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, 15 de setembro de 2022

Dedicatória

Agradecimentos

Resumo

Palavras-chave: LaTeX, metodologia científica, trabalho de conclusão de curso

Abstract

Keywords: LaTeX, scientific method, thesis

Sumário

Lista de Figuras

Lista de Abreviaturas e Siglas

ADM Architecture Development Method.

ATAM Architecture Tradeoff Analysis Method.

IEEE The Institute of Electrical and Electronics Engineers.

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,possíveis usos para a 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, 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. Dentro desse 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 estar alocado em 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. Ou seja, a arquitetura de software é uma abstração do sistema de software.Na arquitetura 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 sua arquitetura de software 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".* [5] [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 são estabelecidos relacionamentos entre os mesmos. Dessa forma são obtidas as funcionalidades que o sistema precisa prover para atingir os seus objetivos [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. Sendo assim, a descrição de arquitetura de software expressa sistema de software durante ciclo de vida do mesmo.A 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 resultante de atividades executadas ao longo de ciclo de vida de sistema de software [7] [8].

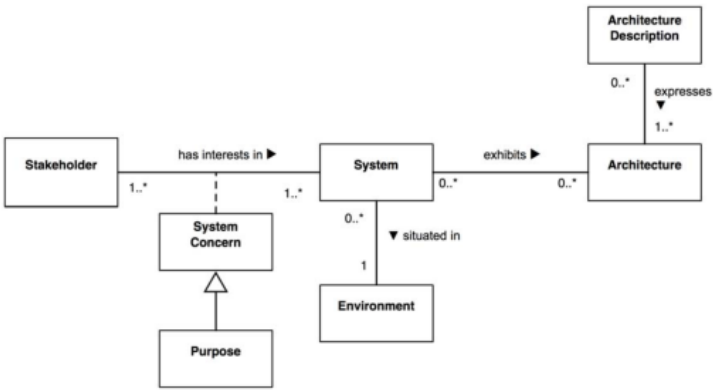

Figura 2.1: Contexto da descrição da arquitetura de software (Fonte: [8]).

Por fim, o diagrama presente na Figura 2.1 representa o contexto no qual ocorre descrição de arquitetura. 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 necessária para o sistema ter seu funcionamento. Tendo essas definições é possível descrever a arquitetura do software [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].

Tendo em vista que cada visão da arquitetura de software aborda um ponto de vista do sistema e esse ponto de vista pode vir de uma parte interessada do sistema, podendo ser o usuário final, o stakeholder , o cliente entre outras partes interessadas. E cada parte interessada pode possuir uma demanda que deve ser abordada na descrição da arquitetura de software.

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 os princípios para particionar o sistema em elementos do sistema (hardware, software, operações etc.) e elementos de projeto; registrar as propriedades importantes e os 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 [9].

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 Possíveis 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 [8].

A descrição de arquitetura de software pode promover a comunicação entre desenvolvedores do sistema de software e partes interessadas (*stakeholders*) no sistema de software. Ela permite facilitar o entendimento de funcionalidades do sistema de software e estrutura do

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

2.3 Ponto de vista e visão de arquitetura de software

A 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 à 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 [8].

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 [7] [8].

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.

2.4 Práticas para descrição de arquitetura de software

Na norma IEEE 1471-2000 IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, proposta pelo The Institute of Electrical and Electronics Engineers (IEEE), são sugeridas práticas para a descrição de arquiteturas de sistemas intensivos em software. Uma descrição que seja realizada levando em consideração as práticas propostas nessa norma, inclui os seguintes elementos [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.

Acerca da descrição de arquitetura como um todo, a norma IEEE 1471-2000 recomenda que a seguinte informação esteja presente na descrição da arquitetura [7]:

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

Em seguida, devem ser identificadas as partes interessadas (*stakeholders*) no sistema de software. A seguir, são relacionadas classes de possíveis partes interessadas (*stakeholders*) a serem identificadas: usuário de sistema, adquirente de sistema, desenvolvedor de sistema e responsável por manutenção de sistema [7].

Após serem identificadas as partes interessadas (*stakeholders*) no sistema e ser registrada informação acerca das mesmas, a norma IEEE 1471-2000 recomenda que a seguinte informação seja registrada acerca do sistema [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.

Em seguida, tem-se o registro de informação acerca dos pontos de vista (*viewpoint*) de arquitetura considerados. A norma IEEE 1471-2000 sugere que seja registrada a seguinte informação acerca de cada ponto de vista [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.

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 [7].

