



Engenharia de Software

Unidade 03

Prof. Daniel Caixeta



Conteúdo programático

01 Processo Unificado (PU)

- 1.1. Conceitos iniciais, breve histórico e evolução.
- 1.2. As características.
 - 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* e sua 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*.



Bibliografia



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 iterativo 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 UML¹ - 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:

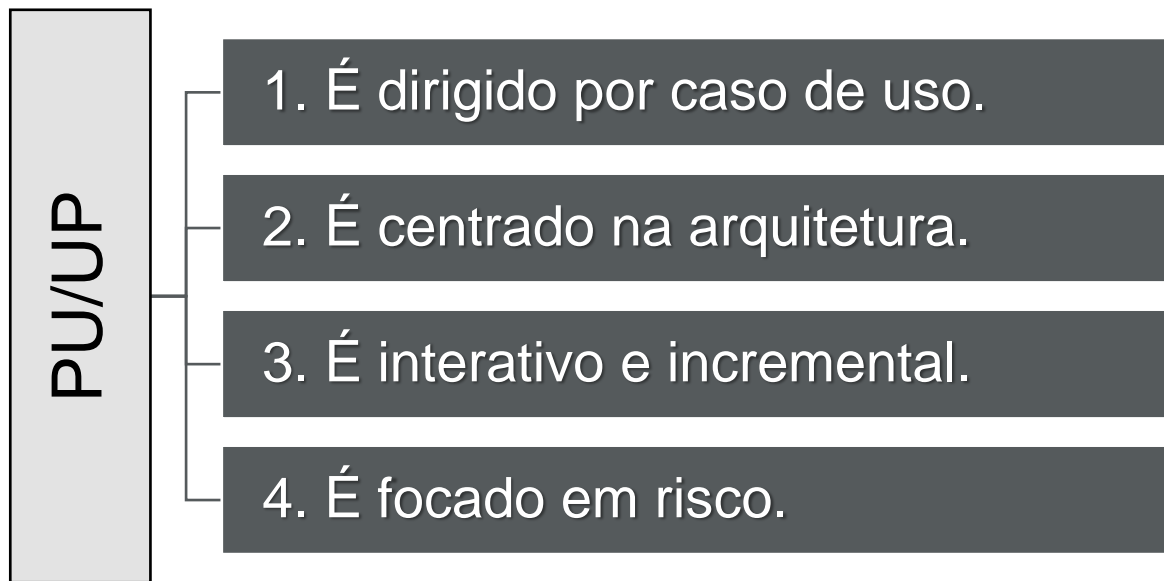


Figura 1. As principais características do Processo Unificado.

1.2.1. Dirigido por caso de uso/*use case*

- 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.
 - b) Criação dos casos de teste.
 - c) Planejamento das iterações.
 - d) Base para a documentação do usuário.
- Porém, a aplicação mais fundamental do UC é a incorporação dos requisitos funcionais de forma organizada, pois cada fluxo principal e/ou alternativo podem corresponder a uma função do sistema.

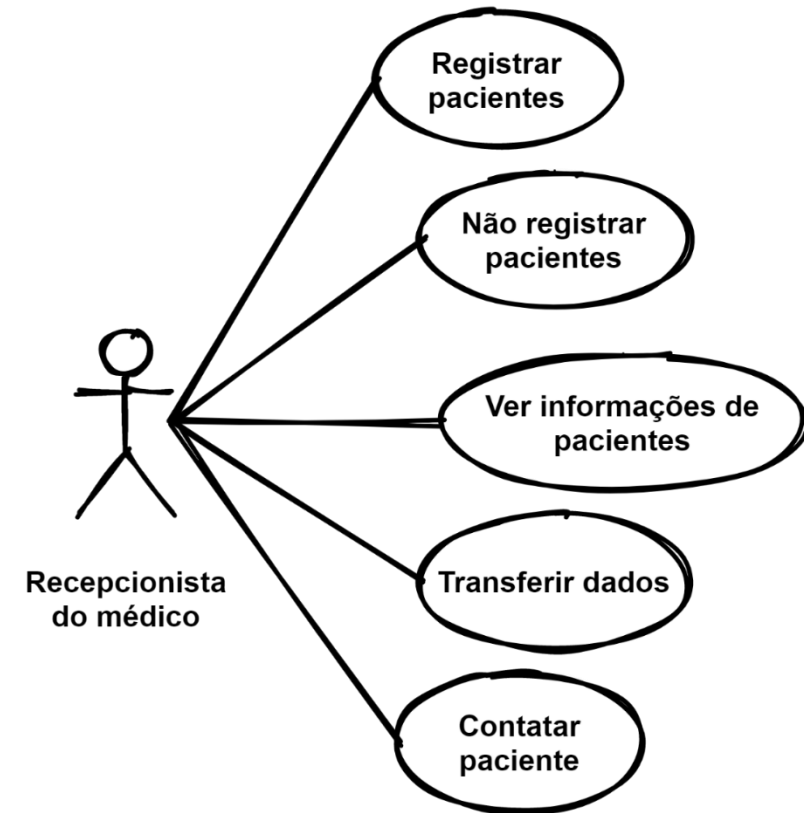


Figura 2. UC envolvendo atividades de uma recepcionista. (SOMMERVILLE, 2018).

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.

