



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
2023



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, 23 de janeiro de 2023

Dedicatória

Dedico este trabalho a todos que estiveram presentes na minha formação acadêmica. Dedico aos professores que se empenharam em passar o seu conhecimento da melhor maneira possível, contribuindo para a minha formação profissional. Em especial, dedico este trabalho à minha família que me apoiou e me acolheu em todas as minhas decisões acadêmicas.

Agradecimentos

Agradeço à minha família por ter me dado condições de ter uma educação de qualidade e por me apoiar em todos os momentos de decisões relevantes na minha formação acadêmica. Aos meus amigos por terem me dado suporte e terem paciência pelos momentos de ausência. Aos meus colegas de trabalho pela parceria no aprendizado diário e aos professores da universidade, especialmente ao professor Fernando Albuquerque, por ter aceitado ser o orientador deste trabalho e por todo o apoio durante o seu processo de desenvolvimento.

Resumo

No desenvolvimento de software existem processos que podem ser adotados na definição de arquitetura de software. Este trabalho visa promover o entendimento de processo de definição de arquitetura de software por meio da configuração de processo de desenvolvimento de software com elementos de processo de definição de arquitetura de software e uso do processo de desenvolvimento de software configurado.

Palavras-chave: Engenharia de Software, Processo de desenvolvimento de software, Arquitetura de Software, Processo de definição de arquitetura de software

Abstract

In software development there are processes that can be adopted in the definition of software architecture. This work aims to promote understanding of the software architecture definition process by configuring the software development process with software architecture definition process elements and using the configured software development process.

Keywords: Software Engineering, Software Development Process, Software Architecture, Software Architecture Definition Process

Sumário

| | | |
|----------|--|-----------|
| 1 | Introdução | 1 |
| 1.1 | Objetivo geral | 2 |
| 1.2 | Objetivos específicos | 2 |
| 1.3 | Motivações e justificativas | 2 |
| 1.4 | Estrutura do documento | 3 |
| 2 | Arquitetura de software | 4 |
| 2.1 | Arquitetura de software | 4 |
| 2.2 | Descrição de arquitetura de software | 6 |
| 2.2.1 | Usos para a descrição da arquitetura de software | 7 |
| 2.3 | Ponto de vista e visão de arquitetura de software | 8 |
| 2.4 | Práticas para descrição de arquitetura de software | 8 |
| 2.5 | Arcabouços de arquitetura | 9 |
| 3 | Processo de definição de arquitetura de software | 11 |
| 3.1 | Ciclo de vida de software | 11 |
| 3.2 | Processos em ciclo de vida de software | 12 |
| 3.3 | Processos técnicos em ciclo de vida de software | 12 |
| 3.4 | Processo de definição de arquitetura | 13 |
| 4 | Métodos para avaliação de arquitetura de software | 15 |
| 4.1 | Avaliação de arquitetura de software | 15 |
| 4.2 | Métodos para avaliação de arquitetura de software | 16 |
| 4.3 | Software Architecture Comparison Analysis Method | 16 |
| 4.4 | Architecture Tradeoff Analysis Method | 18 |
| 4.5 | Software Architecture Analysis Method | 20 |
| 4.5.1 | Descrição de arquitetura | 21 |
| 4.5.2 | Desenvolvimento de cenário | 21 |
| 4.5.3 | Avaliação individual de cenário | 21 |
| 4.5.4 | Avaliação de interação de cenário | 21 |

| | |
|---|-----------|
| 4.5.5 Avaliação geral | 22 |
| 5 Exemplo de configuração de processo de desenvolvimento | 23 |
| 5.1 Open Unified Process | 23 |
| 5.1.1 Elementos relacionados à arquitetura de software | 25 |
| 5.2 Configuração de processo de desenvolvimento de software | 27 |
| 5.2.1 Prática para descrição de arquitetura | 27 |
| 5.2.2 Passos para análise de arquitetura | 30 |
| 5.3 Artefatos construídos | 30 |
| 6 Exemplo de uso de processo de desenvolvimento | 31 |
| 6.1 Ciclo de vida do projeto | 31 |
| 6.2 Fase de concepção | 32 |
| 6.2.1 Atividade Iniciar Projeto | 33 |
| 6.2.2 Atividade Planejar e gerenciar iteração | 34 |
| 6.2.3 Atividade Identificar e refinar requisitos | 35 |
| 6.2.4 Atividade Concordar com abordagem técnica | 36 |
| 6.3 Fase de elaboração | 37 |
| 6.3.1 Atividade Desenvolver a arquitetura | 38 |
| 6.3.2 Atividade Desenvolver incremento da solução | 38 |
| 6.3.3 Atividade Planejar e gerenciar iteração | 39 |
| 6.4 Requisitos funcionais | 40 |
| 6.4.1 Funcionalidade: Autenticar usuário | 41 |
| 6.4.2 Funcionalidade: Visualizar perfil | 41 |
| 6.4.3 Funcionalidade: Consultar medicamentos registrados no sistema | 42 |
| 6.4.4 Funcionalidade: Realizar controle do estoque de medicamentos | 43 |
| 6.5 Cenários Funcionais | 44 |
| 6.6 Arquitetura do software | 45 |
| 6.7 Projeto da interface com o usuário | 45 |
| 6.8 Projeto de banco de dados | 47 |
| 6.9 Ferramentas de desenvolvimento | 48 |
| 7 Conclusão | 52 |
| 7.1 Considerações Finais | 52 |
| 7.2 Trabalhos Futuros | 52 |
| Referências | 54 |