Em seguida, tem-se registro de informação acerca das visões (*views*) de arquitetura. Essas 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 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 que sejam identificadas. O registro dessa informação pode, por exemplo, minimizar a ocorrência de erros e a presença de defeitos; facilitar o alinhamento entre desenvolvedores; facilitar o acompanhamento da evolução das visões de arquitetura. Por fim, é necessário informar a razão para os conceitos escolhidos acerca de arquitetura e prover evidência de que conceitos alternativos foram considerados. Essa informação é relevante, pois pode facilitar o entendimento de motivos de decisões de projeto (*design*) [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 TOGAF consiste de arcabouço para arquitetura organizacional (*enterprise architecture*). 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 TOGAF inclui o Architecture Development Method (ADM), um método que é composto por fases e que se destina ao desenvolvimento e ao gerenciamento de ciclo de vida de arquitetura organizacional [8] [10].

Essa metodologia tem como principal objetivo garantir uma melhor eficiência do negócio a partir de uma estrutura de arquitetura de software consistente. Essa consistência é obtida através da aplicação de métodos e padrões entre os arquitetos de software envolvidos. Como principais funcionalidades, tem-se que o The Open Group´s Architecture Framework (TOGAF) permite que sejam abordadas padronizações que permitem a correção de erros, a documentação da estrutura e remoção de conteúdos que não são relevantes para a 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 modelo 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 a 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*) [11].

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 tornam o processo de compreender e detalhar a estrutura da 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, ou seja, devem ser definido qual é o principal serviço do sistema de software irá oferecer, quem serão os seus usuários finais, entre outras especificações. Após esse estágio tem-se o momento de desenvolvimento do sistema de

software e para verificar a sua funcionalidade é passado para o estágio de produção. O sistema de software passando por todos os testes de usabilidade pode ser liberado para utilização dos usuários finais. Tendo o sistema de software divulgado o estágio pode ser alterado, passando para o estágio de suporte, momento no qual pode ser necessária alguma modificação no sistema de software que deve ser sanado nesse estágio. Essa modificação pode ser de visualização, de uma nova atualização que permita uma melhor usabilidade do sistema de software, entre outras possibilidades. 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

Ao longo de ciclo de vida de software, a definição de arquitetura de software pode ser realizada por meio de processos. 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. Necessidades são 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

Dentre os processos técnicos que podem ser implementados em ciclo de vida de software, tem-se o processo de definição de arquitetura (*architecture definition process*). Esse processo 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. As opções de arquitetura geradas são analisadas levando-se em consideração pontos de vista (*viewpoints*) e visões (*views*) de arquitetura [13].

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 de software [13].

Capítulo 4

Métodos para avaliação de arquitetura de software

Este capítulo aborda 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 comparasion analysis method, architecture tradeoff analysis method, software architecture analysis method.

4.1 Avaliação de arquitetura de software

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

- Avaliar cada arquitetura candidata em relação às restrições e 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 uma linha de base para arquitetura selecionada, essa linha de base deve conter modelos, visualizações e demais especificações referentes à arquitetura de software escolhida.

Entre os processos técnicos em projeto de software,tem-se processo voltado à definição de arquitetura de software.No contexto desse processo, as seguintes atividades devem ser realizadas [13]:

- Preparar para definição arquitetura;
- Desenvolver pontos de vista da arquitetura;
- Desenvolver pontos de vista e visões de arquiteturas candidatas;
- Relacionar a arquitetura ao projeto (*design*);
- Avaliar arquiteturas candidatas;
- Gerenciar a arquitetura candidata.

4.2 Métodos para avaliação de arquitetura de software

Para avaliar arquitetura de software, pode ser adotado o método desenvolvido com 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 métodos [14]:

- Software Architecture Analysis Method (SAAM);
- The Software Architecture Comparison Analysis Method (SACAM);
- Architecture Tradeoff Analysis Method (ATAM).

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, essa escolha varia de acordo com o sistema de software. Dentre as etapas que devem ser seguidas, as seguintes [15]:

- Preparação;
- Coleção de critérios;

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

Na etapa Preparação são identificados os objetivos de negócios que são relevantes para a avaliação entre as arquiteturas candidatas e avalia a documentação para cada arquitetura candidata. Na etapa Coleção de critérios são agrupados os critérios de classificação de acordo com os objetivos de negócio.Na etapa Determinação de diretrizes de extração são determinadas as visões arquitetônicas, táticas, estilos e padrões que são necessários na construção dos cenários. Na etapa de exibição e extração de indicadores 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 coleção de critérios.E por fim, nessa etapa também devem ser implementadas técnicas de recuperação da arquitetura de software.Na etapa de Pontuação a arquitetura candidata é avaliada e recebe uma pontuação. Essa pontuação avalia se a arquitetura candidata consegue dar suporte aos critérios que foram estabelecidos para a arquitetura de software.Por fim, na etapa de resumo é realizado um resumo que contém os resultados e análises das arquiteturas candidatas que foram avaliadas e fornece uma recomendação para a escolha da arquitetura de software dentre as arquiteturas candidatas que foram avaliadas.

