Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Processo de definição de arquitetura de software: Exemplo de configuração e uso de processo de desenvolvimento de software

Mariana Borges de Sampaio

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia de Computação

Orientador
Prof. Dr. Fernando Albuquerque

Brasília
2022

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Processo de definição de arquitetura de software: Exemplo de configuração e uso de processo de desenvolvimento de software

Mariana Borges de Sampaio

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia de Computação

Prof. Dr. Fernando Albuquerque (Orientador)
CIC/UnB

Prof. Dr. Dr.

Prof. Dr. João Luiz Azevedo de Carvalho
Coordenador do Curso de Engenharia de Computação

Brasília, 25 de outubro de 2022

Dedicatória

Dedico este trabalho ...

Agradecimentos

Resumo

Palavras-chave: Engenharia de Software,Sistema de Software,Arquitetura de Software,Processo de Arquitetura de Software

Abstract

Keywords: Software Engineering,Software System,Software Architecture,Software Architecture Process

Sumário

Lista de Figuras

Lista de Abreviaturas e Siglas

ADM Architecture Development Method.

ATAM Architecture Tradeoff Analysis Method.

OpenUp Open Unified Process.

SAAM Software Architecture Analysis Method.

SACAM Software Architecture Comparison Analysis Method.

TOGAF The Open Group´s Architecture Framework.

Capítulo 1

Introdução

Este capítulo consiste de introdução ao trabalho desenvolvido. Entre os elementos deste capítulo, é possível destacar os seguintes: objetivos, motivações e justificativas, delimitação do trabalho, metodologia, estrutura do documento.

1.1 Objetivos

1.2 Motivações e justificativas

1.3 Delimitação do trabalho

1.4 Metodologia

1.5 Estrutura do documento

Capítulo 2

Arquitetura de software

Este capítulo aborda arquitetura de software. Entre os elementos deste capítulo, é possível destacar os seguintes: arquitetura de software,descrição de arquitetura de software,usos para descrição da arquitetura de software, ponto de vista e visão de arquitetura de software,práticas para descrição de arquitetura de software e arcabouços de arquitetura.

2.1 Arquitetura de software

Um sistema de software é um sistema onde software consiste de elemento de importância primária para as partes interessadas (*stakeholders*) no sistema.Entre os elementos de software, tem-se programas de computador.Programas de computador são compostos por instruções de computador e dados que capacitam hardware de computador a realizar funções computacionais e de controle com o propósito de atender necessidades de partes interessadas (*stakeholders*) no sistema. Em processo de desenvolvimento de sistema de software, os participantes podem assumir diversos papéis. Por exemplo,papéis de desenvolvedor ou usuário [1] [2].

Para desenvolver um sistema de software existe um processo que deve ser seguindo, sendo esse o processo de desenvolvimento de sistema de software.Nesse processo, existem diversas atividades que devem ser relaizadas.Por exemplo, atividades responsáveis por projeto de software (*software design*). Algumas dessas atividades são responsáveis por projeto de arquitetura de software (*architectural design*) [3]. Existem diversas definições para o termo arquitetura de software (*software architecture*). Segundo definição em [4], arquitetura de software corresponde à estrutura de sistema software ou às estruturas de sistema de software. Segundo essa definição, sistema de software pode ser composto por uma estrutura ou por mais de uma estrutura.Essas estruturas são compostas por elementos responsáveis por atribuições do sistema de software.Um mesmo elemento pode integrar mais de uma estrutura do sistema de software [4] [5].

Os elementos integrantes de sistema de software são responsáveis por atender necessidades de partes interessadas (*stakeholders*) no sistema.Essas necessidades podem variar entre sistemas de software.Os elementos integrantes de sistema de software estão associados a serviços prestados pelo sistema, características do sistema e funções do sistema.A arquitetura de software é composta por elementos integrantes do sistema de software e por relacionamentos entre os mesmos.A arquitetura de software é uma abstração do sistema de software,certos aspectos dos elementos integrantes do sistema são abstraídos [4] [5].

Segundo definição em [4],todo sistema de software possui uma arquitetura, pois todo sistema de software é composto por elementos e por relacionamentos entre elementos. Embora todo sistema de software tenha uma arquitetura de software, nem todo sistema de software tem informação acerca da mesma registrada em documento que facilite acesso a essa informação. Por diversos motivos, é relevante documentar arquitetura de software em ciclo de vida de sistema de software [4] [5].

Uma outra definição para o termo arquitetura de software é encontrada em [6].Segundo essa definição, arquitetura de software consiste de conjunto de estruturas relevantes ao entendimento de como os elementos integrantes de software estão relacionados e como ocorrem esses relacionamentos. Essas estruturas são compostas por elementos integrantes do software e por relacionamentos entre esses elementos [5] [6].

No contexto deste trabalho, é adotada a definição apresentada pelo [7].Nessa definição, arquitetura é organização fundamental de sistema incorporada em seus componentes, relacionamentos um com o outro, e com o ambiente, e os princípios que guiam seu projeto e evolução. A seguir, se encontra a definição no idioma original:

"The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution" [7].

Em desenvolvimento de sistema de software, o desenvolvimento de arquitetura de software se faz de acordo com o ambiente de desenvolvimento, são consideradas necessidades das partes interessadas (*stakeholders*) no sistema de software. Tendo esse ambiente sido definido, é estruturado o sistema de software. Esse compreende conjunto de componentes organizados de acordo com as suas funções no sistema de software. A fim de obter a conexão entre os componentes integrantes do sistema de software são estabelecidos relacionamentos entre os mesmos [5] [7].

2.2 Descrição de arquitetura de software

Ao longo do ciclo de vida de sistema de software, é relevante definir, documentar, realizar manutenção, melhorar e certificar a implementação de arquitetura de software.A descrição de arquitetura de software expressa sistema de software durante ciclo de vida do mesmo. Informação acerca de arquitetura pode ser registrada por meio de descrição da mesma.A descrição de arquitetura é produto de trabalho resultante de definição, documentação, manutenção, melhoria e certificação de implementação de arquitetura. A descrição de arquitetura pode consistir de coleção de artefatos, resultar da agregação de produtos de trabalho resultantes de atividades executadas ao longo de ciclo de vida de sistema de software [7] [8].

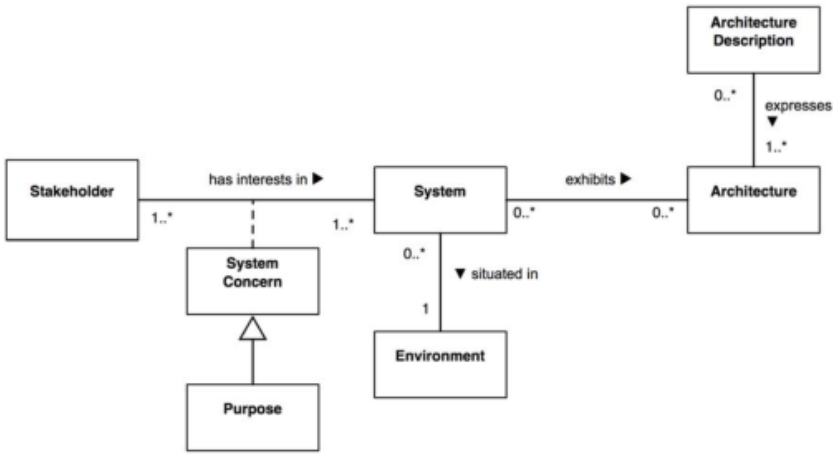

Figura 2.1: Contexto da descrição da arquitetura de software (Fonte: [8]).