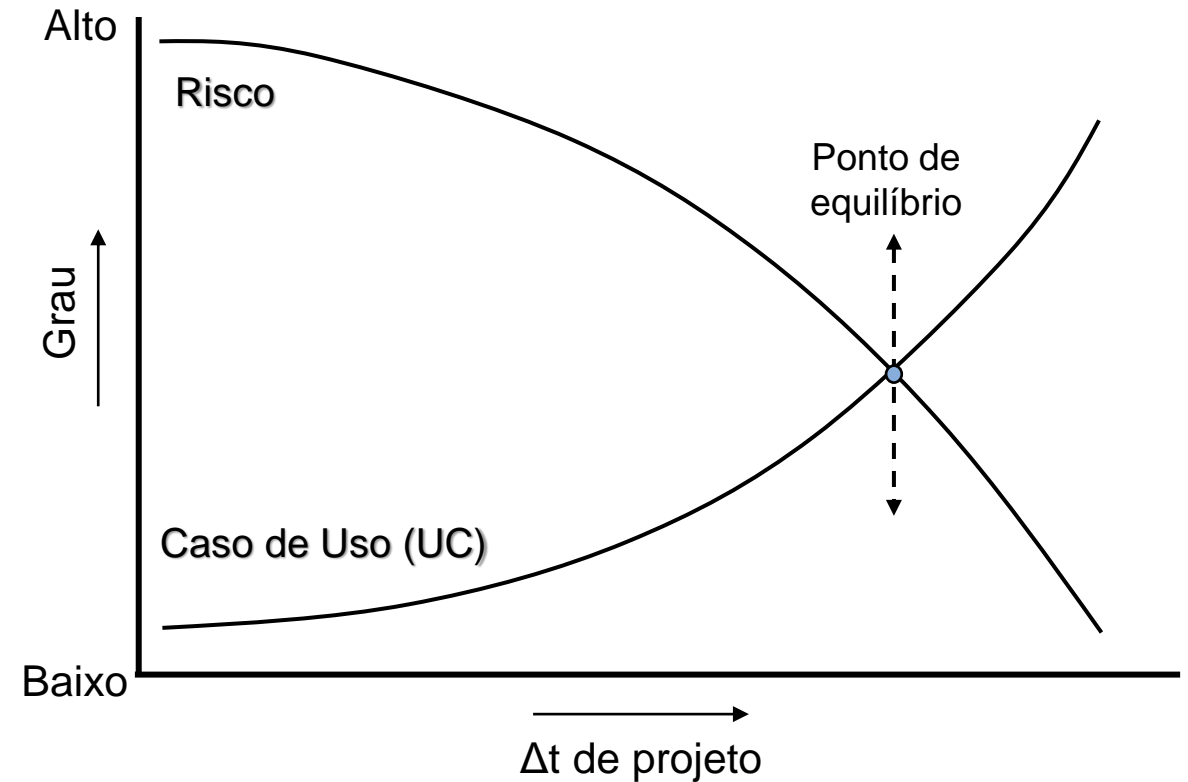


Figura 3. Fator de risco de um UC. (adaptado CAIXETA, 2020).



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.
 - Elaboração: As metas desta fase são desenvolver uma compreensão do problema dominante, estabelecer um *framework* da arquitetura, desenvolver o plano do projeto e identificar os maiores riscos. No final é apresentado um modelo de requisitos, que podem ser: um conjunto de UC, uma descrição da arquitetura e/ou um plano de desenvolvimento do *software*).
 - Construção: Envolve a programação e testes do sistema. Aqui 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: Implica na transferência do sistema de *software* para o ambiente real. [...]. Nesta fase, o sistema deve estar funcionando corretamente em seu ambiente operacional.

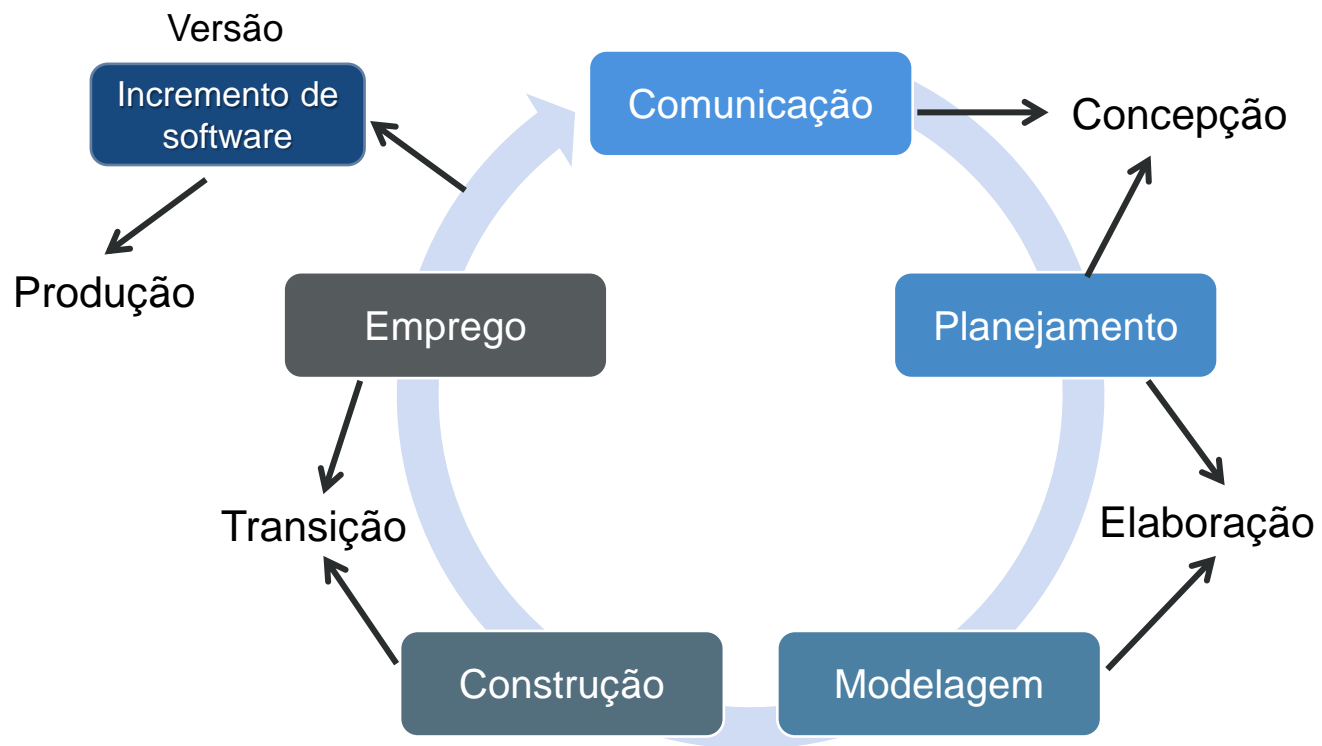


Figura 4. Fluxo genérico das fases RUP. (PRESSMAN, 2011).