Lista de Figuras

| | | |
|------|---|----|
| 2.1 | Contexto da descrição da arquitetura de software. | 6 |
| 4.1 | Técnicas de arquitetura utilizadas pelo SACAM. | 18 |
| 4.2 | Etapas integrantes do SAAM. | 20 |
| 5.1 | Ciclo de vida de projeto segundo o OpenUP | 24 |
| 5.2 | Tarefas e artefatos relacionados a arquiteto. | 25 |
| 6.1 | Fases de trabalho em ciclo de vida. | 31 |
| 6.2 | Fluxo de trabalho fase Concepção. | 32 |
| 6.3 | Elementos da atividade Iniciar projeto. | 33 |
| 6.4 | Elementos da atividade Planejar e gerenciar iteração. | 34 |
| 6.5 | Elementos da atividade Identificar e refinar requisitos. | 36 |
| 6.6 | Elementos da atividade Concordar com abordagem técnica. | 37 |
| 6.7 | Fluxo de trabalho da fase Elaboração.. . . . | 37 |
| 6.8 | Elementos da atividade Desenvolver a arquitetura | 38 |
| 6.9 | Elementos da atividade Desenvolver incremento da solução. | 39 |
| 6.10 | Elementos da atividade Planejar e gerenciar iteração | 39 |
| 6.11 | Atores e relacionamento entre atores. | 41 |
| 6.12 | Diagrama de caso de uso para visualizar perfil. | 42 |
| 6.13 | Diagrama de caso de uso para consultar medicamentos registrados no sistema. | 43 |
| 6.14 | Diagrama de caso de uso para controle de estoque do medicamento. | 45 |
| 6.15 | Código para realizar validação de login. | 46 |
| 6.16 | Tela de login implementada. | 47 |
| 6.17 | Tela principal para o gestor implementada. | 48 |
| 6.18 | Tela principal para o cidadão implementada. | 48 |
| 6.19 | Tela de estoque de medicamentos. | 49 |
| 6.20 | Tela de cadastro de medicamentos. | 49 |
| 6.21 | Modelo relacional do GSUS | 50 |
| 6.22 | Código para realizar conexão com banco de dados. | 51 |

Lista de Abreviaturas e Siglas

ADM Architecture Development Method.

ATAM Architecture Tradeoff Analysis Method.

CSS Cascading Style Sheets.

HTML HyperText Markup Language.

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

Para desenvolver software é necessário executar várias atividades. Por exemplo, atividades para especificação de software, essas são atividades que referem-se às funcionalidades e às restrições relativas ao software; projeto (*design*) e construção de software, essas são atividades que referem-se a como software deve ser desenvolvido para atender às demandas das partes interessadas (*stakeholders*); validação de software, essas são atividades que referem-se a como software deve ser validado para garantir que atende às demandas das partes interessadas (*stakeholders*); evolução de software, essas são atividades que referem-se a como software deve evoluir para atender às mudanças das demandas das partes interessadas (*stakeholders*). Algumas dessas atividades podem ser decompostas em atividades mais simples. Por exemplo, atividades para validação de requisitos, projeto (*design*) de arquitetura de software ou teste unitário de software (*unit testing*) [1].

A engenharia de software enfoca a aplicação de abordagem sistemática, disciplinada e quantificável ao desenvolvimento, operação e manutenção de software. Processos variados são relevantes em engenharia de software. Esses processos podem englobar, por exemplo, atividades relativas à especificação, ao projeto (*design*), à implementação e à validação de software. Alguns desses processos são complexos e podem ser decompostos em processos mais simples. Entre os processos em engenharia de software, tem-se processo de definição de arquitetura de software. Processo de definição de arquitetura de software é relevante processo em engenharia de software. Esse processo engloba atividades relativas à definição, avaliação, descrição, manutenção, certificação e implementação de arquitetura de software. O processo de definição de arquitetura de software está relacionado com a compreensão de como o software deve ser organizado e estruturado. O processo de definição de arquitetura de software é relevante processo no desenvolvimento de software, uma vez que a arquitetura do software pode, por exemplo, afetar atributos como desempenho, robustez, capacidade de distribuição e manutenibilidade de software [1] [2].

Considerando a relevância de processo de definição de arquitetura de software no

contexto de desenvolvimento de software, este trabalho enfoca a configuração de processo de desenvolvimento de software com elementos típicos de processo de definição de arquitetura de software, e o uso de processo de desenvolvimento de software configurado. Particularmente, neste trabalho são enfocadas atividades relacionadas à descrição e à avaliação de arquitetura de software. Ao longo deste trabalho, diversas fontes de informação serão acessadas, particularmente normas (*standards*) acerca de descrição de arquitetura de software e fontes contendo descrições de métodos para a avaliação de arquitetura de software.

1.1 Objetivo geral

Configurar processo de desenvolvimento de software com elementos de processo de definição de arquitetura de software e exemplificar uso de processo de desenvolvimento de software configurado.

1.2 Objetivos específicos

Seguem objetivos específicos do trabalho:

- Descrever conceitos acerca de arquitetura de software;
- Descrever elementos de processo de definição de arquitetura de software;
- Descrever métodos para avaliação de arquitetura de software;
- Configurar processo de desenvolvimento de software com elementos de processo de definição de arquitetura de software;
- Exemplificar o uso de processo de desenvolvimento de software configurado.

1.3 Motivações e justificativas

O processo de definição de arquitetura de software consiste de relevante processo em desenvolvimento de software. Processo criativo que engloba atividades para definição, avaliação, descrição, manutenção, certificação e implementação de arquitetura de software com o intuito de satisfazer requisitos do software em questão. Processo de definição de arquitetura de software pode promover, por exemplo, comunicação com as partes interessadas (*stakeholders*), análise e reúso de software. Considerando-se a relevância de processo de definição de arquitetura de software, é importante promover o acesso à informação acerca de atividades integrantes desse processo. Pode-se promover o

acesso a essa informação, por exemplo, disponibilizando-a em descrição de processo de desenvolvimento de software [1].

