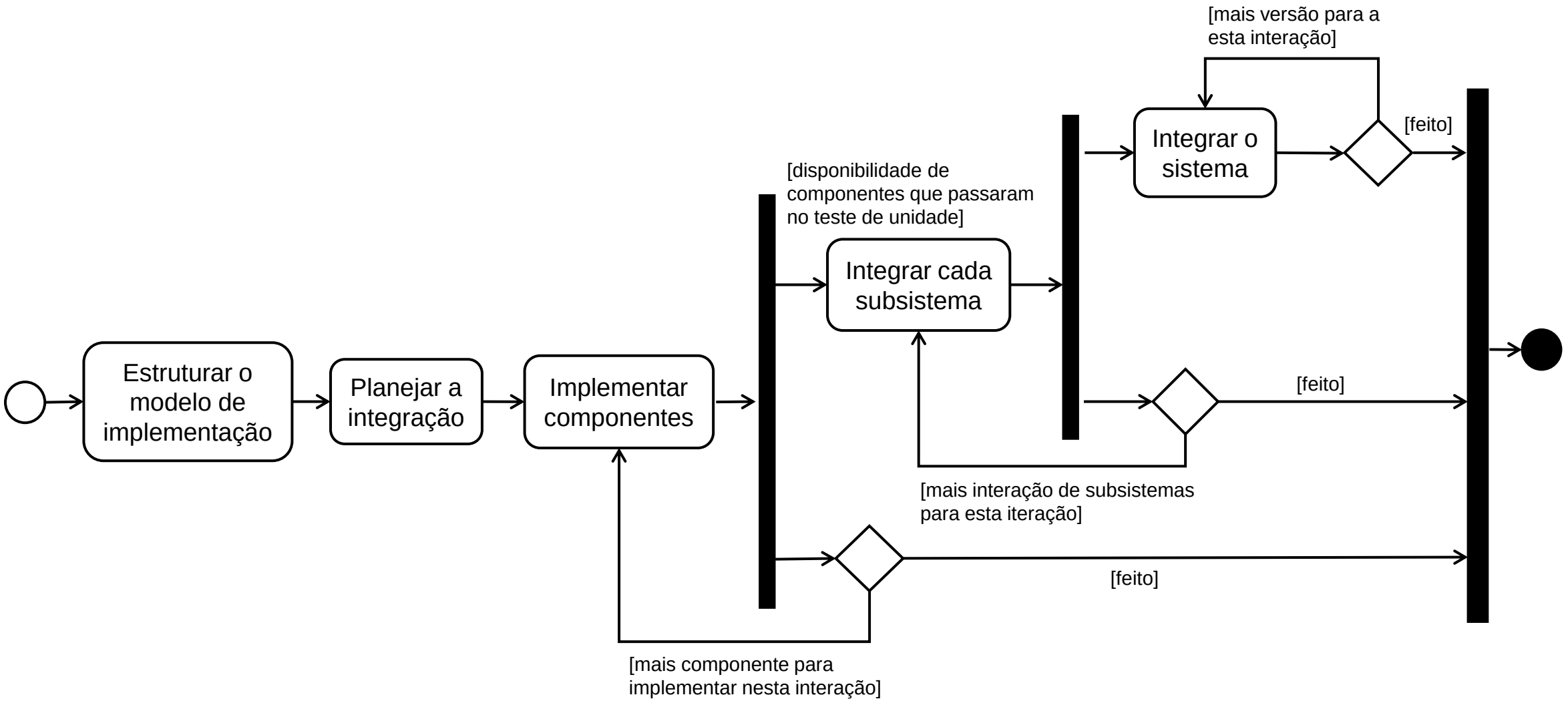


Figura 11. *Workflow* central da disciplina de implementação. (WAZLAWICK, 2013).

cont. [...]