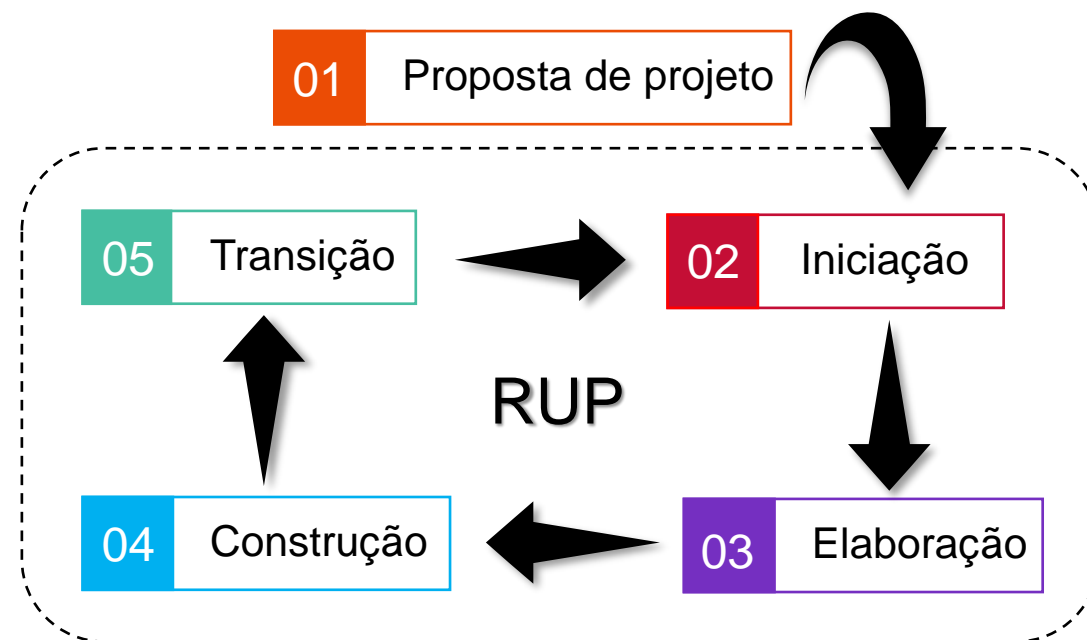


Figura 5. Fluxo comum das fases RUP.



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.
 2. Gerenciar os requisitos.
 3. Adotar arquiteturas baseadas em componentes.
 4. Modelar o *software* visualmente (visão estática e dinâmica com gráficos da UML).
 5. Verificar a qualidade do *software*.
 6. Controlar as mudanças do *software*.
- 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 combina três modelos de processo genéricos (cascata, incremental e orientada a reuso). (SOMMERVILLE, 2018).



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

Os centrais e os de apoio.

1.5. Os *workflows* e sua classificação

- 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 o/a RUP, os *workflows* são orientados em torno de modelos associados à UML, e.g., modelos de sequência, de objetos, de classe, etc.
- São classificados em centrais (seis) e de apoio (três).



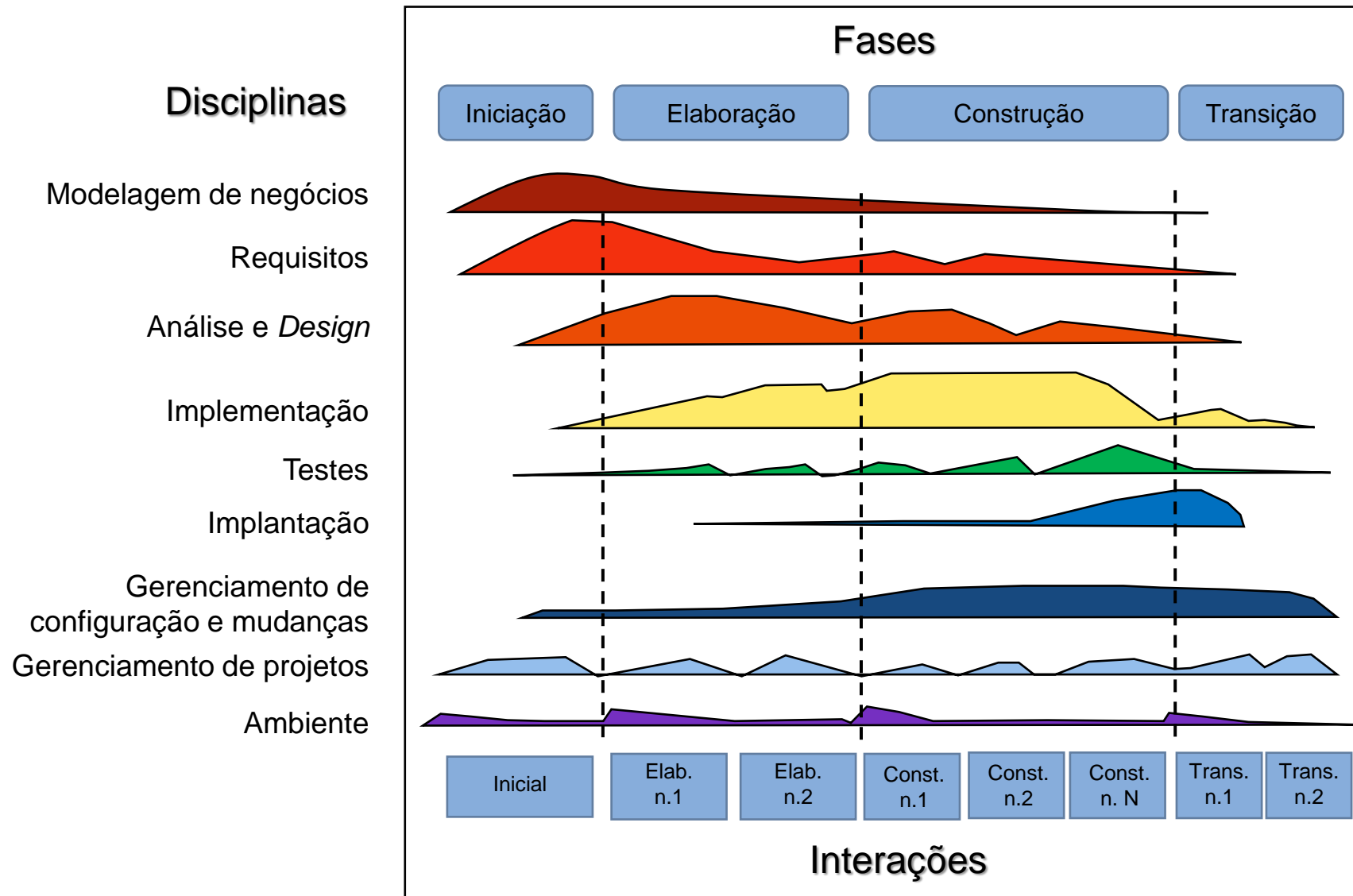


Figura 6. Ciclo de vida do RUP.