1.4 Estrutura do documento

Esta monografia está dividida em fundamentação teórica, exemplo de configuração de processo de desenvolvimento de software, exemplo de uso de processo de desenvolvimento de software configurado, análise de resultados e conclusão. A fundamentação teórica é composta por três capítulos e resultou de acesso a diversas fontes de informação. A fundamentação teórica é apresentada nos capítulos 2, 3 e 4. O capítulo 2 aborda arquitetura de software. Entre os assuntos abordados, é possível destacar arquitetura de software, descrição de arquitetura de software, usos para descrição de arquitetura de software, ponto de vista e visão de arquitetura de software, práticas para descrição de arquitetura de software e arcabouços de arquitetura. O capítulo 3 aborda processo de definição de arquitetura de software. Entre os assuntos abordados, é possível destacar 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. O capítulo 4 aborda avaliação de arquitetura de software. Entre os assuntos abordados, é possível destacar avaliação de arquitetura de software, métodos para avaliação de arquitetura de software, *Software Architecture Comparison Analysis Method (SACAM)*, *Architecture Tradeoff Analysis Method (ATAM)* e o *Software Architecture Analysis Method (SAAM)*. Exemplo de configuração de processo de desenvolvimento de software e exemplo de uso de processo de desenvolvimento de software configurado são descritos em dois capítulos. O capítulo 5 descreve exemplo de configuração de processo de desenvolvimento de software. O capítulo 6 descreve exemplo de uso de processo de desenvolvimento de software configurado. Por fim, considerações finais e sugestão de trabalho futuro estão no capítulo 7.

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 de 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, papel de desenvolvedor ou usuário [3] [4].

Para desenvolver um sistema de software, um processo é seguido, sendo esse o processo de desenvolvimento de sistema de software. Nesse processo, existem diversas atividades que devem ser realizadas. Por exemplo, atividades responsáveis por projeto de software (*software design*). Algumas dessas atividades são responsáveis por projeto de arquitetura de software (*architectural design*) [1]. Existem diversas definições para o termo arquitetura de software (*software architecture*). Segundo definição em [5], arquitetura de software corresponde à estrutura de sistema de 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 [5] [6].

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 [5] [6].

Segundo definição em [5], 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 [5] [6].

Uma outra definição para o termo arquitetura de software é encontrada em [7]. 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 [6] [7].

No contexto deste trabalho, é adotada a definição apresentada em [8]. 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” [8].

Em desenvolvimento de sistema de software, a definição da 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 [6] [8].

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 [8] [9].

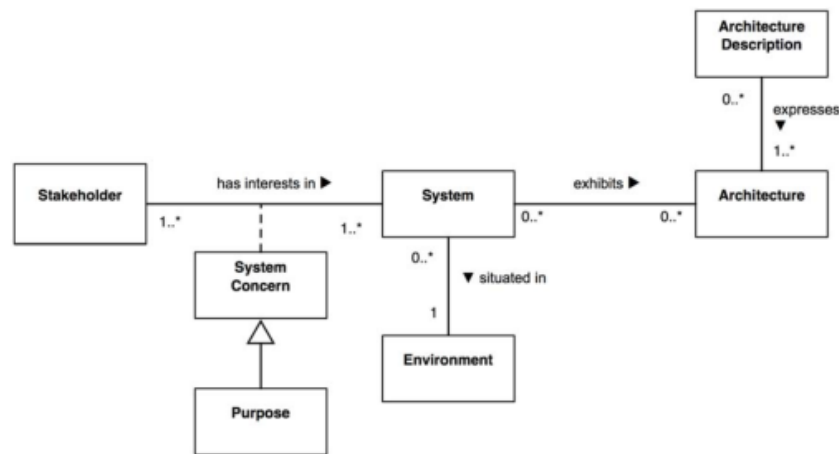


Figura 2.1: Contexto da descrição da arquitetura de software (Fonte: [9]).

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. Com isso, o sistema exibe a arquitetura que é expressa por meio da descrição de arquitetura [8] [9].

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 [10].

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, como 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 [10].

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 [9].

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 [9].

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 [8] [9].

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 [8].

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 [9] [11]. 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 [11].

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 [12].

Por fim, tem-se que diversos *frameworks* estão disponíveis para uso no desenvolvimento de software. Alguns desses frameworks podem tornar o processo de compreender e detalhar a arquitetura de software mais simples para o arquiteto. O uso de alguns desses frameworks pode promover a padronização de conteúdos e de processos relacionados à arquitetura de software [9].

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 [13].

- 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 [13].

3.2 Processos em ciclo de vida de software

Segundo [1], processo inclui atividades que envolvem mudanças. Segundo [3], 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 [13]. Segundo essa definição, processo é um conjunto integrado de atividades que transforma entradas em saídas desejadas. Segundo [13], 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 [14].

- 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

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 [14].

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 [14].

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 [14].

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 [14] :

- 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 elementos dos seguintes: *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*) [15].

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 para o sistema de software. Para essa comparação ser realizada, as seguintes atividades são executadas [16]:

- 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, tem-se as seguintes [16]:

- 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 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 [16]:

- 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 solicitadas. 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.

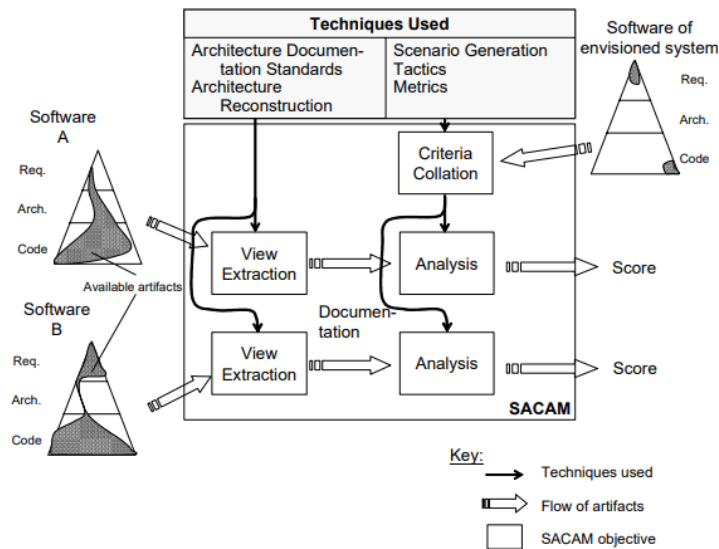


Figura 4.1: Técnicas de arquitetura utilizadas pelo SACAM (Fonte: [16]).