O diagrama na Figura 2.1 representa o contexto no qual ocorre descrição de arquitetura de software(*architecture description*).Segundo esse diagrama, a parte interessada (*stakeholder*) possui interesse no sistema de software.Esse sistema trata de necessidades que são apontadas como o propósito do sistema de software.Este sistema está situado em um ambiente em que permite o seu desenvolvimento. Com isso, o sistema exibe a arquitetura que é expressa por meio da descrição de arquitetura [7] [8].

Descrição de arquitetura de software contém informação relevante ao desenvolvimento de software. A seguir, informação que pode estar presente em descrição de arquitetura de software: concepção fundamental de sistema de interesse em termos de seu propósito, qualidades de sistema (viabilidade, desempenho, segurança, usabilidade, interoperabilidade etc.), restrições, decisões e fundamentação; identificação de partes interessadas (*stakeholders*) no sistema de software, por exemplo, cliente ou usuário; identificação de demandas de partes interessadas no sistema de software; definições de pontos de vista (*viewpoints*) para

documentar procedimentos para criar, interpretar, analisar e avaliar dados arquitetônicos; uma ou mais visualizações do sistema, tendo em consideração que cada visão (*view*) de arquitetura aborda um ponto de vista, e que ponto de vista pode ser originado em parte interessada no sistema, que cada parte interessada pode ter demanda que deve ser abordada na descrição de arquitetura de software [9].

Cada visão da arquitetura de software aborda um ponto de vista, o ponto de vista está associado a uma parte interessada (*stakeholder*) no sistema. Cada parte interessada pode possuir uma demanda que deve ser abordada na descrição da arquitetura. Na descrição de arquitetura de software pode ser registrada informação por meio da qual seja possível fornecer fundamentação para decisões arquiteturais, com informação de rastreabilidade aos requisitos do sistema; estabelecer princípios para particionar o sistema em elementos integrantes do sistema (hardware, software, operações etc.) e elementos de projeto; registrar propriedades importantes e relacionamentos entre esses elementos de maneira consistente com a estrutura analítica do trabalho; demonstrar que requisitos arquitetonicamente significativos são atendidos e alocados para fornecer uma estrutura para especificação e refino do projeto. Por fim, a descrição de arquitetura de software deve fornecer informação que descreva a arquitetura de software por meio de seus componentes, apresentar a arquitetura a ser adotada na implementação do sistema de software. A descrição de arquitetura de software pode ser considerada uma especificação do sistema de software [9].

2.2.1 Usos para a descrição da arquitetura de software

Existem diversos usos para descrição de arquitetura de software. Por exemplo, os seguintes: base para projeto (*design*) e desenvolvimento de sistema; base para análise e avaliação de implementação de arquitetura; documentação de sistema; entrada para ferramenta automatizada; especificação de sistemas que compartilham características; comunicação durante desenvolvimento, produção, implantação, operação e manutenção de sistema; documentação de características e de projeto (*design*) de sistema; planejamento de transição de arquitetura legada para nova arquitetura; gerenciamento de configuração; suporte ao planejamento, definição de cronograma e definição de orçamento; estabelecimento de critérios para avaliar conformidade com arquitetura; embasamento de revisão, análise e avaliação de sistema; embasamento de análise e avaliação de arquiteturas alternativas; reuso de conhecimento acerca de arquitetura; treinamento e educação. Descrição de arquitetura de software pode promover a comunicação entre desenvolvedores do sistema de software e partes interessadas (*stakeholders*) no mesmo.Pode facilitar o entendimento de funcionalidades do sistema de software e o entendimento de estrutura do mesmo. Pode

facilitar a avaliação da capacidade da arquitetura de software atender às necessidades das partes interessadas (*stakeholders*) no sistema [8].

2.3 Ponto de vista e visão de arquitetura de software

Arquitetura de software pode ser projetada considerando-se visões (*views*) de arquitetura. A informação acerca de visões de arquitetura pode ser incluída em descrição de arquitetura de software. O acesso a essa informação pode, por exemplo, facilitar a identificação da presença ou da ausência de componentes necessários, facilitar a avaliação da capacidade da arquitetura atender demandas relacionadas ao sistema de software. Uma visão de arquitetura expressa a arquitetura segundo determinado ponto de vista (*viewpoint*).Portanto, uma visão de arquitetura representa a arquitetura segundo determinada perspectiva. Perspectiva essa que considera determinados interesses.Informação acerca de visão de arquitetura pode ser registrada em modelos de arquitetura. Cada modelo de arquitetura pode integrar mais de uma visão de arquitetura e pode ser construído levando-se em consideração convenções adequadas aos interesses enfocados pela visão [8].

Por sua vez, um ponto de vista (*viewpoint*) define audiência e propósito de visão de arquitetura. Um ponto de vista estabelece convenções para construção e uso de visão de arquitetura. As convenções definidas por um ponto de vista podem ser diversas. Por exemplo, podem ser definidas linguagens, notações, tipos de modelos, regras de projeto (*design*), métodos de modelagem e técnicas de análise. Considerando-se, por exemplo, requisitos relacionados ao sistema de software sendo desenvolvido, arquitetura de software pode ser projetada levando-se em consideração diferentes visões.As visões são construídas e usadas considerando-se convenções que se encontram definidas em pontos de vista.Em projeto de arquitetura de software, é relevante considerar possíveis pontos de vista e visões de arquitetura [7] [8].

2.4 Práticas para descrição de arquitetura de software

Práticas podem ser adotadas quando da descrição de arquitetura de software. Por exemplo, a norma *IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems* sugere prática para descrever arquiteturas de sistemas intensivos em software. A norma *IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems* relaciona elementos a serem incluídos em descrição de arquitetura e recomenda informação que deve estar presente em descrição de arquitetura. Por exemplo, a norma *IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems* recomenda o registro de informação

acerca de partes interessadas (*stakeholders*), acerca do sistema, acerca de pontos de vista (*viewpoint*) de arquitetura e acerca de visões (*views*) de arquitetura [7].

2.5 Arcabouços de arquitetura

Um arcabouço de arquitetura (*architecture framework*) tipicamente estabelece estruturas e práticas para criar, interpretar, analisar e usar descrições de arquitetura. Existem diversos arcabouços de arquitetura. Por exemplo, os seguintes: *The Open Group´s Architecture Framework (TOGAF)* e *The "4+1" View Model of Software Architecture*.

O *The Open Group´s Architecture Framework (TOGAF)* consiste de arcabouço para arquitetura organizacional (*enterprise architecture framework*). Esse arcabouço, que é desenvolvido e mantido pelo *The Open Group*, provê métodos e ferramentas para aceitação, produção, uso e manutenção de arquiteturas organizacionais. O *The Open Group´s Architecture Framework (TOGAF)* inclui o *Architecture Development Method (ADM)*, um método composto por fases e que se destina ao desenvolvimento e ao gerenciamento de ciclo de vida de arquitetura organizacional [8] [10].Essa metodologia pode garantir uma melhor eficiência do negócio a partir de uma estrutura de arquitetura de software consistente. Essa consistência obtida através da aplicação de métodos e padrões entre os arquitetos de software envolvidos. O *The Open Group´s Architecture Framework (TOGAF)* promove correção de erros, documentação da estrutura e remoção de conteúdos irrelevantes à arquitetura de software [10].