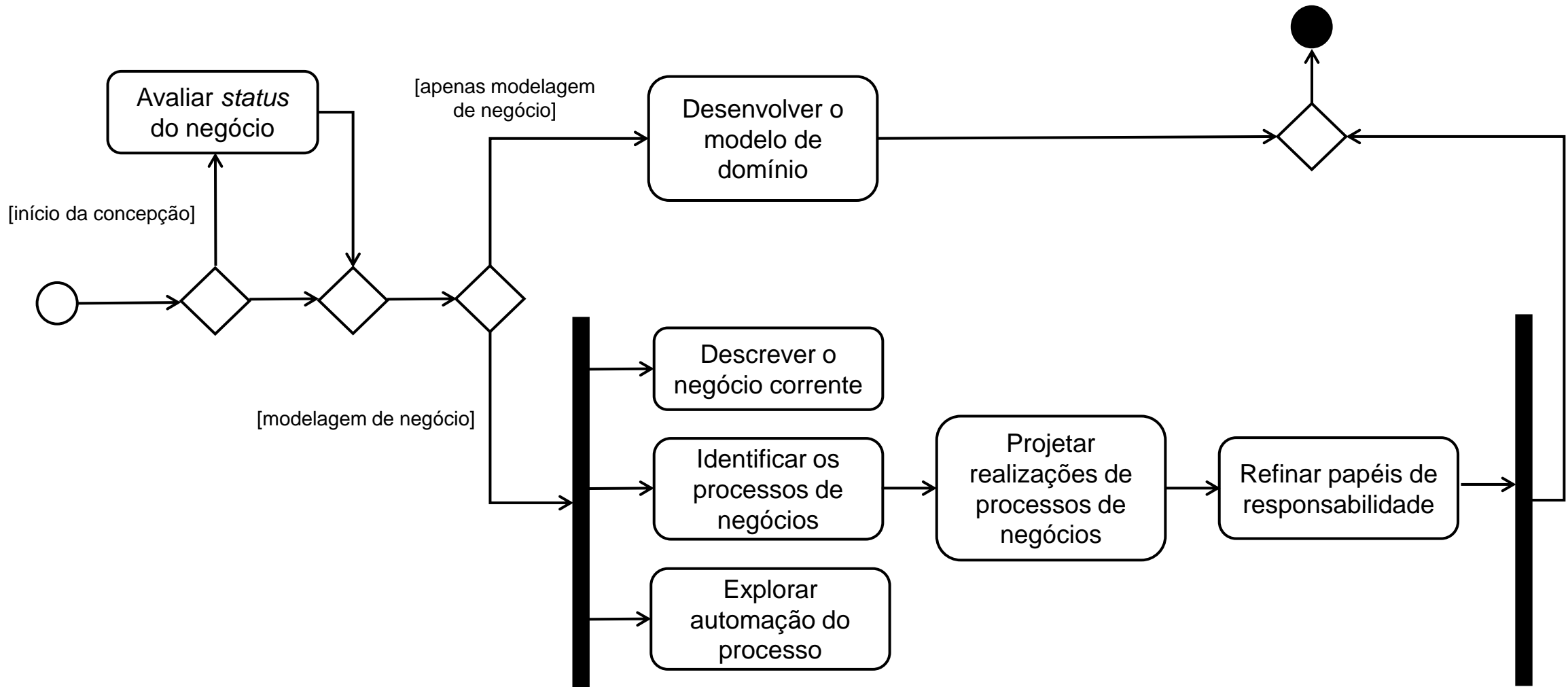


Figura 7. *Workflow central* da disciplina de *modelagem de negócio*. (WAZLAWICK, 2013).

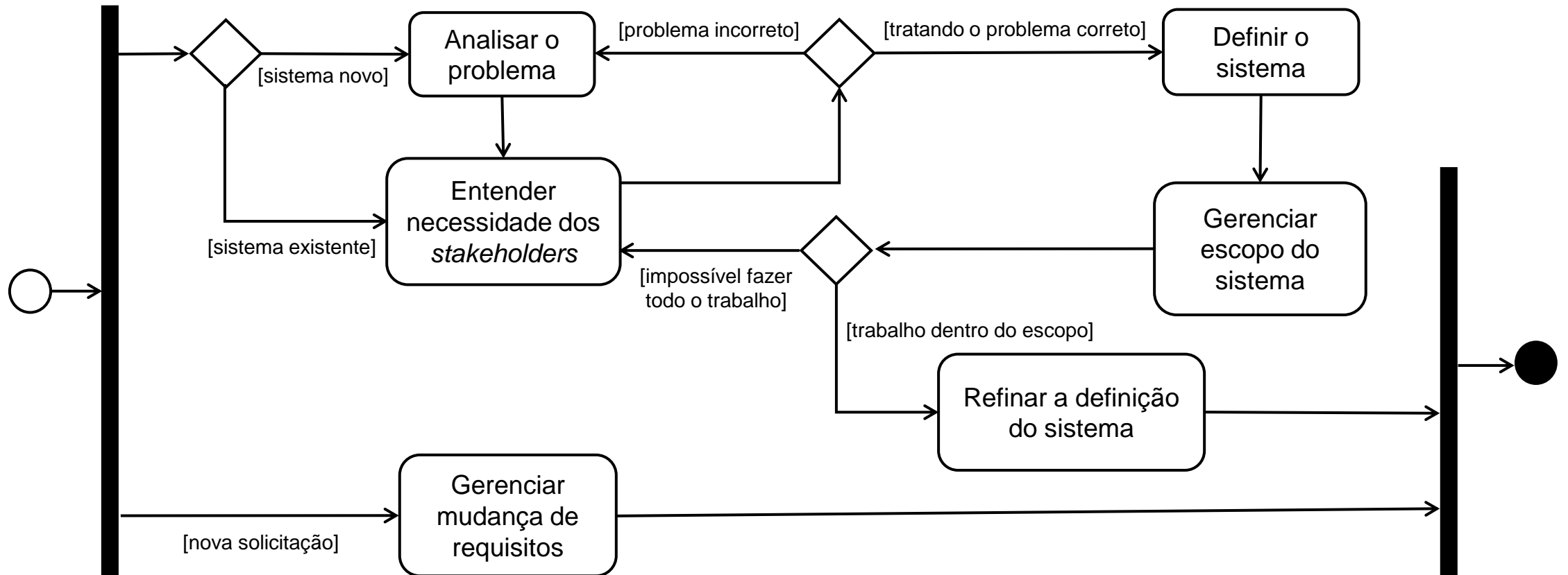


Figura 8. *Workflow central* da disciplina de *requisitos*. (WAZLAWICK, 2013).