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 [16].

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 [17].

- 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 (*stakeholders*) 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 [17].

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 [15] [18].

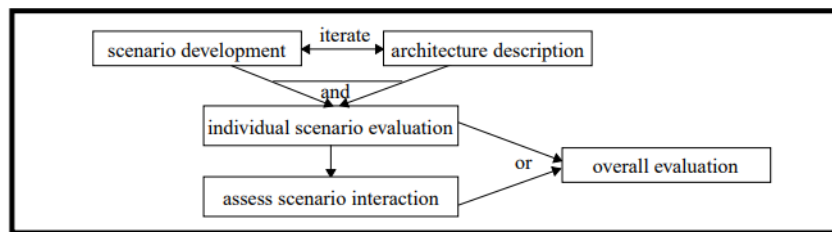


Figura 4.2: Etapas integrantes do SAAM (Fonte: [18]).

Na Figura 4.2 são representadas as seguintes etapas do SAAM [18]:

- 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 [15].

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 [18].

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 atendidas pelo sistema de software. O cenário deve representar as tarefas relevantes do sistema de software para diferentes usuários [18].

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 [18].

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 [18].

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 [18].

Capítulo 5

Exemplo de configuração de processo de desenvolvimento

Este capítulo aborda exemplo de configuração de processo de desenvolvimento. Entre os elementos deste capítulo, é possível destacar os seguintes: open unified process, elementos relacionados à arquitetura de software, configuração do processo de desenvolvimento, prática para descrição de arquitetura, passos para análise de arquitetura e artefatos construídos.

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 [19].

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 [19]. 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: Concepção (*Inception*), Elaboração (*Elaboration*), Construção (*Construction*) e Transição (*Transition*). A fase denominada Concepção tem o propósito de alcançar concordância

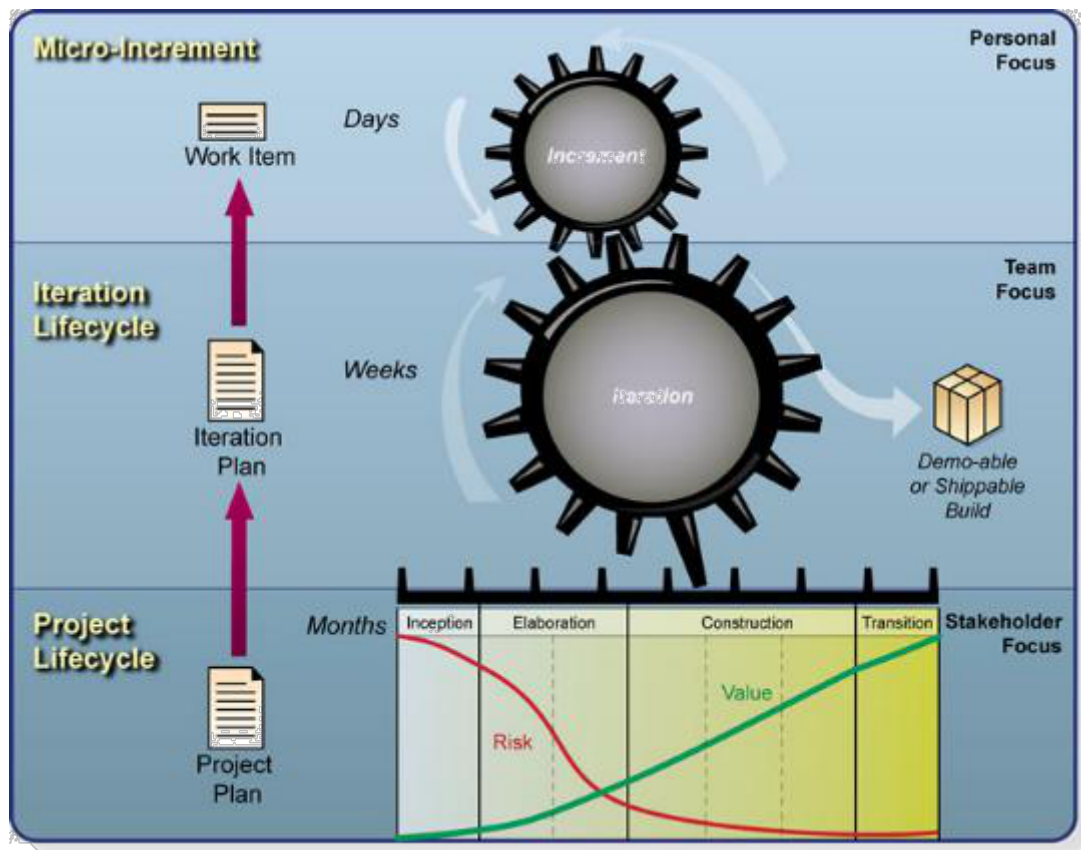


Figura 5.1: Ciclo de vida de projeto segundo o OpenUP (Fonte: [19]).

entre os interessados (*stakeholders*) sobre os objetivos do ciclo de vida do projeto. A fase denominada 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 denominada Construção tem o propósito de completar o desenvolvimento do sistema com base na arquitetura (*baseline architecture*) definida. Por fim, a fase denominada 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*) [19].

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 [19].