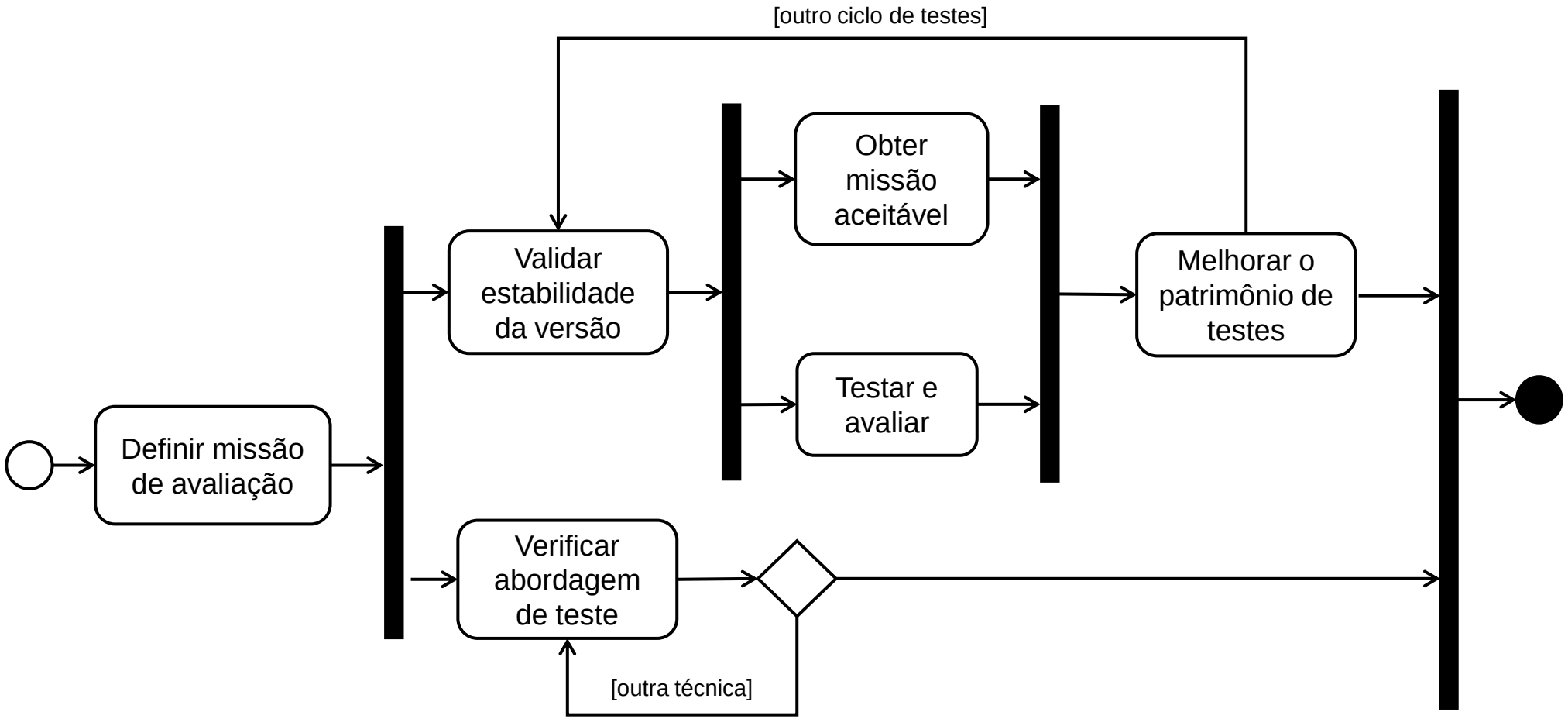


Figura 12. *Workflow* central da disciplina de testes. (WAZLAWICK, 2013).

cont. [...]