Por sua vez, o arcabouço (*framework*) *The "4+1" View Model of Software Architecture* sugere modelo para descrever arquitetura de software em visões (*views*). Esse arcabouço prescreve as seguintes visões: visão lógica, visão de processo, visão de desenvolvimento e visão física.A descrição da arquitetura de software pode ser organizada segundo essas visões e ilustrada por casos de usos ou por cenários. A visão lógica provê suporte primariamente a requisitos funcionais. Segundo essa visão, o sistema é decomposto em um conjunto de abstrações identificadas principalmente no domínio do problema. Caso seja adotado o paradigma de desenvolvimento orientado a objetos (*object oriented development*), essas abstrações são representadas por classes e objetos.A visão lógica pode ser descrita por diagramas de classes, diagramas de objetos e diagramas por meio dos quais sejam representados estados e transições. A visão de processo considera requisitos não funcionais, enfoca aspectos de projeto relacionados à concorrência, distribuição, integridade, tolerância a falhas, e como abstrações identificadas na visão lógica são mapeadas para a visão de processos. Nesse contexto, o termo processo designa agrupamento de tarefas (*tasks*). A arquitetura de processos pode ser descrita segundo diferentes níveis de abstração usando-se diagramas onde sejam representados elementos como processos, tarefas e mensagens.

A visão de desenvolvimento descreve organização estática do software no seu ambiente de desenvolvimento.A arquitetura é descrita por meio de diagramas onde podem ser representados módulos, subsistemas e seus relacionamentos. A visão física descreve mapeamento do software no hardware [11].

Por fim, tem-se que diversos *frameworks* estão disponíveis para uso no desenvolvimento de software. Eles têm grande relevância pois podem tornam o processo de compreender e detalhar a arquitetura de software mais simples para o arquiteto. Permitindo assim, uma padronização no conteúdo e no processo de arquitetura de software [8].

Capítulo 3

Processo de definição de arquitetura de software

Este capítulo aborda processo de definição de arquitetura de software. Entre os elementos deste capítulo, é possível destacar os seguintes: ciclo de vida de software, processos em ciclo de vida de software, processos técnicos em ciclo de vida de software e processo de definição de arquitetura de software.

3.1 Ciclo de vida de software

Existem diversos possíveis estágios em ciclo de vida de software. Cada estágio tem determinados propósitos e características. Estágios podem ser interdependentes, podem se sobrepor, podem ter diferentes durações etc. A seguir, são relacionados possíveis termos para designar estágios de ciclo de vida de software [12].

- Conceito;
- Desenvolvimento;
- Produção;
- Utilização;
- Suporte;
- Aposentadoria;

O conceito é o primeiro estágio do ciclo de vida do sistema de software, nesse estágio é realizada a contextualização, deve ser definido qual é o principal serviço que o sistema de software irá oferecer, quais serão os seus usuários finais, entre outras especificações. Após esse estágio tem-se o desenvolvimento do sistema de software e o estágio de produção.

Em seguida, o sistema de software passa a ser utilizado pelos usuários. O sistema de software pode ser alterado, passando para o estágio de suporte, momento no qual pode ser realizada modificação no sistema de software. Por fim, quando o sistema de software não tiver mais uso, ele pode ser passado para o estágio de aposentadoria, sendo assim encerrado o ciclo de vida do software [12].

3.2 Processos em ciclo de vida de software

Segundo [3],o processo inclui atividades que envolvem mudanças. Segundo [1] , diferentes processos podem apresentar diferentes pontos de início e diferentes pontos de término. As especificações de processos podem apresentar diferentes níveis de detalhamento de fluxos de trabalho.

Processos diversos podem ser adotados em ciclo de vida de software. Processo para construir protótipo com o propósito de ajudar a evitar más decisões sobre requisitos de software, processo para levantamento e especificação de requisitos de software etc. Processos podem estruturar o trabalho de diversos modos. Por exemplo, o processo pode estruturar o trabalho para desenvolvimento de software por meio de entregas iterativas, de modo que, conforme mudanças ocorram, possam ser integradas ao sistema.

No contexto deste trabalho, é adotada a definição apresentada em [12]. Segundo essa definição, processo é um conjunto integrado de atividades que transforma entradas em saídas desejadas. Segundo [12], cada processo possui objetivo final a ser alcançado, possui propósito. Cada processo provê resultado distinto. Cada resultado tem como função fornecer benefício que motive a seleção e o uso do processo. A partir do momento em que é obtido o resultado esperado, o processo cumpriu o seu propósito.

3.3 Processos técnicos em ciclo de vida de software

O sistema de software deve atender necessidades de partes interessadas (*stakeholders*) no mesmo. Por exemplo, necessidades de usuários do sistema de software. Necessidades podem ser atendidas por meio de processos em ciclo de vida de software. Por exemplo, por meio de processos técnicos. Por meio de processos técnicos, necessidades de partes interessadas são transformadas em produto. Os processos técnicos podem ser implementados para criar ou usar sistema de software, e podem ser aplicados em diferentes estágios de ciclo de vida de software. A seguir, são relacionados nomes de possíveis processos técnicos [13].

- Processo de análise de negócios ou missão;
- Processo de definição de necessidades e requisitos das partes interessadas;

- Processo de definição de requisitos de sistema de software;
- Processo de definição de arquitetura;
- Processo de definição de projeto;
- Processo de análise;
- Processo de implementação;
- Processo de integração;
- Processo de verificação;
- Processo de transição;
- Processo de validação;
- Processo de operação;
- Processo de manutenção;
- Processo de descarte.

3.4 Processo de definição de arquitetura de software

Ao longo de ciclo de vida de software, a definição de arquitetura de software pode ser realizada por meio de processo. Dentre os processos técnicos em ciclo de vida de software, tem-se o processo de definição de arquitetura (*architecture definition process*) composto pelas seguintes atividades: preparar para a definição da arquitetura; desenvolver pontos de vista de arquitetura; desenvolver modelos e visões de arquiteturas candidatas; relacionar a arquitetura com o projeto (*design*); avaliar arquiteturas candidatas; gerenciar a arquitetura selecionada. O processo de definição de arquitetura tem como objetivo gerar opções de arquitetura que possam ser implementadas no desenvolvimento do sistema de software. As opções de arquitetura geradas devem englobar demandas solicitadas pelas partes interessadas (*stakeholders*) e funcionalidades a serem providas pelo sistema de software [13].

As opções de arquitetura geradas são analisadas levando-se em consideração pontos de vista (*viewpoints*) e visões (*views*) de arquitetura. A fim de verificar se as opções de arquitetura que foram propostas são adequadas, são analisadas interações entre processo de definição de arquitetura de software, processo de análise de negócios ou missão, processo de definição de requisitos de sistema de software, processo de definição de projeto e processo de definição de necessidades e requisitos de partes interessadas (*stakeholders*).

Dessa forma é possível analisar formas de suprir demandas quanto ao software, e encontrar assim a melhor solução [13].

A seguir, são relacionados resultados desejáveis de processo de definição de arquitetura de software:demandas identificadas por partes interessadas no sistema de software são abordadas pela arquitetura escolhida;pontos de vista da arquitetura são desenvolvidos;são definidos contexto,limites e interfaces externas do sistema de software;são desenvolvidas visões e modelos do sistema de software; conceitos, propriedades,características,comportamentos, funções ou restrições significativas para decisões de arquitetura são alocados a entidades arquiteturais; elementos do sistema e suas interfaces são identificadas; opções de arquitetura são avaliadas; base arquitetônica para processos ao longo de ciclo de vida é alcançada; alinhamento da arquitetura com requisitos e características de projeto é alcançado; sistemas ou serviços necessários para definição da arquitetura estão disponíveis; é definida rastreabilidade entre elementos da arquitetura e requisitos das partes interessadas (*stakeholders*) no sistema [13].