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 [19]:

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

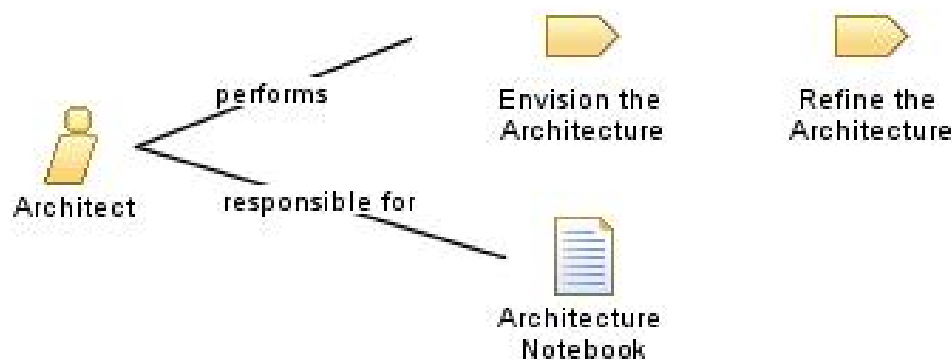


Figura 5.2: Tarefas e artefatos relacionados a arquiteto (Fonte: [19]).

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 comunicação entre equipes. A seguir, são relacionados passos nessa tarefa segundo o OpenUP [19]:

- 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 [19]. A seguir, são relacionados passos nessa tarefa segundo o OpenUP [19]:

- 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 esses mecanismos serem aplicados. A seguinte informação pode ser registrada nesse artefato [19]:

- 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 [8].

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 [8]:

- 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 [8]:

- 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 [8]:

- Propósitos ou missões do sistema;

- Adequação do sistema no atendimento das suas missões;
- Riscos de desenvolvimento;
- Riscos de operação 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 [8]:

- 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 [8]:

- 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 [8].

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 [15] [18]:

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

5.3 Artefatos construídos

Diante do que foi abordado até o momento, foram construídos artefatos novos, seguindo como referência a norma *IEEE 1471-2000 - IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. Sendo assim, nesses artefatos foram abordadas a prática para descrição da arquitetura conforme especificado no capítulo 5.2.1 e os passos para análise de arquitetura, conforme especificado no capítulo 5.2.1.

Capítulo 6

Exemplo de uso de processo de desenvolvimento

Este capítulo aborda exemplo de uso de processo de desenvolvimento anteriormente configurado. Entre os elementos que integram este capítulo, é possível destacar os seguintes: ciclo de vida do projeto, requisitos funcionais, cenários funcionais, arquitetura do software, projeto da interface com o usuário, projeto do banco de dados e ferramentas de desenvolvimento.

6.1 Ciclo de vida do projeto

O ciclo de vida adotado no projeto foi embasado no ciclo de vida descrito no processo de desenvolvimento OpenUP [19]. Esse ciclo de vida é composto pelas seguintes fases: Concepção, Elaboração, Construção e Transição. O fluxo de trabalho nesse ciclo de vida é apresentado na Figura 6.1. Em cada fase do ciclo de vida existe um processo proposto. Esse projeto seguiu processos da Concepção e Elaboração.

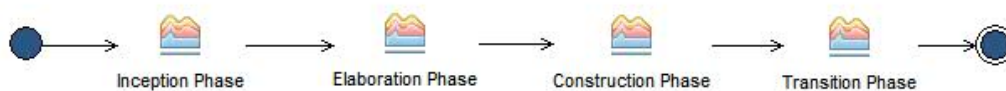


Figura 6.1: Fases de trabalho em ciclo de vida (Fonte: [19]).

6.2 Fase de concepção

A fase denominada Concepção é caracterizada por uma sequência de atividades que pode ser seguida no projeto. Em decorrência dessas atividades, são realizadas tarefas e são criados artefatos. O fluxo de trabalho do proposto no OpenUP se encontra na Figura 6.2.