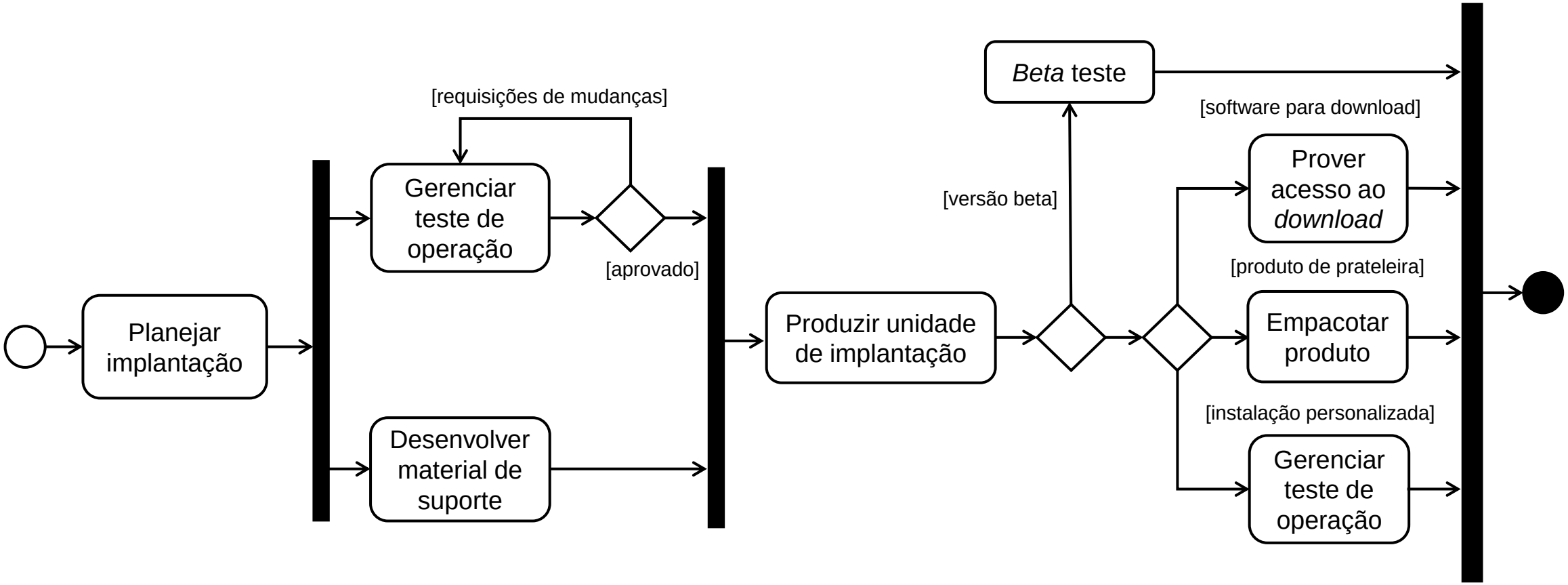


Figura 13. *Workflow* central da disciplina de implantação. (WAZLAWICK, 2013).

cont. [...]