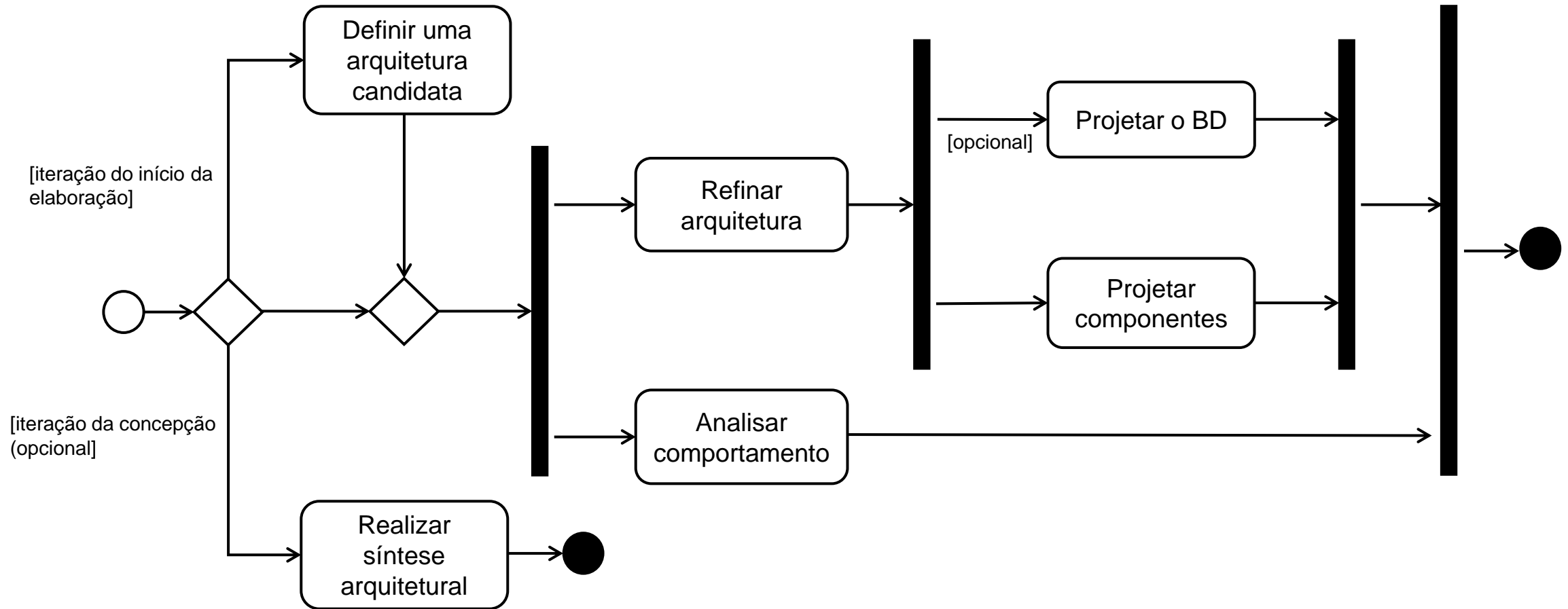


Figura 9. *Workflow central* da disciplina de *análise e projeto (design)*. (WAZLAWICK, 2013).

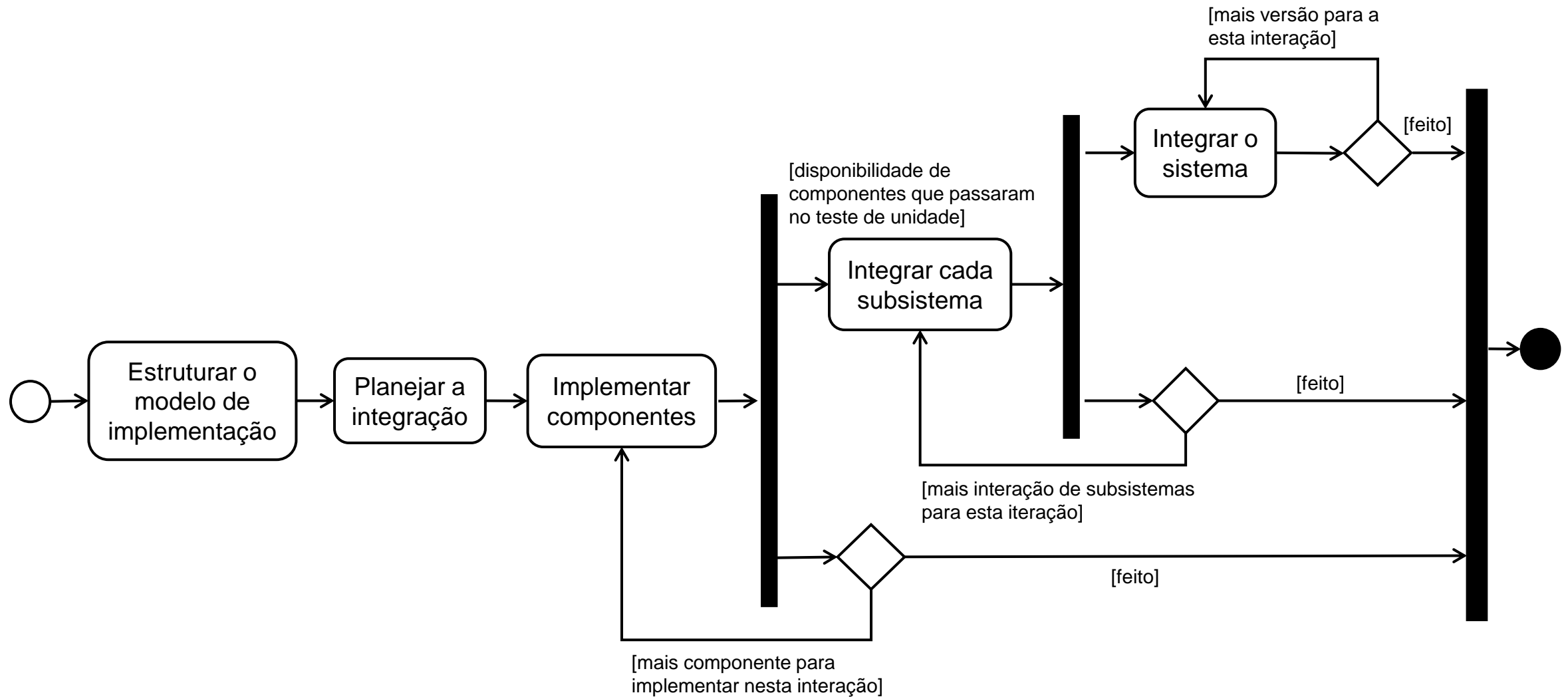


Figura 10. *Workflow central* da disciplina de *implementação*. (WAZLAWICK, 2013).