No método SACAM é obtida, como resultado da avaliação da arquitetura e informação acerca da pontuação de cada arquitetura candidata com as suas respectivas justificativas pelo motivo da pontuação que cada arquitetura candidata irá receber. 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 as técnicas que podem ser adotadas,tem-se as seguintes [15]:

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

A técnica de geração de cenário permite capturar os atributos de qualidade que são estabelecidos de acordo com o objetivo do sistema de software e refina esses atributos em cenários de atributos de qualidade. A técnica de táticas tem como finalidade obter as qualidades particulares que foram solicitas o Software Architecture Comparison Analysis Method (SACAM) utiliza as táticas nas avaliações da arquitetura de software como indicador para avaliar se as visões extraídas suportam o critério que está sendo avaliado.A técnica de métricas permite realizar análises quantitativas que fornecem indicadores úteis de complexidade geral e aponta aonde a mudança na arquitetura de software pode ser mais difícil ou mais provável. Nesse método as métricas são utilizadas no nível de código, se disponível, ou em um nível de design detalhado.Na técnica de padrões de documentação arquitetônica o Software Architecture Comparison Analysis Method (SACAM) exige a disponibilidade de documentação arquitetônica para realizar a análise de critérios e a avaliação de comparação entre as arquiteturas candidatas. Por fim, na técnica de reconstrução de arquitetura, caso a documentação da arquitetura esteja desatualizada ou indisponível a arquitetura precisa ser reconstruída.

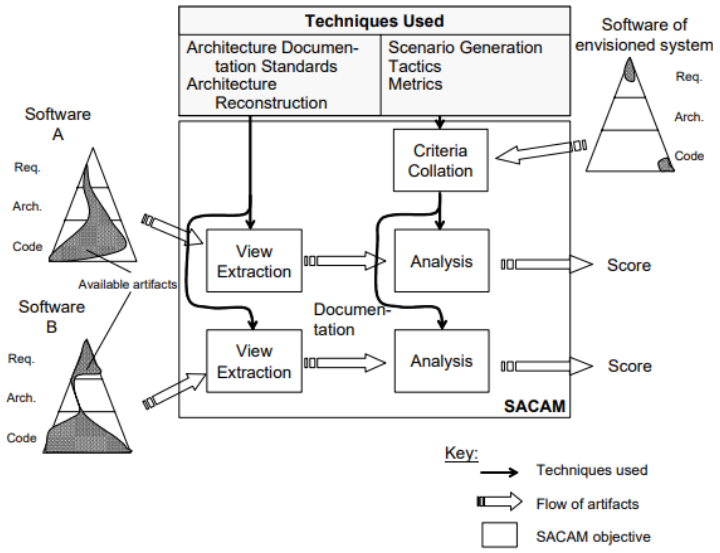

Figura 4.1: Técnicas de arquitetura utilizadas pelo SACAM (Fonte: [15]).

Na Figura 4.1 são mostradas técnicas que podem ser utilizadas. A partir dessas técnicas são gerados os artefatos necessários para que seja possível avaliar as arquiteturas candidatas e atribuir pontuações a cada arquitetura. Dessa forma é possível obter informação que possibilite comparar as arquiteturas candidatas e eleger a mais adequada para o sistema de software.

Sendo assim, esse método auxilia no processo de seleção da arquitetura de software fornecendo resultados da análise da arquitetura de software. Para fornecer esses resultados

o SACAM compara as arquiteturas com base em um conjunto de critérios que são estabelecidos de acordo com os objetivos de negócio da empresa que está interessada na arquitetura de software [15].

4.4 Architecture Tradeoff Analysis Method

O método Architecture Tradeoff Analysis Method (ATAM) tem como objetivo entender as consequências das decisões arquiteturais que foram tomadas tendo como ponto de partida os requisitos de atributos de qualidade do sistema de software. Sendo assim, é possível determinar se os objetivos do sistema de software poderão ser atendidos pela arquitetura de software que será escolhida [16].

Esse método orienta os usuários interessados a procurar pontos problemáticos e soluções para esses pontos na arquitetura de software. Além disso, esse método também pode ser utilizado para analisar sistemas legados. Esses sistemas legados são sistemas antigos, mas que continuam em operação. Sendo assim, esse método 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 de análise que é organizado como o modelo arquitetônico são determinados, verificando se no modelo arquitetônico estão presentes os atributos de qualidade que foram demandados pelas partes interessadas no sistema de software. Para realizar esse método, são seguidos os passos a seguir descritos [16]:

No passo de apresentação, tem-se [16]:

- Apresentação;
- Impulsionadores dos negócios atuais;
- Arquitetura atual.