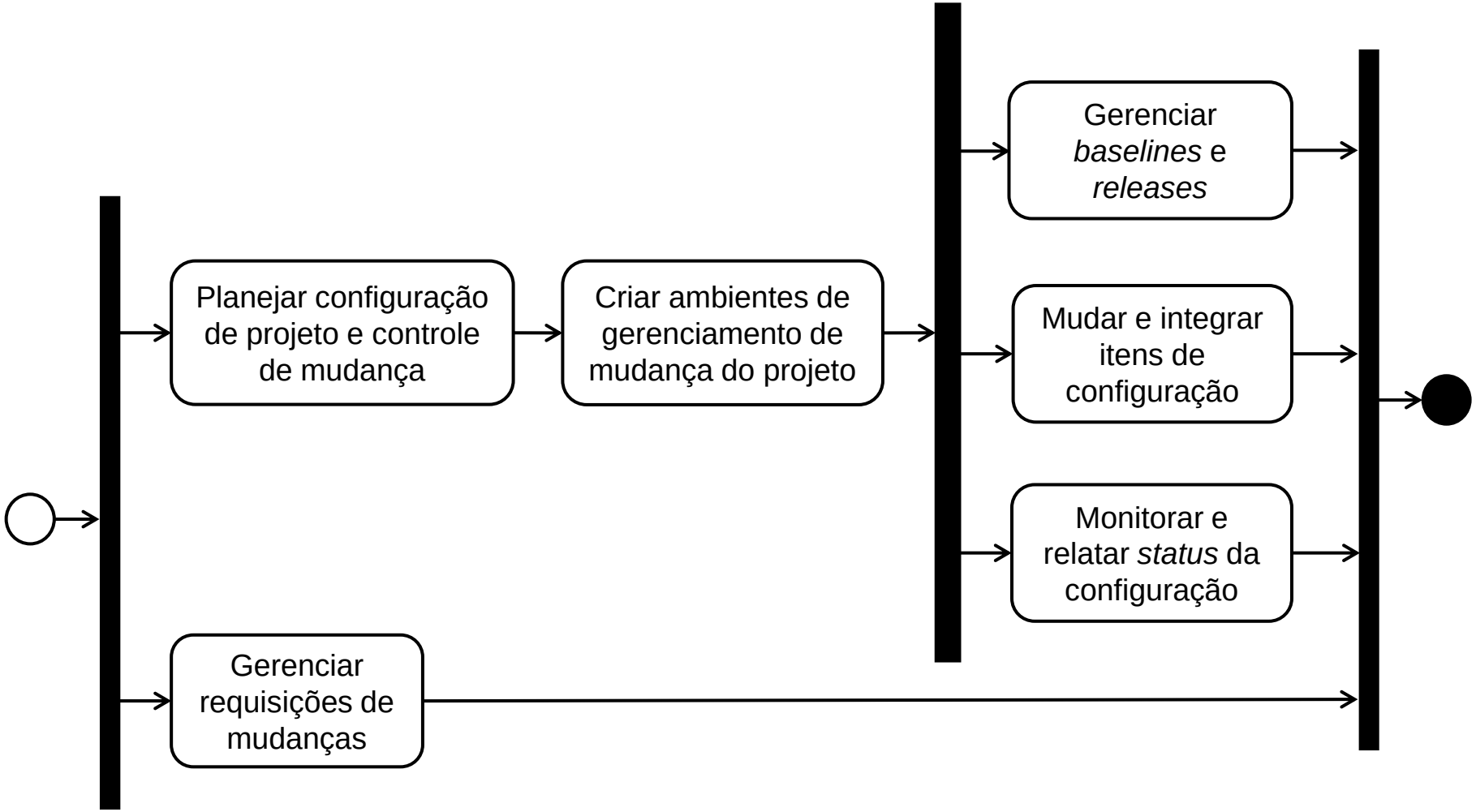


Figura 14. *Workflow* de apoio da disciplina de gerenciamento de configuração e mudanças. (WAZLAWICK, 2013).

cont. [...]