Capítulo 4

Métodos para avaliação de arquitetura de software

Este capítulo aborda avaliação de arquitetura de software, particularmente métodos para avaliação de arquitetura de software. Entre os elementos que integram este capítulo, é possível destacar os seguintes: avaliação de arquitetura de software, métodos para avaliação de arquitetura de software, *Software Architecture Comparison Analysis Method (SACAM)*, *Architecture Tradeoff Analysis Method (ATAM)* e *Software Architecture Analysis Method (SAAM).*

4.1 Avaliação de arquitetura de software

No contexto do processo de definição de arquitetura, tem-se atividade para avaliar arquiteturas candidatas.Essa atividade é composta pelas seguintes tarefas [13] :

- Avaliar cada arquitetura candidata em relação às restrições e aos requisitos;
- Avaliar cada arquitetura candidata em relação às demandas das partes interessadas (*stakeholders*) no sistema de software usando critérios de avaliação;
- Selecionar as arquiteturas preferidas e capturar decisões e justificativas para eleger as candidatas;
- Estabelecer a linha de base de arquitetura (*architecture baseline*) da arquitetura selecionada, essa linha de base deve conter modelos, visualizações e demais especificações referentes à arquitetura de software selecionada.

4.2 Métodos para avaliação de arquitetura de software

Para avaliar arquitetura de software, pode ser adotado o método proposto para esse propósito.Por meio de método para avaliação de arquitetura de software,procura-se eliminar arquiteturas que não sejam adequadas ao sistema de software sendo desenvolvido.A seguir,são descritos elemtnos dos seguintes [14]:*Software Architecture Comparison Analysis Method (SACAM)*,*Architecture Tradeoff Analysis Method (ATAM)*,*Software Architecture Analysis Method (SAAM)*. Serão descritos métodos que são embasados em cenários (*scenario-based software architecture analysis methods*).

4.3 Software Architecture Comparison Analysis Method

O *Software Architecture Comparison Analysis Method (SACAM)* tem como objetivo fornecer justificativas para escolha de determinada arquitetura de software.Para isso, são comparadas arquiteturas candidatas sendo para o sistema de software.Para essa comparação ser realizada, as seguintes atividades são executadas [15]:

- Extrair visões de arquitetura;
- Compilar critérios de avaliação das arquiteturas candidatas.

Para alcançar os objetivos necessários, esse método possui algumas etapas que devem ser seguidas a fim de eleger a arquitetura de software mais adequada.Dentre as etapas que devem ser seguidas, as seguintes [15]:

- Preparação;
- Agrupamento de critérios;
- Determinação de diretrizes de extração;
- Exibição e extração de indicadores;
- Pontuação;
- Resumo.

Na etapa Preparação (*Preparation*) são identificados os objetivos de negócios que são relevantes para a avaliação entre as arquiteturas candidatas e são examinadas documentações disponíveis das arquiteturas candidatas.Na etapa Agrupamento de critérios (*Criteria Collation*) são agrupados os critérios de classificação de acordo com os objetivos de negócio.Na etapa Determinação de diretrizes de extração (*Determination of Extraction*

Directives)são determinadas as visões arquitetônicas, táticas, estilos e padrões que são necessários na construção dos cenários.Na etapa Exibição e extração de indicadores (*View and Indicator Extraction*) são extraídas as visualizações de arquitetura para cada arquitetura candidata de acordo com as diretrizes que foram definidas na etapa de determinação de diretrizes de extração.Também são analisados os indicadores que permitem que os cenários sejam projetados seguindo os critérios que foram estabelecidos na etapa de agrupamento de critérios.E por fim, nessa etapa também devem ser implementadas técnicas de recuperação da arquitetura de software.Na etapa de Pontuação (*Scoring*) a arquitetura candidata é avaliada e recebe uma pontuação. Essa pontuação informa se a arquitetura candidata consegue prover suporte aos critérios que foram estabelecidos para a arquitetura de software.Por fim, na etapa de Resumo (*Summary*) é realizado um resumo que contém os resultados e análises das arquiteturas candidatas que foram avaliadas. No SACAM é obtida pontuação de cada arquitetura candidata e justificativas para pontuação de cada arquitetura candidata. Além disso, são gerados artefatos que documentam a arquitetura. A fim de obter esses resultados existem técnicas que o arquiteto de software pode adotar. Dentre essas técnicas, tem-se as seguintes [15]:

- Geração de cenários;
- Táticas;
- Métricas;
- Padrões de documentação arquitetônica;
- Reconstrução da arquitetura.

A técnica de Geração de cenário (*Scenario Generation*) permite capturar atributos de qualidade estabelecidos de acordo com o objetivo do sistema de software e refinar esses atributos em cenários de atributos de qualidade. A técnica Táticas (*Tactics*) tem como finalidade obter qualidades particulares solicitas. O *Software Architecture Comparison Analysis Method (SACAM)* utiliza as táticas para avaliar se as visões extraídas suportam o critério sendo avaliado. A técnica Métricas (*Metrics*) permite realizar análises quantitativas que fornecem indicadores úteis de complexidade geral e aponta onde mudança na arquitetura de software pode ser mais difícil ou mais provável. Métricas podem ser utilizadas no nível de código ou em nível de projeto (*design*) detalhado.Na técnica Padrões de documentação arquitetônica (*Architectural Documentation Standards*),o SACAM requer a disponibilidade de documentação arquitetônica para realizar a análise de critérios e comparação entre arquiteturas candidatas. Por fim, na técnica Reconstrução de arquitetura (*Architecture Reconstruction*),caso a documentação da arquitetura esteja desatualizada ou indisponível a arquitetura precisa ser reconstruída.

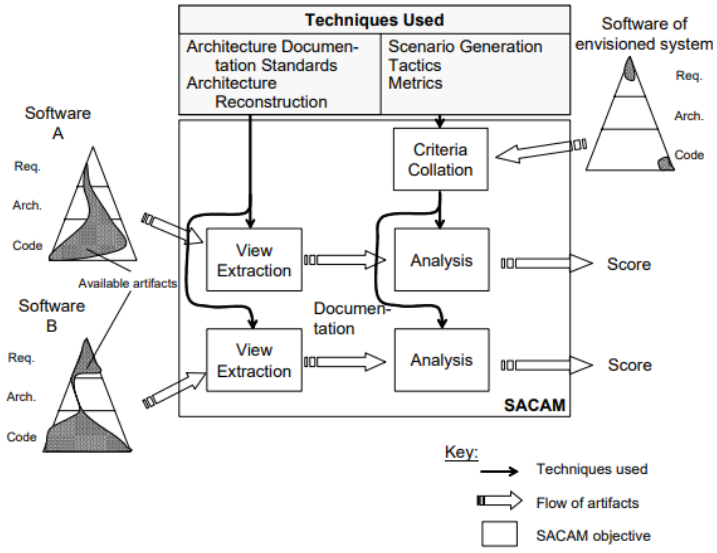

Figura 4.1: Técnicas de arquitetura utilizadas pelo SACAM (Fonte: [15]).

Na Figura 4.1 são representadas técnicas utilizadas pelo SACAM.A partir dessas técnicas são gerados artefatos necessários para avaliar arquiteturas candidatas e atribuir pontuações a cada arquitetura. Dessa forma é possível comparar arquiteturas candidatas e eleger a mais adequada.O SACAM auxilia na seleção da arquitetura de software fornecendo resultados da análise de arquitetura de software. Para fornecer esses resultados, o SACAM compara arquiteturas com base em critérios estabelecidos de acordo com objetivos de negócio da empresa interessada na arquitetura de software [15].