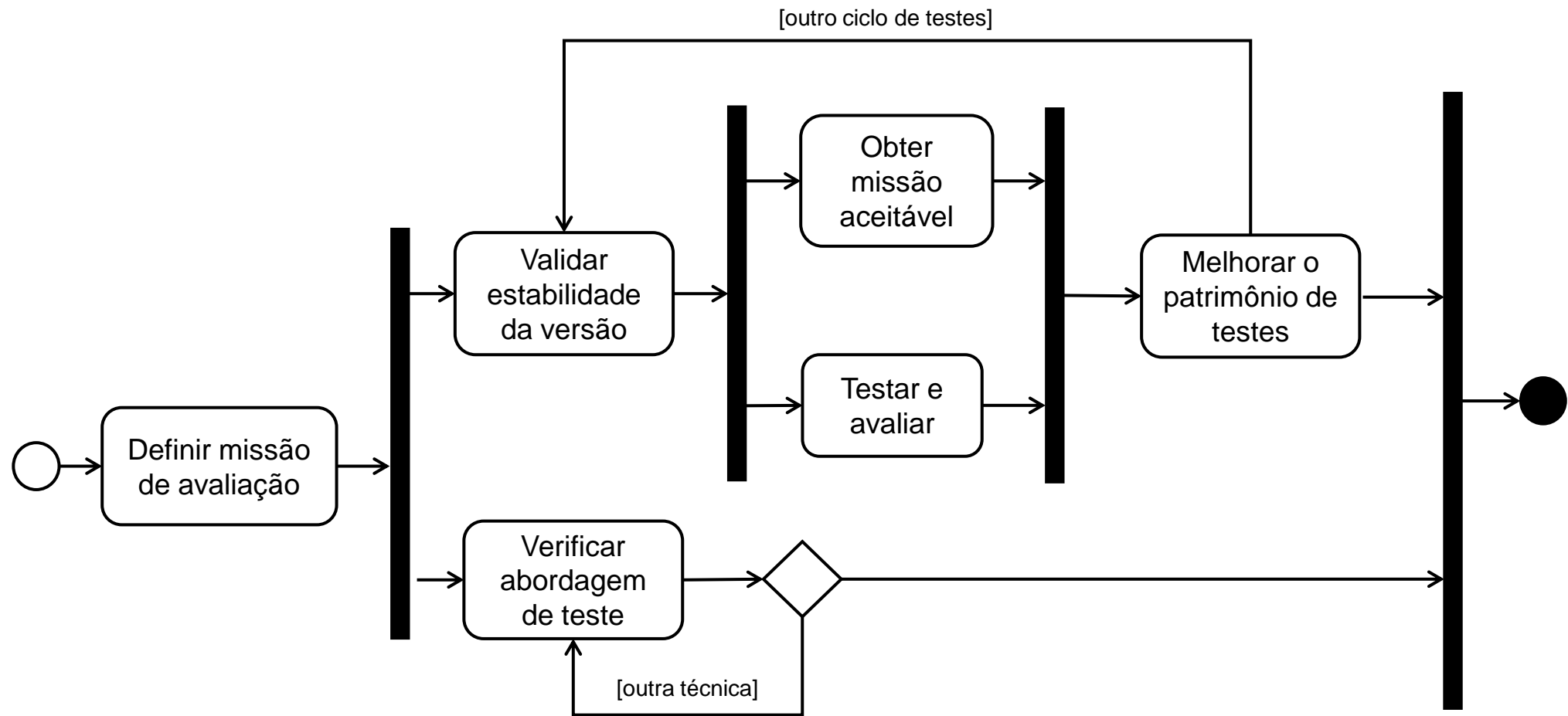


Figura 11. *Workflow central* da disciplina de *testes*. (WAZLAWICK, 2013).

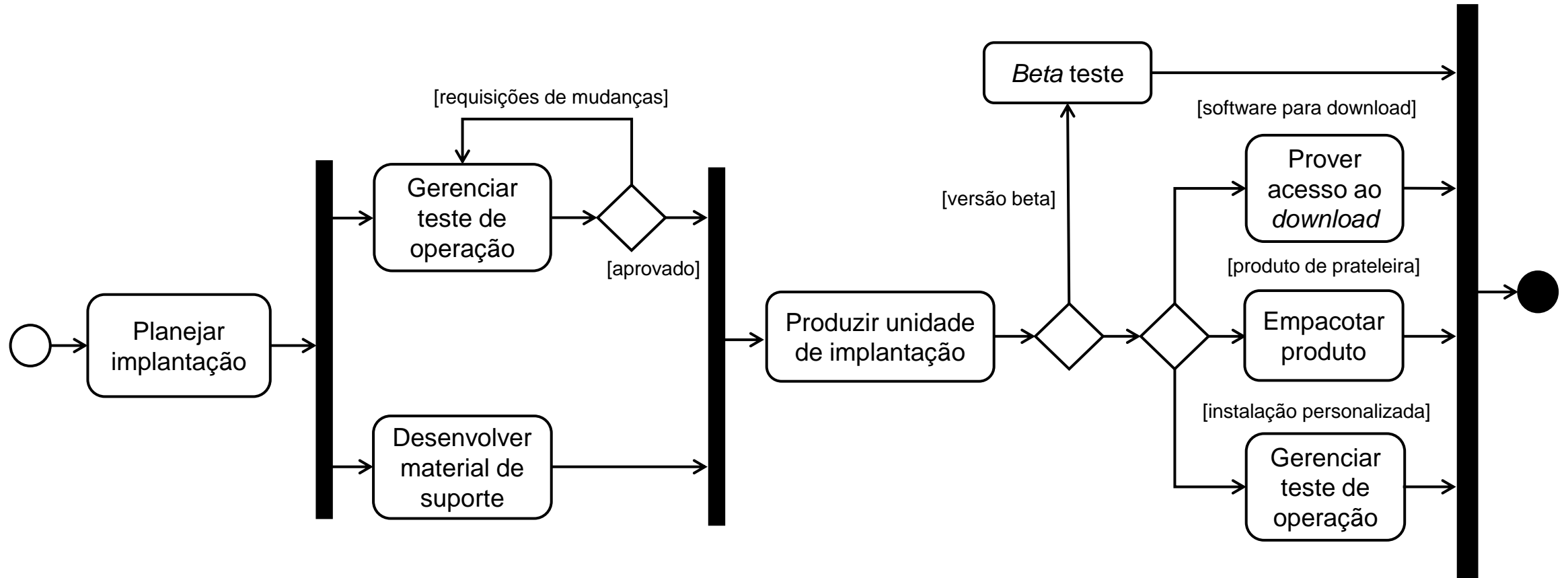


Figura 12. *Workflow central* da disciplina de *implantação*. (WAZLAWICK, 2013).