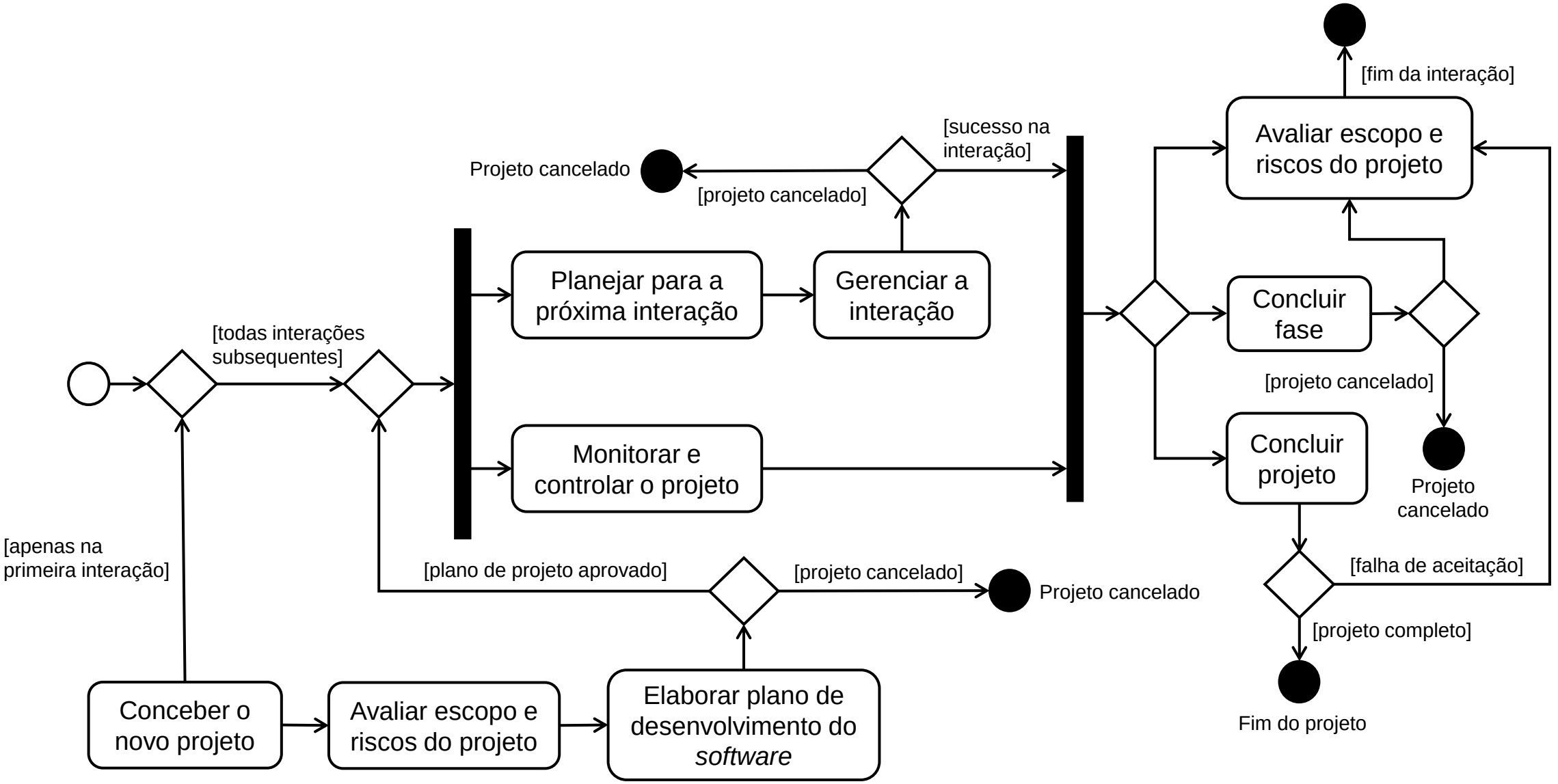


Figura 15. *Workflow* de apoio da disciplina de gerenciamento de projeto. (WAZLAWICK, 2013).

cont. [...]