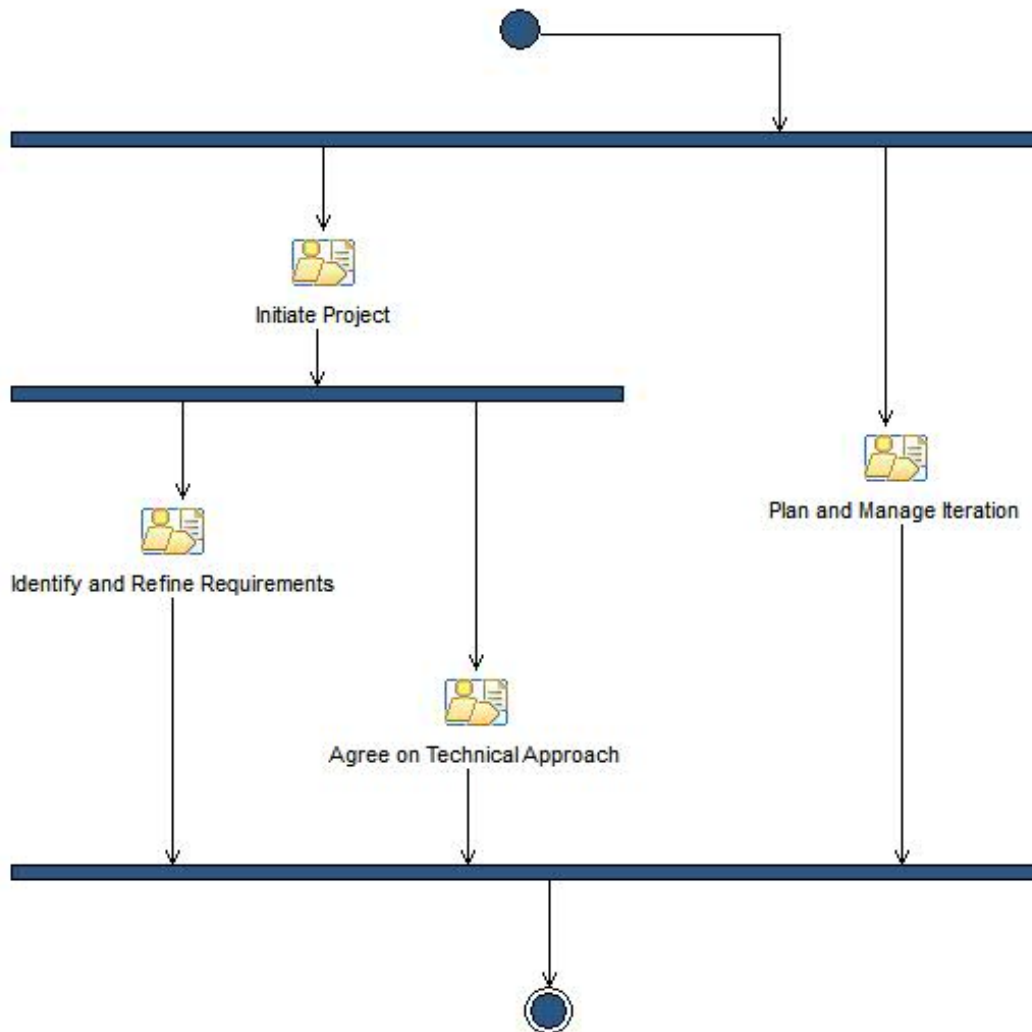


Figura 6.2: Fluxo de trabalho fase Concepção (Fonte: [19]).

As atividades que foram executadas na fase Concepção foram as seguintes:

- Iniciar projeto;
- Planejar e gerenciar iteração;

- Identificar e refinar requisitos;
- Concordar com abordagem técnica.

6.2.1 Atividade Iniciar Projeto

A atividade Iniciar projeto tem como objetivo iniciar o projeto e obter um acordo das partes interessadas no projeto em relação ao escopo do projeto. Nessa atividade foram realizadas tarefas e construídos artefatos. Elementos associados a essa atividade são apresentados na Figura 6.3. A seguir, tarefas e artefatos associados à atividade:

- Tarefa: Desenvolver visão técnica;
 - Artefato: Documento de visão.
- Tarefa: Planejar projeto.
 - Artefato: Plano de projeto.

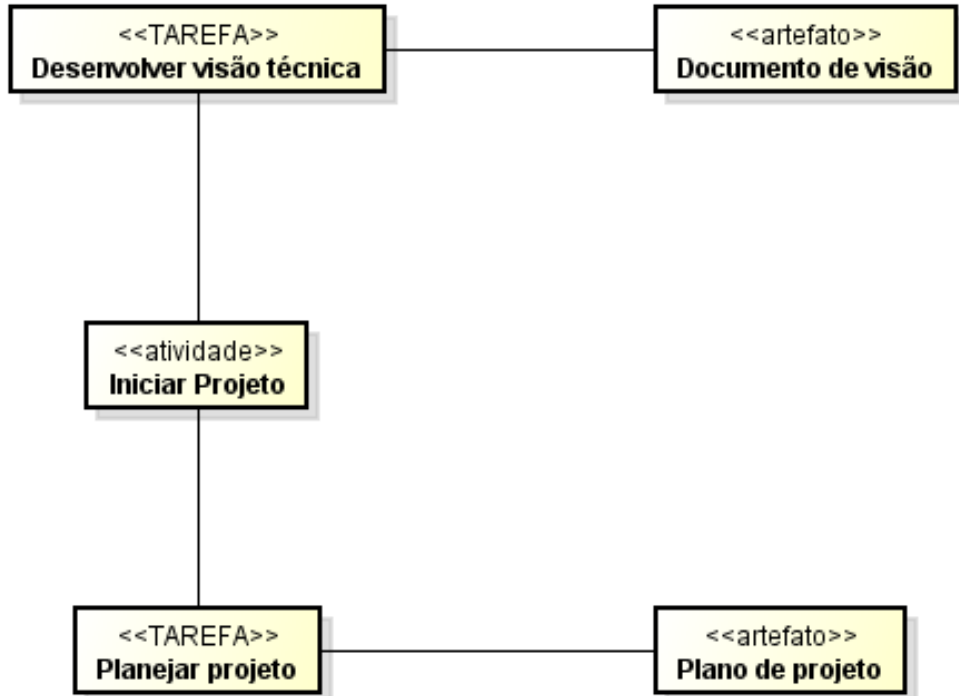


Figura 6.3: Elementos da atividade Iniciar projeto (Fonte: [19]).

A tarefa Desenvolver visão técnica tem como objetivo definir a visão para o sistema. São descritos problemas e recursos com base nas solicitações das partes interessadas. Nessa tarefa foi construído o artefato Documento de visão. Esse artefato define a visão das partes interessadas no sistema e provê informação acerca da solução técnica a ser desenvolvida. Essa definição é especificada em termos das principais necessidades e características requeridas por partes interessadas em relação ao sistema de software. O artefato Documento de visão contém um esboço dos requisitos básicos do sistema de software. O documento produzido adotou modelo (*template*) sugerido no OpenUP. Já a tarefa Planejar projeto é uma tarefa colaborativa que descreve o acordo inicial sobre como o projeto atingirá seus objetivos. O artefato resultante fornece uma visão geral resumida do planejamento do projeto, contém informações necessárias para gerenciar o projeto. Como parte desse plano, tem-se informação acerca de iterações do projeto e dos seus objetivos [19].

6.2.2 Atividade Planejar e gerenciar iteração

A atividade Planejar e gerenciar iteração permite que os membros da equipe se inscrevam em tarefas de desenvolvimento. De forma que pode ser realizado o monitoramento e a comunicação referente ao estado do projeto às partes interessadas. Essa atividade tem como objetivo identificar e tratar exceções e problemas. Nessa atividade foi realizada tarefa e construído artefato.