4.4 Architecture Tradeoff Analysis Method

O método *Architecture Tradeoff Analysis Method (ATAM)* visa entender consequências de decisões arquiteturais tomadas, tendo como ponto de partida os requisitos de atributos de qualidade do sistema de software.Sendo assim, é possível determinar se objetivos do sistema de software poderão ser atendidos pela arquitetura de software escolhida.Esse método orienta os usuários interessados a procurar pontos problemáticos e soluções para esses pontos na arquitetura de software.Esse método pode também ser utilizado para analisar sistemas legados, sistemas antigos que ainda continuam em operação. O ATAM tem as seguintes características [16].

- Pode ser implementado no início do ciclo de vida de desenvolvimento de software;
- Pode produzir análises compatíveis com o nível de detalhamento em relação a especificação do projeto arquitetônico.

O ATAM é um método para analisar se em modelo arquitetônico estão presentes os atributos de qualidade demandados pelas partes interessadas (*stakeholders*) no sistema de software. O método é composto por passos (*steps*) a seguir descritos. No passo para apresentar o ATAM (*Present the ATAM*), o método é descrito para as partes interessadas (*stakeholders*) no sistema de software. No passo para apresentar impulsionadores dos negócios atuais (*Present business drivers*), o gerente de projeto descreve objetivos do negócio e principais impulsionadores arquitetônicos. No passo para apresentar arquitetura (*Present architecture*), o arquiteto de software descreve a arquitetura candidata e como devem ser abordados os objetivos do negócio. No passo para identificar abordagens arquitetônicas (*Identify architectural approaches*), as abordagens devem ser identificadas, mas não analisadas. No passo para gerar árvore de utilidades de atributos de qualidade (*Generate quality attribute utility tree*), deve-se elicitar fatores de qualidade que compõem as características do sistema de software, por exemplo, desempenho e segurança. Esses fatores devem ser especificados de acordo com o nível de cenário que será analisado. Também devem conter respostas geradas por cada estímulo ao sistema de software. Devem ser priorizadas as utilidades de atributos de qualidade nesse passo. No passo para analisar abordagens arquitetônicas (*Analyze architectural approaches*), de acordo com os fatores de qualidade priorizados no passo anterior, são analisadas abordagens arquitetônicas que possuem esses fatores. Nesse passo são identificados riscos arquitetônicos, pontos frágeis e pontos onde podem ocorrer mudanças na arquitetura de software. No passo para realizar brainstorming e priorizar cenários (*Brainstorm and prioritize scenarios*), de acordo com os cenários analisados no passo anterior, são priorizados os cenários mais bem votados nesse passo. Essa votação deve incluir as partes interessadas (*stakeholders*) no sistema de software. No passo para analisar abordagens arquitetônicas (*Analyze architectural approaches*), é realizada análise das visões arquitetônicas aprovadas em passos anteriores. Os cenários avaliados são considerados para teste. Esses cenários podem revelar a necessidade de novas estruturas na arquitetura, novos riscos ou outros pontos na arquitetura de software. No passo para apresentar resultados (*Present results*), nele são apresentadas as descobertas realizadas para as partes interessadas (*stakehoders*) no sistema de software. Deve ser feito um relatório com informações obtidas e estratégias propostas para a arquitetura de software.Por fim, é relevante destacar que o ATAM depende da comunicação entre as partes interessadas (*stakeholders*) no sistema de software [16].

4.5 Software Architecture Analysis Method

O *Software Architecture Analysis Method (SAAM)* é um método embasado em cenários.Por meio de cenário é possível ilustrar atividade que o sistema de software deve suportar ou

mudança que pode ocorrer durante o uso do sistema de software. O SAAM visa analisar a arquitetura de software. O SAAM guia o arquiteto de software na identificação de pontos problemáticos na arquitetura. Para exemplificar possíveis pontos problemáticos, tem-se ponto de conflito entre requisitos ou ponto em que arquitetura candidata pode possuir demanda não implementada ou incompleta, demanda essa solicitada por parte interessada (*stakeholder*) no sistema de software. O SAAM permite comparar arquiteturas candidatas e provê suporte à escolha de arquitetura que melhor se alinhe com o sistema de software [14] [17].

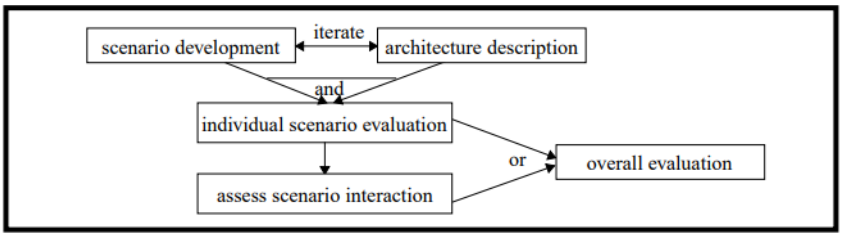

Figura 4.2: Etapas integrantes do SAAM (Fonte: [17]).

Na Figura 4.2 são representadas as seguintes etapas do SAAM [17]:

- Descrição de arquitetura;
- Desenvolvimento de cenário;
- Avaliação individual de cenário;
- Avaliação de interação de cenário;
- Avaliação geral.

Por meio da avaliação de cenários, o SAAM permite a comparação entre arquiteturas candidatas a fim de eleger a arquitetura de software mais adequada para o sistema de software. O SAAM possibilita que sejam integrados os diversos interesses das partes interessadas (*stakeholders*) no sistema de software e estabelece um cenário que entregue as demandas solicitadas pelas partes interessadas (*stakeholders*) através de uma arquitetura de software para o sistema de software [14].

4.5.1 Descrição de arquitetura

A descrição da arquitetura candidata corresponde à primeira etapa do método SAAM. Nessa etapa, a arquitetura candidata deve ser descrita usando-se notação compreendida pelas partes interessadas (*stakeholders*) no sistema de software. A descrição deve indicar componentes integrantes da arquitetura candidata e os seus relacionamentos [17].

4.5.2 Desenvolvimento de cenário

Na segunda etapa tem-se o desenvolvimento de cenários. Nesse caso, devem ser desenvolvidas as tarefas que descrevam atividades que o sistema deve suportar e mudanças que serão feitas no sistema ao longo do tempo. No momento em que são desenvolvidos os cenários, deve-se dar relevância à captura de necessidades a serem atendidadas pelo sistema de software. O cenário deve representar as tarefas relevantes do sistema de software para diferentes usuários [17].

4.5.3 Avaliação individual de cenário

Na terceira etapa avalia-se os cenários desenvolvidos na etapa anterior. Inicialmente deve ser avaliado se com a arquitetura candidata é possível desenvolver o cenário proposto ou se é necessário alterar a arquitetura candidata. Caso seja necessária alteração, o cenário em questão é chamado de cenário indireto. Sendo necessária alteração na arquitetura, devem ser listadas as alterações necessárias e deve ser estimado o custo para realizar a alteração. Visto que uma alteração na arquitetura implica que novo componente ou novo relacionamento seja introduzido na arquitetura candidata. Ao final dessa avaliação, deve ser feito um quadro-resumo contendo os cenários (diretos e indiretos). Para cada cenário indireto, é necessário descrever o impacto que a modificação causa no sistema de software. Esse quadro-resumo permite comparar arquiteturas candidatas, visto que é determinado se o cenário precisa de modificações ou não [17].

4.5.4 Avaliação de interação de cenário

Na quarta etapa tem-se a avaliação de interações entre cenários. Cenários indiretos podem requerer alterações em componentes ou em relacionamentos. Para determinar interações entre cenários, identifica-se cenários que afetam conjunto comum de componentes. Identificar interações é relevante pois identifica até que ponto a arquitetura candidata suporta cenários sendo estabelecidos para o sistema de software [17].