Na fase de apresentação é realizada a descrição do método para as partes interessadas no sistema de software. Na fase de impulsionadores dos negócios atuais o gerente de projeto descreve quais são os objetivos do negócio e quais serão os principais impulsionadores arquitetônicos. Na fase de arquitetura atual o arquiteto de softwares deve descrever a arquitetura candidata e foca em como devem ser abordados os objetivos do negócio.

No passo de investigação e análise, tem-se [16]:

- Identificar as abordagens arquitetônicas;

- Gerar uma árvore de utilidades de atributos de qualidade;
- Analisar as abordagens arquitetônicas.

No passo de identificar as abordagens arquitetônicas as abordagens devem ser identificadas, mas não analisadas. No passo de gerar uma árvore de utilidades de atributos de qualidade deve ser realizada a elicitação de fatores de qualidade que compõem as características do sistema de software tendo como exemplo o desempenho,segurança, entre outras características do sistema de software. Esses fatores de qualidade devem ser especificados de acordo com o nível de cenário que será analisado. Também devem conter as respostas que são geradas a cada estímulo no sistema de software. Com isso devem ser priorizadas as utilidades de atributos de qualidade nesse passo. No passo de analisar as abordagens arquitetônicas tem-se que de acordo com os fatores de qualidade que foram priorizado no passo anterior devem ser analisadas as abordagens arquitetônicas que possuem esses fatores. Sendo assim, nesse passo devem ser identificados os riscos arquitetônicos, os pontos frágeis e os pontos aonde podem ocorrer mudanças na arquitetura de software.

No passo de teste, tem-se [16]:

- Realizar *brainstorming* e priorizar cenários;
- Analisar as abordagens arquitetônicas.

No passo de realizar *brainstorming* e priorizar cenários tem-se que de acordo com os cenários que foram analisados no passo anterior devem ser priorizados os cenários que forem mais bem votados dentro desse passo. Essa votação deve incluir todas as partes interessadas no sistema de software. No passo de analisar as abordagens arquitetônicas deve ser realizada a análise das visões arquitetônicas que foram aprovadas pelos passos anteriores. Nesse passo, os cenários avaliados são considerados para teste. No entanto esses cenários ainda podem revelar a necessidade de novas estruturas na arquitetura, novos riscos e até mesmo outros pontos na arquitetura de software.

O último passo é dedicado ao de comunicação, nesse passo tem-se [16]:

- Apresentação de resultados.

Como último passo tem-se a apresentação de resultados, nesse passo devem ser apresentadas as descobertas realizadas para as partes interessadas . Deve ser feito um relatório com o detalhamento das informações que foram obtidas e as estratégias propostas para a arquitetura de software.

Por fim, pode-se perceber que o método Architecture Tradeoff Analysis Method (ATAM) depende da comunicação entre todas as partes interessadas no sistema de software. Essa

comunicação é o que permite a evolução da arquitetura de software. Com isso, as técnicas têm como objetivo garantir que as necessidades do sistema de software sejam supridas pela arquitetura de software projetada [16].

4.5 Software Architecture Analysis Method

O Software Architecture Analysis Method (SAAM) consiste de método embasado em cenários, os cenários representam as necessidades que o sistema de software deve possuir. Sendo assim, cada cenário permite ilustrar as atividades que o sistema de software pode suportar e quais as mudanças que podem ocorrer durante o uso do sistema de software.

Logo , nessa avaliação também é determinado se o cenário precisa de modificações na sua arquitetura ou não. Caso o cenário precise de modificações são chamados de indiretos e caso o cenário não precise de modificações são chamados de diretos [14].

No contexto desse trabalho é adotado o método de arquitetura baseado em cenários, este é um método que analisa as propriedades da arquitetura de softwareSoftware Architecture Analysis Method (SAAM). Esse método tem como principal objetivo analisar a arquitetura de software de maneira geral, ou seja, tendo o entendimento de como o sistema de software deve funcionar de acordo com a arquitetura de software. Esse método guia o arquiteto de softwares a identificar os possíveis pontos problemáticos da arquitetura [14] [17].

Para exemplificar esses possíveis pontos problemáticos tem-se o ponto de conflito entre os requisitos ou o ponto em que arquitetura candidata pode possuir uma demanda não implementada ou incompleta, demanda essa que foi solicitada por uma das partes interessadas no sistema de software no momento das especificações da arquitetura de software [14]. O Software Architecture Analysis Method (SAAM) permite que seja realizada a comparação entre os cenários das arquiteturas candidatas e a partir dessa comparação pode ser escolhida a arquitetura que se alinhe melhor com o sistema de software [14].