Figura 6.4: Elementos da atividade Planejar e gerenciar iteração (Fonte: [19]).

Elementos dessa atividade são apresentados na Figura 6.10. A seguir, tarefa e artefato associados à atividade:

- Tarefa: Planejar iteração.
 - Artefato: Plano de iteração.

A tarefa Planejar iteração tem como objetivo identificar o próximo incremento da capacidade do sistema de software e criar um plano refinado a fim de atingir essa capacidade em uma iteração. Essa tarefa é repetida para cada iteração em uma entrega. Isso permite que a equipe aumente a precisão das estimativas para uma iteração, pois mais detalhes são conhecidos ao longo do projeto. Como artefato construído, tem-se o Plano de iteração. Esse artefato descreve objetivos, atribuições de trabalho e critérios de avaliação para a iteração [19].

6.2.3 Atividade Identificar e refinar requisitos

A atividade Identificar e refinar requisitos tem como objetivo detalhar um conjunto de requisitos do sistema, detalhar serviços que o sistema de software deve prover, especificando-os por meio de casos de uso ou de cenários referentes ao sistema de software. Nessa atividade foi realizada tarefa e construídos artefatos. Elementos dessa atividade são apresentados na Figura 6.5. A seguir, tarefa e artefatos associados à atividade:

- Tarefa: Identificar e esboçar requisitos;
 - Artefato: Glossário;
 - Artefato: Requisitos com abrangência de sistema;
 - Artefato: Caso de uso;
 - Artefato: Modelo de caso de uso.

A tarefa Identificar e esboçar requisitos descreve como identificar e delinear os requisitos do sistema para que o escopo do trabalho possa ser definido. Um objetivo dessa tarefa é identificar os requisitos funcionais e não funcionais do sistema de software. Esses requisitos formam a base da comunicação e do acordo entre as partes interessadas e a equipe de desenvolvimento sobre o que o sistema deve fazer para satisfazer às partes interessadas. O artefato Glossário define termos usados no projeto. O artefato Requisitos com abrangência de sistema captura atributos de qualidade e restrições que têm escopo de todo o sistema. Os artefatos Modelo de caso de uso e Caso de uso têm como objetivo capturar comportamentos do sistema de acordo com ações dos usuários do sistema [19].

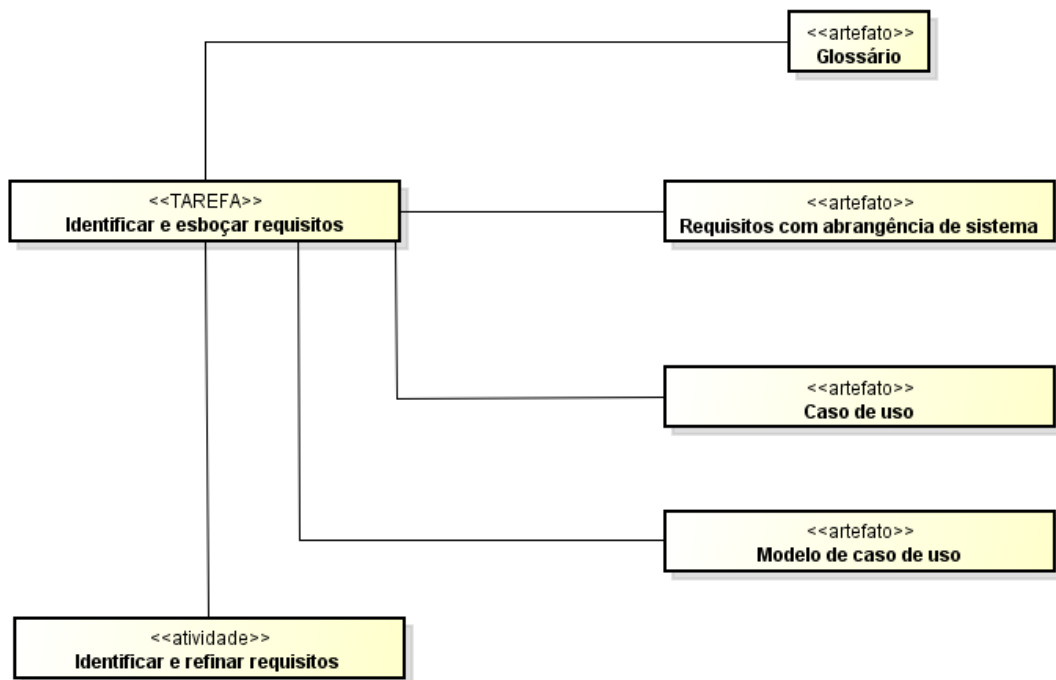


Figura 6.5: Elementos da atividade Identificar e refinar requisitos (Fonte: [19]).

6.2.4 Atividade Concordar com abordagem técnica

A atividade Concordar com abordagem técnica tem como objetivo chegar a um acordo sobre uma abordagem técnica que seja viável para o desenvolvimento do software. Nessa atividade foi realizada tarefa e construído artefato. Elementos dessa atividade são apresentados na Figura 6.6. A seguir, tarefa e artefato associados à atividade:

- Tarefa: Visualizar arquitetura.
 - Artefato: Caderno de arquitetura.

A tarefa Visualizar arquitetura tem como objetivo desenvolver a visão da arquitetura por meio da análise dos requisitos que são significativos para o sistema de software, analisando assim os requisitos funcionais e não funcionais do sistema a fim de identificar restrições, decisões e objetivos da arquitetura. Ao identificar esses elementos é possível fornecer uma orientação e direcionamento para a equipe iniciar o desenvolvimento. O artefato Caderno de arquitetura descreve lógica, suposições, explicações e implicações de decisões tomadas ao projetar a arquitetura [19].