4.5.5 Avaliação geral

Na quinta etapa, deve ser realizada avaliação geral da arquitetura candidata. É necessário analisar de forma ponderada cada cenário e as interações do cenário em termos de sua relevância na arquitetura. Através dessa ponderação deve ser possível fazer uma classificação geral dos cenários. Esse processo deve envolver partes interessadas (*stakeholders*) no sistema. Com essa ponderação, é refletida a relevância relativa dos fatores de qualidade que os cenários manifestam na arquitetura [17].

Capítulo 5

Exemplo de configuração de processo de desenvolvimento

Este capítulo aborda configuração de processo de desenvolvimento.Entre os elementos que integram este capítulo,é possível destacar os seguintes:*Open Unified Process (OpenUp)*,elementos do OpenUP relacionados à arquitetura de software e configuração de processo de desenvolvimento de software.

5.1 Open Unified Process

O processo de desenvolvimento de software, a ser configurado no contexto deste trabalho,é denominado *Open Unified Process (OpenUp)*.O OpenUp sugere que o desenvolvimento de software seja realizado em incrementos, dessa forma unidades pequenas de trabalho podem produzir trabalho contínuo que demonstre métrica que permita mensurar o progresso do projeto. Sendo assim, é possível acompanhar e documentar o desenvolvimento do software de forma incremental, artefatos são construídos na medida que progride o desenvolvimento do software [18].

Essa forma de trabalho promove realimentação (*feedback*) que permite a adequação do projeto quando necessário, além de prover suporte à tomada de decisão durante o desenvolvimento de software. A partir da interação entre as equipes é possível determinar para qual fase o projeto seguir, visto que a partir da interação entre as equipes tem-se uma estimativa do quanto o projeto teve andamento no período analisado [18]. A Figura 5.1 ilustra o ciclo de vida de projeto no OpenUP.

No processo OpenUp, o ciclo de vida de projeto é composto pelas seguintes fases: iniciação (*Inception*), elaboração (*Elaboration*), construção (*Construction*) e transição (*Transition*). A fase de iniciação tem o propósito de alcançar concordância entre os interessados (*stakeholders*) sobre os objetivos do ciclo de vida do projeto. A fase de

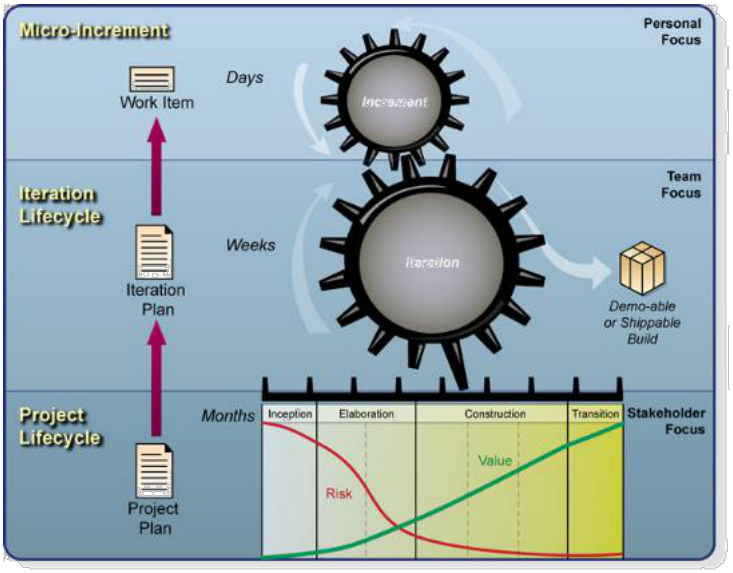

Figura 5.1: Ciclo de vida de projeto segundo o OpenUp (Fonte: [18]).

elaboração tem o propósito de estabelecer uma arquitetura (*baseline architecture*) e prover uma base estável para o esforço de desenvolvimento a ser realizado. A fase de construção tem o propósito de completar o desenvolvimento do sistema com base na arquitetura (*baseline architecture*) definida. Por fim, a fase de transição tem o propósito de garantir que o software está pronto para entrega aos usuários.

No OpenUp é possível fornecer para as equipes e para as partes interessadas (*stakeholders*) no projeto, os pontos em que são tomadas decisões de projeto a fim de definir qual plano seguir. O plano de projeto permite definir o ciclo de vida do projeto. O OpenUp provê informação acerca de como produto de software pode ser documentado e implementado. A descrição desse processo contém informação acerca de fases integrantes de ciclo de vida de projeto, processos de entrega (*delivery processes*), práticas (*practices*), papéis (*roles*), produtos do trabalho (*work products*), tarefas (*tasks*), orientação (*guidance*) e ferramentas (*tools*) [18].

5.1.1 Elementos relacionados à arquitetura de software

No OpenUp, existem atividades (*activity*), tarefas (*task*), artefatos (*work product*) e papéis (*role*) relacionados à arquitetura de software. Entre as atividades, tem-se as atividades Concordar com uma abordagem técnica (*Agree on a Technical Approach*) e Desenvolver a arquitetura (*Develop the Architecture*).A atividade Concordar com uma abordagem técnica visa definir abordagem técnica que suporte requisitos do projeto, considerando restrições impostas ao sistema e à equipe de desenvolvimento. Por sua vez, a atividade

Desenvolver a arquitetura visa refinar a arquitetura inicial em software funcional, produzir software estável que contemple os riscos técnicos [18].

No OpenUP, tarefas têm finalidades como detalhar requisitos, realizar a especificação da interface e da arquitetura do sistema de software. Tarefas são realizadas para concretizar atividade. Para cada tarefa são relacionados passos (*steps*).Para cada atividade existem tarefas a serem realizadas [7]. No OpenUp, arquiteto (*architect*) é responsável por conceber e refinar arquitetura de software, isso inclui tomar decisões que restringem o projeto (*design*) e a implementação do sistema. Na Figura 5.2 tem-se tarefas e artefatos relacionados a arquiteto. A seguir, são relacionadas tarefas realizadas por arquiteto e artefato sob sua responsabilidade [18]:

- Conceber a arquitetura;
- Refinar a arquitetura;
- Caderno de arquitetura.

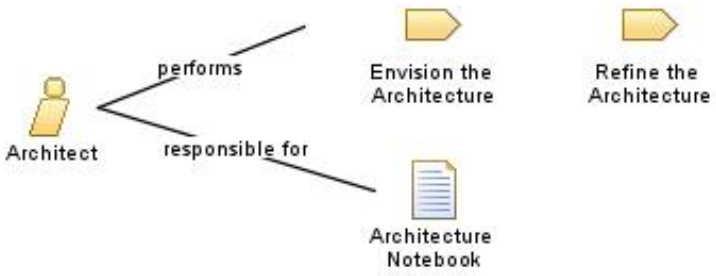

Figura 5.2: Tarefas e artefatos relacionados a arquiteto (Fonte: [18]).

A tarefa Conceber a arquitetura (*Envision the Architecture*) integra a atividade Concordar com uma abordagem técnica. A tarefa Conceber a arquitetura enfoca a visualização da arquitetura e a delimitação das decisões acerca da arquitetura de software. Esse detalhamento guiará o desenvolvimento e o teste do sistema de software. Os resultados obtidos são utilizados como referências futuras na comunição entre equipes. A seguir, são relacionados passos nessa tarefa segundo o OpenUp [18]:

- Identificar objetivos arquitetônicos;
- Identificar requisitos arquitetonicamente significativos;
- Identificar as restrições na arquitetura;
- Identificar as principais abstrações;

- Identificar oportunidades de reutilização;
- Definir abordagem para particionar o sistema de software;
- Definir a abordagem para implantar o sistema de software;
- Identificar mecanismos de arquitetura;
- Identificar interfaces para sistemas externos;
- Verificar consistência arquitetônica;
- Capturar e comunicar as decisões arquitetônicas.