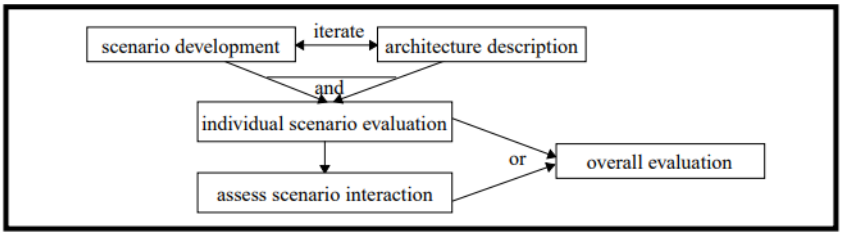

Figura 4.2: Etapas integrantes do SAAM (Fonte: [17]).

Na Figura 4.2 tem-se as seguintes etapas do SAAM [17]:

- Descrição de arquitetura;

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

Sendo assim, o método SAAM ao comparar os cenários permite que seja realizada a comparação entre as arquiteturas candidatas a fim de eleger a arquitetura de software mais adequada para o sistema de software. Ele permite que sejam integrados os diversos interesses das partes interessadas no sistema de software e estabelece um cenário que entregue todas as demandas solicitadas pelas partes interessadas através de uma arquitetura de software consistente 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 em uma notação que seja compreendida por todas as partes interessadas no sistema de software [17].

As descrições arquitetônicas devem indicar os componentes que serão necessários para a arquitetura candidata e os seus respectivos 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 os tipos de atividades que o sistema deve suportar e quais as mudanças que serão feitas no sistema ao longo do tempo [17].

No momento em que forem desenvolvidos os cenários, deve-se dar relevância à captura das necessidades a serem atendidadas pelo sistema de software. Visto que dessa forma 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 é realizada a avaliação dos cenários que foram desenvolvidos na etapa anterior. Inicialmente deve ser avaliado se com a arquitetura candidata que se tem é possível desenvolver o cenário proposto ou se é necessária alguma alteração na arquitetura candidata. Caso seja necessária alteração,o cenário em questão é chamado de cenário indireto [17].

Sendo necessária as alteração na arquitetura para executar o cenário indireto, devem ser listadas as alterações necessárias e também deve ser estimado o custo para realizar a alteração. Visto que uma alteração na arquitetura implicar que novo componente ou novo relacionamento seja introduzido na arquitetura candidata [17].

Ao final dessa avaliação, deve ser feito um quadro-resumo contendo todos os cenários (diretos e indiretos) e para cada cenário indireto, ou seja, cenário que precise de modificação na arquitetura candidata para executa-lo, é necessário descrever qual o impacto que essa modificação causa no sistema de software. Esse quadro-resumo permite que seja possível comparar as 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. Os cenários indiretos podem precisar de alterações em componentes ou em relacionamentos. Para determinar a interação do cenário é preciso identificar os cenários que afetam um conjunto comum de componentes [17].

Identificar essa interação é relevante pois ela identifica até que ponto a arquitetura candidata consegue suportar os cenários que estão sendo estabelecidos para o sistema de software [17].

4.5.5 Avaliação geral

Na quinta e última etapa deve ser realizada uma avaliação geral da arquitetura candidata. Sendo assim, 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 [17].

Esse processo deve envolver as partes interessadas no sistema. Com essa ponderação, será 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, atividades, tarefas e artefatos relacionados à arquitetura de software no OpenUp, atividades relacionadas à arquitetura de software e configuração do processo de definição de arquitetura de software.

5.1 Open Unified Process

O processo de desenvolvimento de software, a ser consfigurado no contexto deste trabalho,é denominado Open Unified Process (OpenUp).Na configuração desse processo de desenvolvimento serão adicionadas atividades, tarefas e artefatos descritos nas seguintes normas: ISO 12207 - Systems and software engineering - Software life cycle processes, ISO 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, ISO 15289-2027 - Systems and software engineering - Content of life-cycle information items (documentation) e ISO 42010-2021 - Systems and software engineering - Architecture description.A configuração do processo de desenvolvimento terá o propósito de melhor prover suporte ao projeto (design) de software. Particularmente, à definição de arquitetura de software.

O Open Unified Process (OpenUp) sugere que o desenvolvimento de software seja organizado em incrementos, dessa forma unidades pequenas de trabalho podem produzir um trabalho contínuo que demonstre uma métrica que permita mensurar o progresso do projeto. Sendo assim, é possível acompanhar e documentar o desenvolvimento do *software* de forma incremental, sendo artefatos construídos na medida em que progride o desenvolvimento do software. 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 [18].

A partir da interação entre as equipes é possível determinar para qual fase o projeto pode seguir, visto que a partir da interação entre as equipes tem-se uma estimativa do quanto o projeto teve andamento durante o período analisado [18].