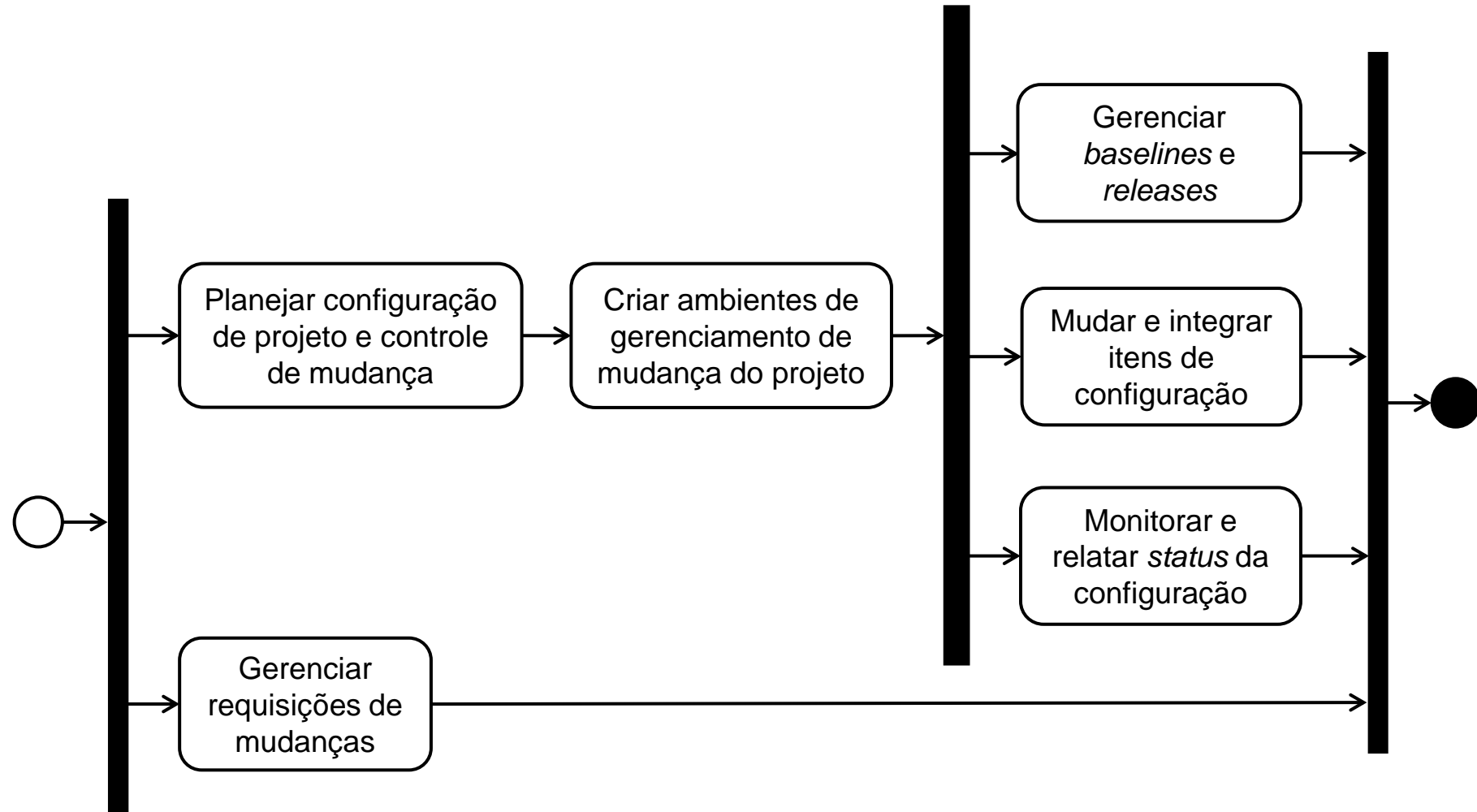


Figura 13. Workflow de apoio da disciplina de gerenciamento de configuração e mudanças. (WAZLAWICK, 2013).