A tarefa Refinar a arquitetura (*Refine the architecture*) integra a atividade Desenvolver a arquitetura. A tarefa Refinar a arquitetura tem o propósito de delinear e definir decisões arquitetônicas. Sendo assim, a partir da implementação do sistema de software e das suas modificações, a arquitetura de software evolui. Isso ocorre pois, por meio da implementação, é possível avaliar se a arquitetura de software é viável e se fornece o suporte que o sistema de software necessita ao longo do ciclo de vida [18].A seguir, são relacionados passos nessa tarefa segundo o OpenUp [18]:

- Refinar os objetivos arquiteturais e os requisitos arquitetonicamente significativos;
- Identificar os elementos de projeto (*design*) arquitetonicamente significativos;
- Refinar mecanismo de arquitetura;
- Definir a arquitetura de desenvolvimento de teste;
- Identificar oportunidades adicionadas de reutilização;
- Validar a arquitetura;
- Mapear o software para o hardware;
- Comunicar as decisões.

O artefato Caderno de arquitetura (*Architecture notebook*) descreve decisões tomadas acerca da arquitetura. No OpenUP, é sugerido que por meio desse artefato seja possível descrever mecanismos arquiteturalmente significativos e onde devem esse mecanismos serem aplicados. A seguinte informação pode ser registrada nesse artefato [18]:

- Objetivos e filosofia da arquitetura;
- Suposições arquiteturais e dependências;
- Referências para requisitos arquiteturalmente significativos;

- Referências para elementos de projeto (*design*) arquiteturalmente significativos;
- Interfaces de sistema críticas;
- Instruções de empacotamento para subsistemas e componentes;
- Camadas e subsistemas críticos;
- Abstrações chave;
- Cenários chave que descrevem comportamento crítico do sistema.

5.2 Configuração de processo de desenvolvimento de software

A configuração do OpenUP proposta neste trabalho, considera a definição de arquitetura de software no cenário de arquitetura de novo sistema de software. A descrição que será realizada para a arquitetura de software poderá ser utilizada ao longo de ciclo de vida do sistema de software. Dessa forma é possível acompanhar alterações de acordo com o uso do sistema de software. A descrição pode também ser usada para avaliar mudanças que podem acontecer ao longo de ciclo de vida de software [7].

A configuração do processo de desenvolvimento OpenUP proposta neste trabalho consiste em incluir prática para descrever arquitetura de software recomendada em *ISO 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, e incluir, em tarefas integrantes do OpenUP, passos (*steps*) com o propósito de analisar arquitetura de software por meio do método de análise de arquitetura *Software Architecture Analysis Method (SAAM).*

5.2.1 Prática para descrição de arquitetura

A prática a ser incluída no OpenUP consiste naquela descrita na norma *IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.* Descrição de arquitetura que seja construída considerando-se a prática que é descrita nessa norma, deve conter os elementos relacionados a seguir [7]:

- Identificação, versão e informação geral;
- Identificação de partes interessadas (*stakeholders*) e interesses dessas partes que sejam considerados relevantes à arquitetura;
- Especificações de cada ponto de vista (*viewpoint*) selecionado para organizar a representação da arquitetura e razão para essa seleção;

- Uma ou mais visões;
- Registros de inconsistências entre elementos constituintes requeridos pela descrição da arquitetura;
- Razão para seleção da arquitetura.

A norma *IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems* sugere incluir a seguinte informação em descrição de arquitetura [7]:

- Data de emissão e estado;
- Organização emissora;
- Histórico de modificações;
- Sumário;
- Escopo;
- Contexto;
- Glossário;
- Referências.

A norma *IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems* recomenda que sejam identificadas as partes interessadas (*stakeholders*) no sistema e que seja registrada informação acerca delas. São classes de partes interessadas (*stakeholders*) no sistema: usuário de sistema, adquirente de sistema, desenvolvedor de sistema e responsável por manutenção de sistema. Acerca do sistema, a norma sugere registrar a seguinte informação [7]:

- Propósitos ou missões do sistema;
- Adequação do sistema no atendimento das suas missões;
- Riscos de desenvolvimento;
- Riscos de operção do sistema;
- Manutenibilidade, implementação e capacidade de otimização do sistema.

A especificação de cada ponto de vista (*viewpoint*) pode incluir informação sobre práticas associadas ao ponto de vista. Por exemplo, informação acerca de testes formais ou informais de consistência e completude que podem ser aplicados ao modelo presente na visão analisada; informação acerca de técnicas para avaliar ou analisar o que será

implementado no sistema; informação acerca de padrões que podem auxiliar na construção da visão. Acerca de cada ponto de vista de arquitetura, a norma *IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems* sugere registro da seguinte informação [7]:

- Nome do ponto de vista;
- Partes interessadas (*stakeholders*) a serem abordadas pelo ponto de vista;
- Demandas a serem contempladas pelo ponto de vista;
- Linguagens, técnicas ou métodos a serem usados na construção de visão;
- Referência.

Em seguida, tem-se registro de informação acerca das visões (*views*) de arquitetura. As visões de arquitetura variam de acordo com o ponto de vista e cada visão de arquitetura corresponde a um ponto de vista. A norma *IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems* sugere que seja registrada a seguinte informação acerca de cada visão de arquitetura [7]:

- Identificador;
- Informação introdutória;
- Representação do sistema construído segundo as recomendações do ponto de vista associado;
- Informação de configuração.

Deve ser analisada a consistência entre visões (*views*) de arquitetura e ser registrada informação acerca de inconsistências identificadas. O registro dessa informação pode minimizar erros e a presença de defeitos; facilitar o alinhamento entre desenvolvedores; facilitar o acompanhamento da evolução das visões de arquitetura. Caso a inconsistência tenha sido solucionada deve ser informado como foi solucionada. É necessário informar a razão para os conceitos escolhidos acerca de arquitetura e prover evidência de que conceitos alternativos foram considerados. Essa informação pode facilitar o entendimento de decisões de projeto (*design*) tomadas [7].

5.2.2 Passos para análise de arquitetura

Quanto aos passos (*steps*) a serem incluídos em tarefas integrantes do OpenUp com o propósito de analisar a arquitetura de software por meio do método de análise de arquitetura de software denominado *Software Architecture Analysis Method (SAAM)*, são sugeridos os seguintes passos considerando as etapas do método [14] [17]:

- Descrever arquitetura candidata;
- Desenvolver cenários;
- Classificar cenários;
- Realizar avaliação individual de cenário;
- Realizar avaliação de interação de cenário;
- Realizar avaliação geral.

Capítulo 6

Exemplo de uso de processo de desenvolvimento

Este capítulo aborda o exemplo de uso do processo de desenvolvimento apresentado nos capítulos anteriores configurado. Entre os elementos que integram este capítulo, é possível destacar os seguintes: ciclo de vida do OpenUp, fase de inciação ,fase de concepção, especificação de requisitos, conceber arquitetura de software e refinar arquitetura de software.

6.1 Ciclo de vida do OpenUp

Para realizar a implementação do estudo de caso que será abordado neste capítulo, foi seguido como base a para iniciação do projeto o ciclo de vida do OpenUp [18].Este ciclo conta como base um fluxo de traballho que conta com as seguintes fases:

- Fase de iniciação;
- Fase de elaboração;
- Fase de construção;
- Fase de transição.