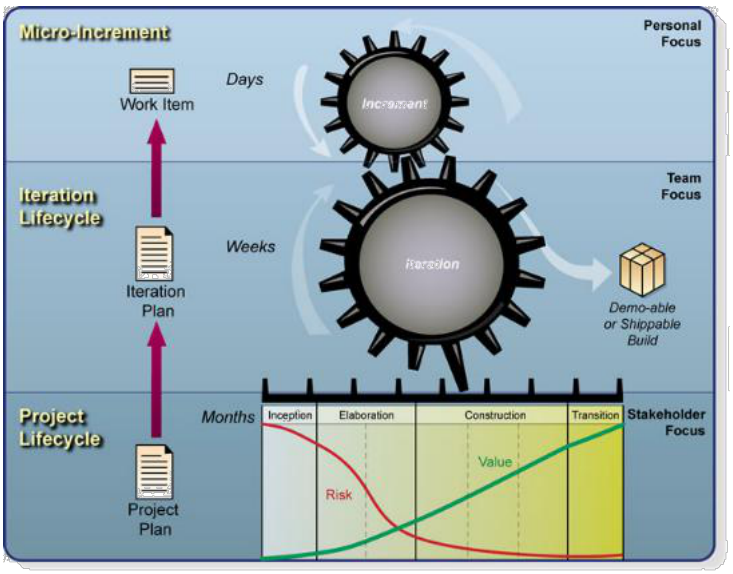

Figura 5.1: Ciclo de vida de projeto segundo o OpenUp (Fonte: [18]).

O ciclo de vida de projeto é composto pelas seguintes fases: iniciação, elaboração, construção e transição. 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 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. A fase de transição tem o propósito de garantir que o software está pronto para entrega aos usuários.

Com isso, no OpenUp é possível fornecer para as equipes e para as partes interessadas (*stakeholder*) no projeto os pontos em que são tomadas decisões de projeto a fim de definir qual plano seguir. Dessa forma, o plano de projeto permite definir o ciclo de vida do projeto. O Open Unified Process (OpenUp) provê informação acerca de como produto de software pode ser documentado e implementado de forma incremental.O processo provê informação acerca de fases em ciclo de vida de projeto, por exemplo, acerca de responsáveis por cada fase [18].

No processo de configuração do OpenUP, será incluída informação acerca de processo de definição de arquitetura de software, seguindo-se recomentações em normas anteriormente

relacionadas. Sendo assim, a informação apresentada ao longo do capítulo 03 e do capítulo 04 deste trabalho será usada a fim de projetar arquiteturas candidatas para o sistema de software que será apresentado como exemplo de uso neste trabalho. As arquiteturas candidatas serão avaliadas através do Software Architecture Analysis Method (SAAM) a fim de eleger a melhor arquitetura para o sistema de software.Nesse processo de configuração serão determinadas atividades, tarefas e artefatos.

5.2 Atividades, tarefas e artefatos relacionados à arquitetura de software no OpenUp

A fim de definir as atividades, tarefas e artefatos que serão configurados neste projeto para desenvolver a arquitetura de software tem-se um responsável por definir o processo que deve ser seguido. Tendo como base o Open Unified Process (OpenUp), tem-se que o arquiteto é responsável por refinar a arquitetura de software, o que inclui tomar as principais decisões técnicas que restringem o projeto geral e a implementação do sistema.

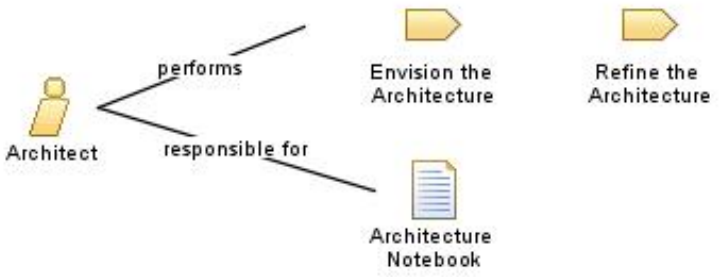

Figura 5.2: Tarefas e artefatos relacionados a arquiteto segundo o OpenUp (Fonte: [18]).

Na Figura 5.2 tem-se que o arquiteto tem como tarefas [18]:

- Projetar a arquitetura de software;
- Refinar a arquitetura de software

Como artefato que deve ser produzido pelo arquiteto de softwares tem-se o [18]:

- Caderno de arquitetura.

As tarefas têm como finalidade detalhar os requisitos e realizar a especificação da interface e da arquitetura do sistema de software. As tarefas são realizadas a fim de concretizar a atividade que será abordada. Sendo assim, para cada atividade existe uma lista de tarefas que devem ser realizadas[7].

Como tarefas, serão executadas as tarefas do Open Unified Process (OpenUp):

- Projetar a arquitetura de software;
- Refinar a arquitetura de software.