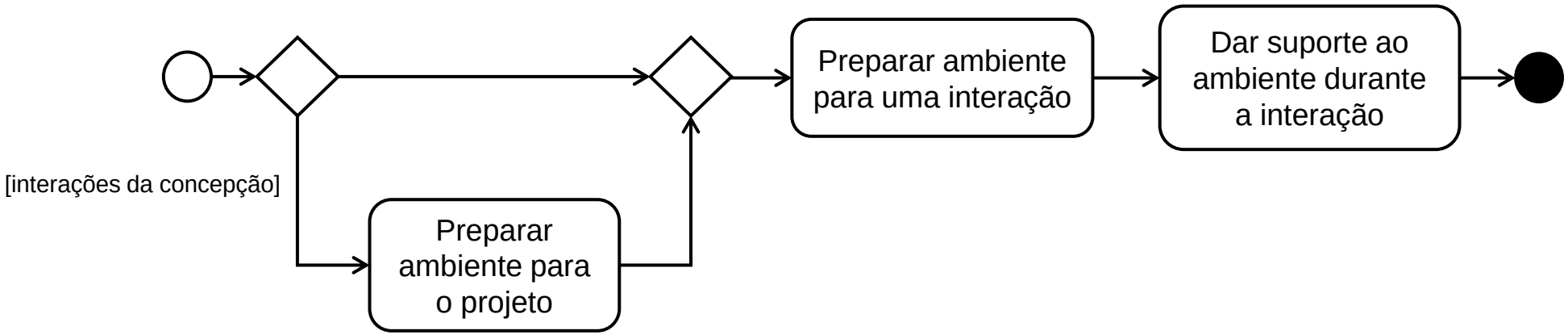


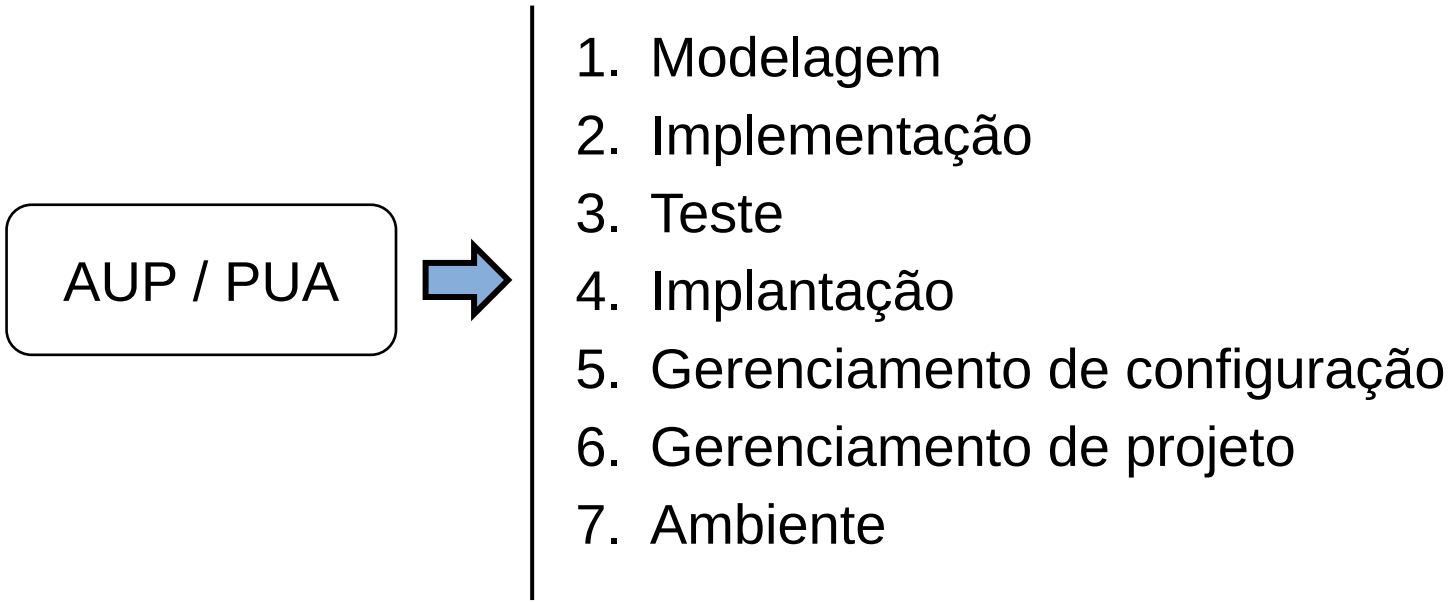
Figura 16. *Workflow* de apoio da disciplina de ambiente. (WAZLAWICK, 2013).

1.6. Outros modelos

Versões de PU.