O fluxo de trabalho para esse ciclo de vida é apresentado na Figura 6.1.Em cada fase existem atividades e tarefas que devem ser executadas para seguir a configuração proposta pelo OpenUp.No contexto deste traballho serão enfocadas as fases de iniciação e a fase de elaboração.

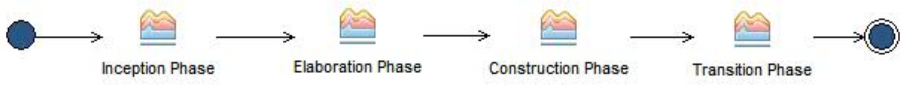

Figura 6.1: Fases do ciclo de vida do OpenUp (Fonte: [18]).

6.1.1 Fase de iniciação

A fase de iniciação tem como principal foco ter um acordo que resolva o problema unindo as necessidades das partes interessadas (*stakeholders*) e capturando os recursos necessários para o que o sistema de software seja resolvido.Na fase de inciação, tem-se como atividade a identificação e refinamento de requisistos, para isso foram desenvolvidos os seguintes artefatos [18]:

- Atributos de qualidade e restrições com abrangência de sistema(*system wide requirements*);
- Cenários de caso de uso;
- Glossário;
- Concordar com a abordagem técnica.

O artefato de atributos de qualidade e restrições com abrangência de sistema (*system wide requirements*) tem como objetivo capturar os atributos de qualidade e as restrições que tem escopo em todo sistema. Ele também captura os requisitos funcionais do sistema.

Por definição, tem-se que os requisitos funcionais do sistema de software correspondem à declarações de serviços que o sistema deve fornecer, em como o sistema deve reagir a entradas específicas no sistema de software. Além de indicar como o sistema deve se comportar em determinadas situações.Em alguns casos, os requisitos funcionais também podem explicitar o que o sistema não deve fazer ao receber uma entrada específica [3].

O artefato atributos de qualidade e restrições com abrangência de sistema (*system wide requirements*) tem como propósito identificar [18]:

- Descrever os atributos de qualidade do sistema e as restrições que as opções de projeto devem satisfazer para entregar as metas, objetivos ou capacidades do negócio;
- Capturar requisitos funcionais que não são expressos como casos de uso;
- Negociar e selecionar opções de design concorrentes;
- Avaliar o dimensionamento, custo e viabilidade do sistema proposto;

- Entender os requisitos de nível de serviço para gerenciamento operacional da solução.

Os cenários de caso de uso têm como objetivo capturar o comportamento do sistema entre um ou mais atores para produzir um resultado que seja de fácil observação para os usuários que devem interagir com o sistema.O ator é o papel que uma pessoa ou um sistema externo desempenha ao interagir com um sistema que está sendo analisado. Cada ator fornece uma perspectiva diferente do sistema de software, permitindo uma visão única em cada análise pelo ator [18].

A funcionalidade de um sistema é definida por diferentes casos de uso, cada um dos quais representa um objetivo específico (obter o resultado observável de valor) para um determinado ator [18].

O glossário tem como objetivo reunir os principais termos que são utilizados no projeto.A coleção de termos presente no glossário esclarece o vocabulário que é utilizado no projeto [18].

Em seguida, pode ser analisada a abordagem técnica que será utilizada. Para essa análise tem-se uma tarefa a ser cumprida que é a visualização da arquitetura.Essa tarefa é desenvolvida através da análise dos requisitos arquiteturais significativos e identificação de restrições, decisões e objetivos arquiteturais. Essa tarefa é relevante pois a partir das informações que são obtidas é possível fornecer orientações o suficiente para que a implementação do sistema possa ser iniciada [18].

Sendo assim, tem-se que nesta tarefa será construído um artefato. Com isso, tem-se que como tarefa,tem-se:

- Concordar com a abordagem técnica.

E como artefato, tem-se:

- Caderno de arquitetura.

Esse artefato descreve a lógica, as suposições, a explicação e as implicações das decisões que foram tomadas na formação da arquitetura [18].

6.1.2 Fase de concepção

Especificação de requisitos

Especificação de requisitos funcionais

Especificação de requisitos não funcionais

6.2 Conceber arquitetura de software

6.3 Refinar arquitetura de software

Capítulo 7

Avaliação do projeto prático diante da configuração e uso de processo de desenvolvimento do software

Capítulo 8

Conclusão

Referências

[1] *Software and systems engineering vocabulary.* Disponível em:`https://pascal.computer.org/sev_display/index.action`. Acesso em: 01 de março de 2022. 2, 10

[2] ISO/IEC/IEEE: *Systems and software engineering — Vocabulary.* Institute of Electrical and Electronics Engineers, 2010. 2

[3] Sommerville, Ian: *Engenharia de Software.* Pearson Education,São Paulo, 2011. 9.ed. 2, 10, 29

[4] BASS, Len ; CLEMENTS, Paul ;KAZMAN, Rick: *Software architecture in practice.* Addison-Wesley Professional, 2003. 2, 3

[5] University, Carnegie Mellon: *What is your definition of software architecture?* Disponível em:`https://resources.sei.cmu.edu/asset_files/factsheet/2010_010_001_513810.pdf`, 2017. Acesso em 01 de março de 2022. 2, 3

[6] CLEMENTS, Paul , et.al: *Documenting Software Architectures: Views and Beyond.* Addison-Wesley, 2010. 3

[7] ISO/IEC/IEEE: *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.* Institute of Electrical and Electronics Engineers, 2000. 3, 4, 6, 7, 24, 25, 26

[8] ISO/IEC/IEEE: *Systems and software engineering - Architecture description.* Institute of Electrical and Electronics Engineers, 2011. 4, 6, 7, 8

[9] ISO/IEC/IEEE: *Systems and software engineering - Content of life-cycle information items (documentation).* Institute of Electrical and Electronics Engineers, 2019. 5

[10] *The togaf standard, version 9.2 overview.* Disponível em:`https://pubs.opengroup.org/architecture/togaf9-doc/arch/`. Acesso em: 03 de março de 2022. 7

[11] Kruchten, Philippe: *Architectural blueprints—the "4+1" view model of software architecture.* IEEE Software, página 42, 1995. `https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf`. 8

[12] ISO/IEC/IEEE: *Systems and software engineering-Life cycle management.* Institute of Electrical and Electronics Engineers, 2020. 9, 10

[13] ISO/IEC/IEEE: *Systems and software engineering-Software life cycle processes*. Institute of Electrical and Electronics Engineers, 2017. 10, 11, 12, 13

[14] DOBRICA, Liliana; NIEMELA, Eila: *A survey on software architecture analysis methods*. IEEE Transactions on Software Engineering, página 638, 2002. https://www.researchgate.net/publication/3188246_A_survey_on_software_architecture_analysis_methods. 14, 18, 26

[15] STOERMER, Christoph; BECHMANN, Felix e Chris VERHOEF: *Sacam: The software architecture comparison analysis method*, 2003. https://resources.sei.cmu.edu/asset_files/TechnicalReport/2003_005_001_14219.pdf. 14, 15, 16

[16] KAZMAN, Rick;KLEIN, Mark e Paul CLEMENTS: *Atam: Method for architecture evaluation*, 2000. https://resources.sei.cmu.edu/asset_files/TechnicalReport/2000_005_001_13706.pdf. 16, 17

[17] KAZMAN, Rick, et.al.: *Scenario-based analysis of software architecture*. IEEE Software, 1996. 18, 19, 26

[18] *Openup*. Disponível em:https://download.eclipse.org/technology/epf/OpenUP/published/openup_published_1.5.1.5_20121212/openup/index.htm. Acesso em: 04 de abril de 2022. 20, 21, 22, 23, 28, 29, 30