Na tarefa de Projetar a arquitetura de software enfoca a visualização da arquitetura inicial e delimitar as decisões da arquitetura de software. Esse detalhamento guiará o desenvolvimento e o teste do sistema de software. Ele é baseado na coleta de experiência que é obtida ao utilizar sistemas de softwares semelhantes ao que será apresentado no exemplo de uso. Os resultados que são obtidos são utilizados como referências futuras para realizar a comunição entre as equipes. Para essa tarefa devem ser seguidos os seguintes passos[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 de Refinar a arquitetura de software tem o propósito de delinear e definir as decisões arquitetônicas. Sendo assim, a partir da implementação do sistema de software e das suas modificações a arquitetura de software evolui. Isso ocorre pois apenas a partir da implementação é possível avaliar se a arquitetura de software é viável e fornece o suporte que o sistema de software necessita ao longo do ciclo de vida[18].

Dentre os passos que devem ser executados nessa tarefa, tem-se[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.

5.3 Atividades relacionadas à arquitetura de software

A configuração do processo de desenvolvimento de software considera a definição de arquitetura de software no cenário de arquitetura de novos sistemas de software. Esse cenário foi escolhido pois através dele será possível acompanhar a arquitetura de software ao longo de vida do sistema de software[7].

A arquitetura de software contribui para o desenvolvimento, operação e manutenção do sistema de software. Por isso realizar a configuração de um processo para determinar a arquitetura é relevante para o desenvolvimento do sistema de software. Com isso, as atividades, tarefas e artefatos que serão implementados têm como finalidade garantir as seguintes características na arquitetura de software[7]:

- Manter a integridade dos conceitos do sistema por meio do seu desenvolvimento;
- Certificar os sistemas que foram construídos assegurando-os conceitos de sistema através de fases operacionais e evolutivas.

Para seguir com o processo de arquitetura de software também serão feitos artefatos. Esses artefatos têm como objetivo apresentar através de uma documentação as decisões que foram realizadas durante o ciclo de vida do sistema de software que afetam na definição da arquitetura de software [7].

Os artefatos sugeridos estão entre aqueles relacionados nas seguintes normas:ISO 12207 - Systems and software engineering - Software life cycle processes, ISO 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, ISO 15289-2027 - Systems and software engineering - Content of life-cycle information items (documentation) e ISO 42010-2021 - Systems and software engineering - Architecture description.Como foi abordado serão aplicados os seguintes artefatos [7]:

- Documentação da arquitetura;
- Identificação das partes interessadas;
- Selecionar os pontos de vista arquitetônicos;

- Visões da arquitetura;
- Consistência entre as visões arquitetônicas.

Cada artefato possui uma especificação em sua documentação que deve ser seguida. Na documentação da arquitetura devem ser especificadas as seguintes informações sobre a arquitetura de software[7]:

- Data de atualização da arquitetura de software e o versionamento da arquitetura;
- Organização que solicitou o sistema de software;
- Histórico de alterações da arquitetura de software;
- Sumário;
- Escopo;
- Contexto da arquitetura de software;
- Glossário;
- Referências da arquitetura de software.

Na identificação das partes interessadas devem ser especificados os[7]:

- Usuários do sistema de software;
- Desenvolvedores do sistema de software.

Nesse artefato também devem ser específicas algumas necessidades que o sistema de software possui. Para isso, devem ser especificados os seguintes itens[7]:

- Propósito do sistema;
- Adequar o sistema de software para que o propósito seja atingido com sucesso;
- Viabilidade para construir o sistema;
- Identificar quais os riscos que o sistema de software pode apresentar de acordo com o ponto de vista do usuário final, das organizações interessadas e para os desenvolvedores do sistema de software;
- Manutenibilidade, implementação e capacidade de otimização do sistema.

Ao selecionar os pontos de vista do sistema de software deve ser pensado nas necessidades que as partes interessadas querem no sistema, sendo assim, nesse artefato devem ser especificados os seguintes itens[7]:

- Nome do ponto de vista;
- Partes interessadas a serem abordadas pelo ponto de vista;
- Demandas a serem abordadas pelo ponto de vista;
- Linguagens, técnicas de modelagem ou métodos analíticos que devem ser implementados na construção que será desenvolvida de acordo com o ponto de vista escolhido;
- Referência para escolher o ponto de vista.

No artefato que contém as visões arquitetônicas devem ser especificados os seguintes itens[7]:

- Identificador e demais informações introdutórias do sistema;
- Representante do sistema construído com as linguagens, métodos e modelagem ou análise técnicas do ponto de vista que está sendo analisado;
- Informações de configuração sendo definido de acordo com a organização do usuário.

Para o artefato relacionado a consistência entre as visões arquitetônicas devem ser documentadas as inconsistências que foram encontradas. E caso a inconsistência tenha sido resolvida deve ser especificado como essa inconsistência foi solucionada, pois, dessa forma pode ser mantida uma consistência na documentação do sistema de software[7].