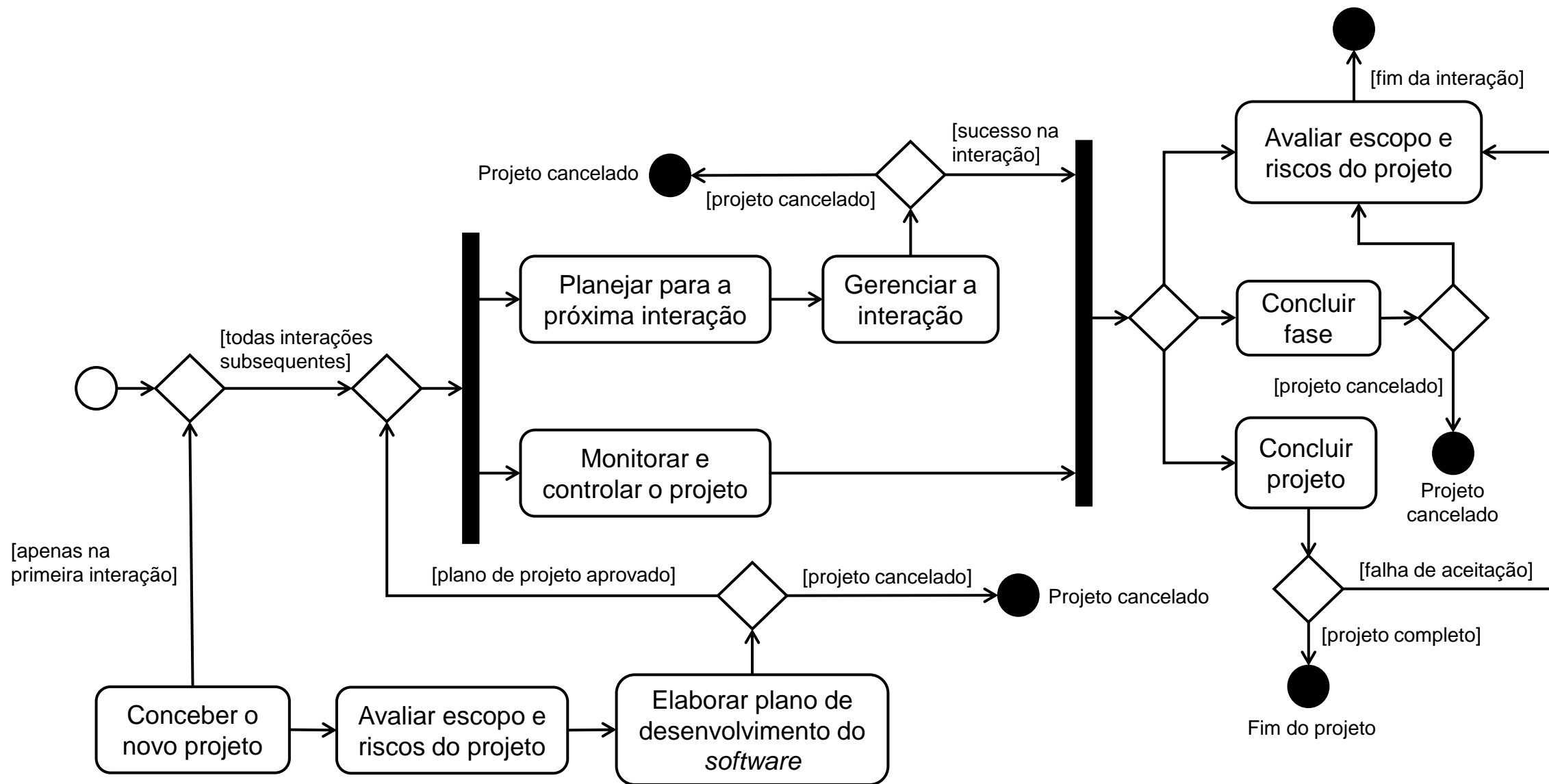


Figura 14. *Workflow de apoio da disciplina de gerenciamento de projeto.* (WAZLAWICK, 2013).

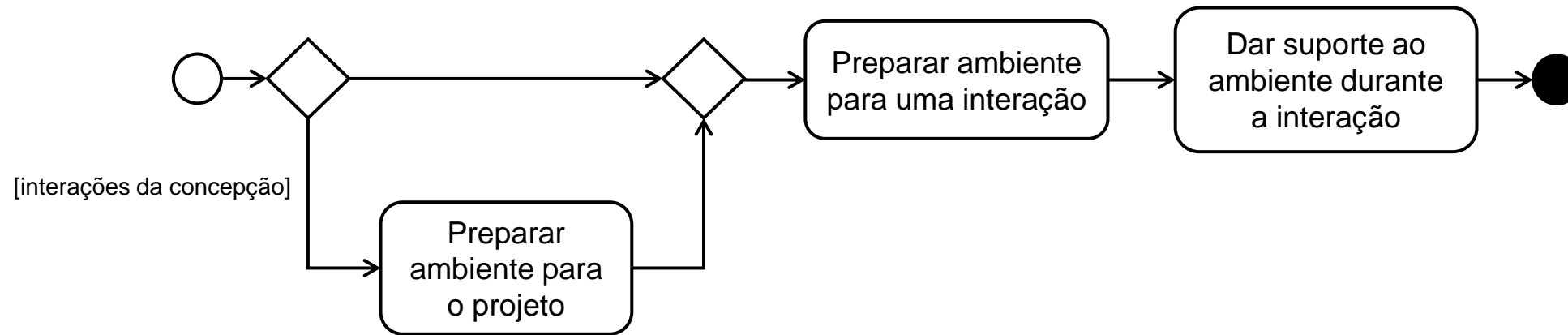


Figura 15. *Workflow de apoio da disciplina de ambiente.* (WAZLAWICK, 2013).

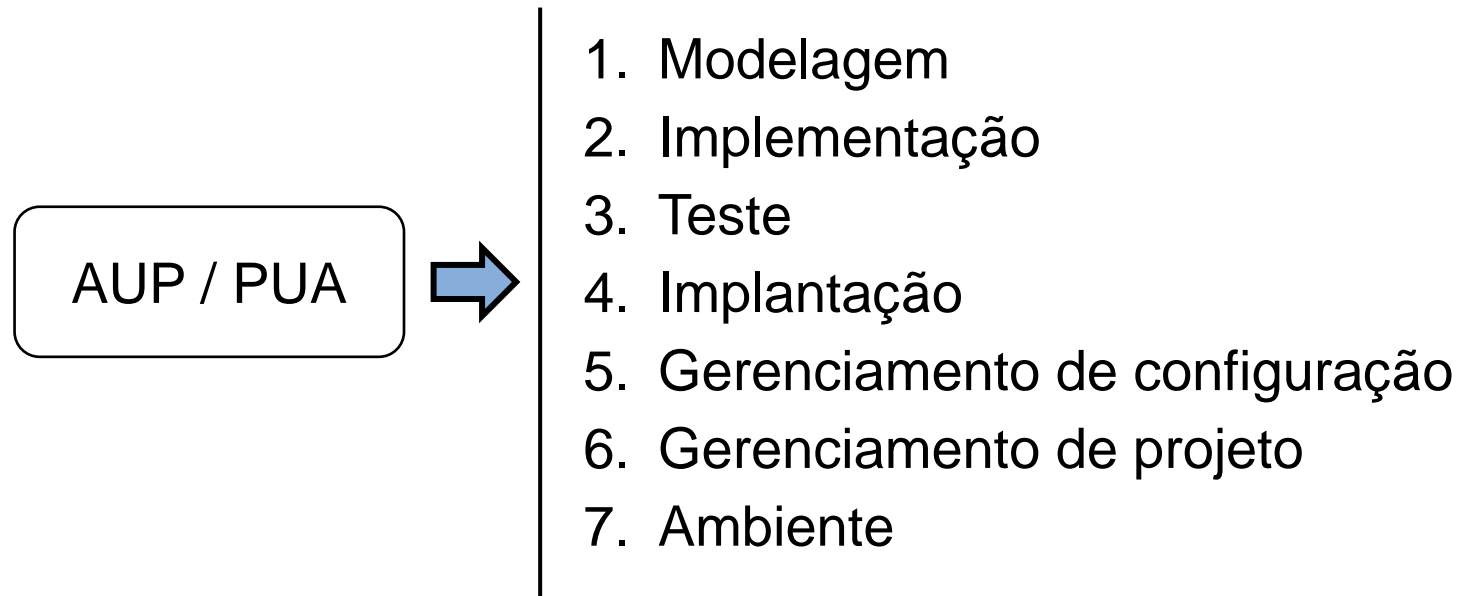


1.6. Alguns modelos

Versões de PU segundo Wazlawick (2013)

1.6.1. AUP - *Agile Unified Process*.

- 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*.

- Anteriormente conhecido como *Basic Unified Process* (BUP) ou *OpenUP/Basic*, o *Open Unified Process / Processo Unificado Aberto* (OpenUP) é uma implementação aberta (*open source*²) desenvolvida como parte do *Eclipse Process Framework* (EPF)³.
- 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.

2. Código aberto.

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

1.6.3. OUM – *Oracle Unified Method*.

- 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 instanciación 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*.

- 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, que:
 - a. São grandes o suficiente para comportar várias equipes de desenvolvimento trabalhando em paralelo.
 - b. Necessitam de desenvolvimento concorrente de *hardware* e *software*.
 - c. Cujas arquitetura é impactada por questões relativas à implantação.
 - d. Incluem a reengenharia de uma infraestrutura de T.I 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