1.6.1. AUP - *Agile Unified Process.* (WAZLAWICK, 2013).

- O *Agile Unified Process (AUP)* / Processo Unificado Ágil (PUA), é uma versão simplificada do RUP. (Ambler & Jeffries, 2002).
- Aplica técnicas ágeis como desenvolvimento dirigido por testes (TDD - *Test Driven Development*), modelagem ágil e refatoração.
- Ao contrário do RUP, o AUP possui sete disciplinas:



1.6.2. OpenUP - *Open Unified Process.* (WAZLAWICK, 2013).

- Anteriormente conhecido como *Basic Unified Process* (BUP) ou *OpenUP/Basic*, o *Open Unified Process / Processo Unificado Aberto* (OpenUP) é uma implementação aberta (*open source*[3]) desenvolvida como parte do *Eclipse Process Framework* (EPF)[4].
- A primeira versão do modelo foi originada pela IBM.
- Entre 2005 e 2006, essa versão foi abraçada pela Fundação Eclipse.
- O OpenUP aceita, embora de forma simplificada, a maioria dos princípios UP. Porém, é um método independente, onde não são exigidos grande precisão e detalhes nos documentos.

3. Código aberto.
4. Disponível em: <www.methodsandtools.com/archive/archive.php?id=69p2>. Acesso em: 20 ago. 2023.

1.6.3. OUM – *Oracle Unified Method.* (WAZLAWICK, 2013).

- O *Oracle Unified Method*, ou OUM (Oracle, 2009), é um *framework* de processo de desenvolvimento de *software* iterativo e incremental adequado a uso com produtos Oracle: bancos de dados, aplicações e *middleware*.
- É uma implementação do Processo Unificado que suporta, entre outras características, *Service Oriented Architecture* (SOA), *Enterprise Integration*, *software* personalizado, gerenciamento de identidade (*Identity Management*, IdM), governança, risco e adequação (*Governance*, *Risk and Compliance*, GRC), segurança de banco de dados, gerenciamento de performance e inteligência empresarial.
- É, ao mesmo tempo, uma instanciação do Processo Unificado e um modelo orientado a ferramentas da Oracle.
- São quatorze as disciplinas do OUM, sendo elas:

OUM

1. Requisitos de negócios
2. Análise de requisitos
3. Análise
4. Design
5. Implementação
6. Teste
7. Implantação
8. Gerenciamento de performance
9. Arquitetura técnica
10. Aquisição e conversão de dados
11. Documentação
12. Adoção e aprendizagem
13. Transição
14. Operações e suporte

1.6.4. RUP-SE - *Rational Unified Process-Systems Engineering.* (WAZLAWICK, 2013).

- O RUP-SE é uma extensão do modelo RUP para Engenharia de Sistemas. É uma versão de RUP especialmente adequada para o desenvolvimento de sistemas de grande porte, envolvendo *software*, *hardware*, pessoas e componentes de informação.
- O RUP-SE é especialmente adequado a projetos:
    a. Que são grandes o suficiente para comportar várias equipes de desenvolvimento trabalhando em paralelo.
    b. Que necessitam de desenvolvimento concorrente de *hardware* e *software*.
    c. Cuja arquitetura é impactada por questões relativas à implantação.
    d. Que incluem a reengenharia de uma infraestrutura de tecnologia de informação para dar suporte à evolução do negócio.
- O *framework* é disponibilizado como um *plugin* ao modelo RUP original.

BIBLIOGRAFIA

PRESSMAN, R. S. Engenharia de Software: Uma abordagem profissional. 7ª edição. Dados eletrônicos. Porto Alegre: AMGH-McGrawHill, 2011.

SOMMERVILLE, I. Engenharia de Software. 10ª edição. São Paulo: Pearson Prentice Hall, 2018.

WAZLAWICK, R. S. Engenharia de software: conceitos e práticas. Rio de Janeiro: Elsevier, 2013.

Obrigado!

Engenharia de Software