As atividades, tarefas e artefatos que foram escolhidos para configurar o processo têm como finalidade permitir integrar diferentes abordagens a fim de tornar o projeto mais documentado. E para demonstrar a configuração desse processo será implementado em um exemplo de uso.

5.4 Configuração do processo de definição de arquitetura de software

No contexto deste trabalho serão adicionadas atividades, tarefas e artefatos que são sugeridos pelo The Institute of Electrical and Electronics Engineers (IEEE).Para isso, serão abordadas atividades que são descritas pelo ISO/IEC/IEEE 12207-2017.Dentre essas atividades tem-se[13]:

- Preparar-se para a definição da arquitetura de software;
- Desenvolver pontos de vista de arquitetura;
- Desenvolver modelos e visualizações de arquiteturas candidatas;

- Relacionar a arquitetura com o design;
- Avaliar as arquiteturas candidatas;
- Gerenciar a arquitetura selecionada.

Além dessa abordagem de atividades que são abordadas na ISO/IEC/IEEE 12207-2017, existem outras atividades que são relevantes dentro do contexto deste trabalho. Na ISO/IEC/IEEE 1471-2000 também são abordadas sugestões de atividades que podem ser realizadas. A abordagem utilizada é relacionada aos diferentes cenários em que o sistema de software será desenvolvido. Dentre as possíveis atividades, tem-se[7]:

- Cenário: Arquitetura de sistemas únicos;
- Cenário: Arquitetura iterativa para sistemas evolutivos;
- Cenário: Arquitetura de sistemas existentes.

Cada atividade aborda uma sequência de tarefas a serem desenvolvidas, isso ocorre tanto na ISO/IEC/IEEE 12207-2017 como na ISO/IEC/IEEE 1471-2000. A escolha das atividades a serem desenvolvidas devem ser escolhidas de acordo com o sistema de software que será desenvolvido. Sendo assim, para o contexto deste trabalho foram escolhidas atividades que têm origem tanto do Open Unified Process (OpenUp) como no The Institute of Electrical and Electronics Engineers (IEEE).

Para configurar o processo que será implementado neste trabalho, serão abordadas as seguintes atividades[7]:

- Cenário: Arquitetura de sistemas únicos.

Essa atividade foi escolhida pois, o exemplo de uso que será apresentado neste trabalho é um sistema novo. Nesse cenário a projeção da arquitetura do sistema de software será voltado a uma resposta em relação a visão do usuário final. Sendo assim, respeitada as necessidades em relação a funcionalidade e as limitações do sistema de software. Nesse caso, tem-se que as partes interessadas no sistema de software, serão[7]:

- Usuário final;
- Desenvolvedor.

A descrição que será realizada para a arquitetura de software poderá ser utilizada ao longo do ciclo de vida do sistema de software. Dessa forma é possível acompanhar eventuais necessidades de alterações de acordo com o uso do sistema de software e também pode ser utilizado como meio para avaliar eventuais mudanças que podem acontecer ao longo do ciclo de vida[7].

Como artefatos, serão desenvolvidas as que são sugeridas pela ISO/IEC/IEEE 1471-2000[7]:

- Documentação da arquitetura;
- Identificação das partes interessadas;
- Selecionar os pontos de vista arquitetônicos;
- Visões da arquitetura;
- Consistência entre as visões arquitetônicas.

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.

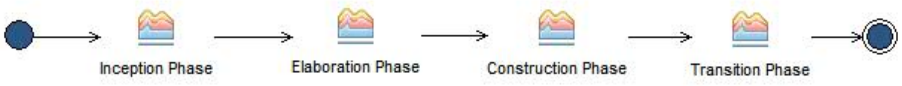

Figura 6.1: Fases do ciclo de vida do OpenUp (Fonte: [18]).

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, 12

[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, 12, 34

[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, 8, 26, 28, 29, 30, 31

[8] ISO/IEC/IEEE: *Systems and software engineering - Architecture description.* Institute of Electrical and Electronics Engineers, 2011. 4, 5, 6, 9, 10

[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. 9

[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`. 9, 10

[12] ISO/IEC/IEEE: *Systems and software engineering-Life cycle management.* Institute of Electrical and Electronics Engineers, 2020. 11, 12

[13] ISO/IEC/IEEE: *Systems and software engineering-Software life cycle processes*. Institute of Electrical and Electronics Engineers, 2017. 13, 14, 15, 30

[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. 16, 21, 22

[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. 16, 17, 18, 19

[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. 19, 20, 21

[17] KAZMAN, Rick, et.al.: *Scenario-based analysis of software architecture*. IEEE Software, 1996. 21, 22, 23

[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. 25, 26, 27, 33, 34, 35