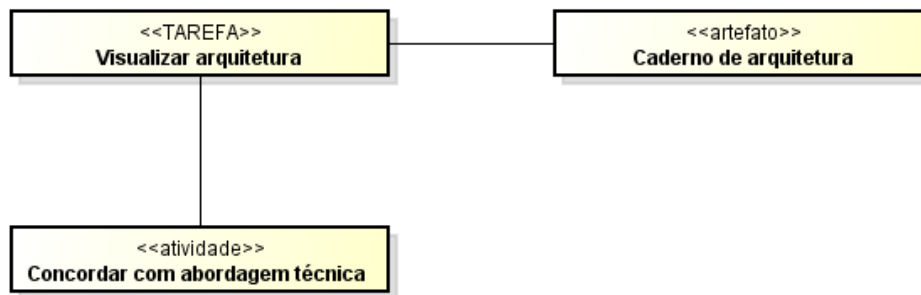


Figura 6.6: Elementos da atividade Concordar com abordagem técnica (Fonte: [19]).

6.3 Fase de elaboração

A fase denominada Elaboração é caracterizada por uma sequência de atividades que pode ser seguida no projeto. Em decorrência dessas atividades, são realizadas tarefas e são criados artefatos. O fluxo de trabalho proposto no OpenUP se encontra na Figura 6.7.

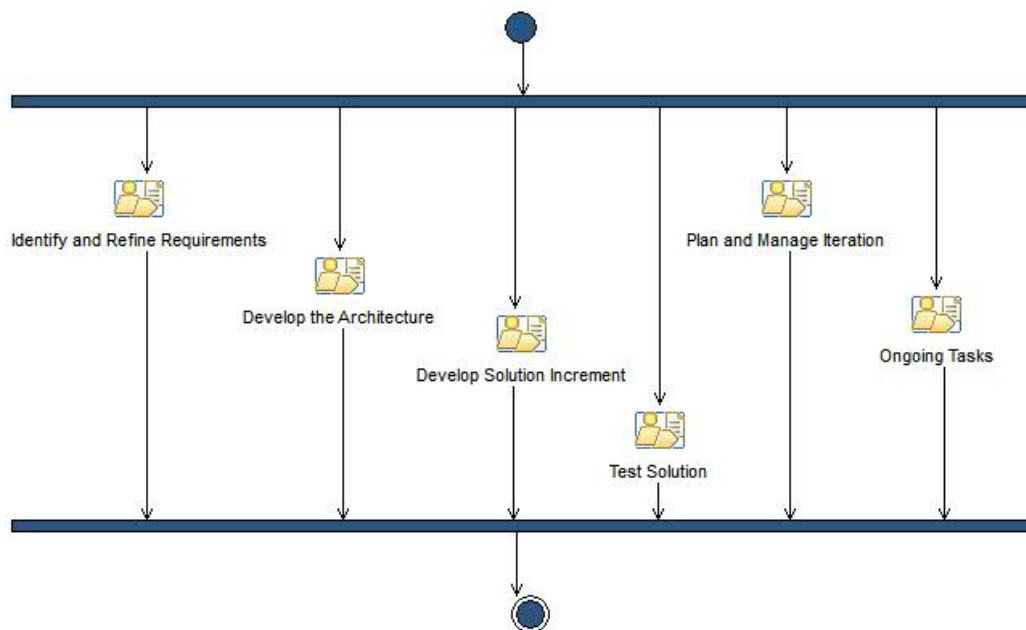


Figura 6.7: Fluxo de trabalho da fase Elaboração. (Fonte: [19]).

As seguintes atividades da fase Elaboração foram executadas nesse projeto:

- Desenvolver a arquitetura;

- Desenvolver incremento da solução;
- Planejar e gerenciar iteração.

6.3.1 Atividade Desenvolver a arquitetura

A atividade Desenvolver a arquitetura tem como objetivo desenvolver os requisitos que são considerados significativos mediante a arquitetura do sistema de software. Nessa atividade foi realizada tarefa e construído artefato. Elementos dessa atividade são apresentados na Figura 6.8. A seguir, tarefa e artefato associados à atividade:

- Tarefa: Refinar a arquitetura.
 - Artefato: Caderno de arquitetura.

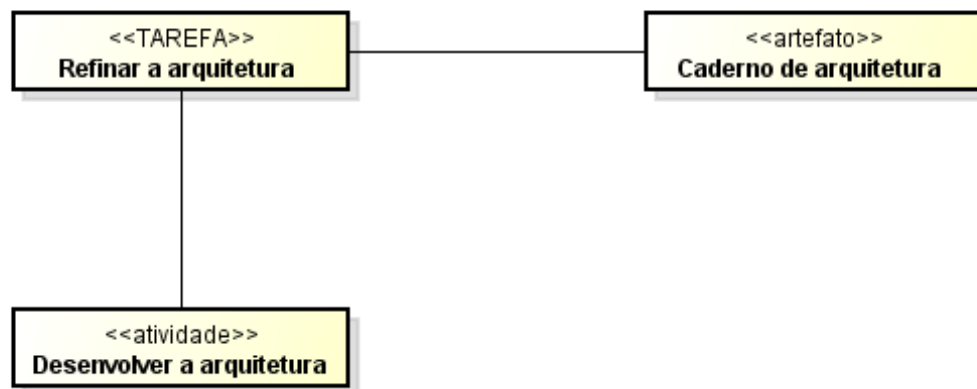


Figura 6.8: Elementos da atividade Desenvolver a arquitetura (Fonte: [19]).

A tarefa Refinar a arquitetura permite que a arquitetura seja refinada a um nível apropriado de detalhe para dar suporte ao desenvolvimento. O Caderno de arquitetura desenvolvido na fase Concepção foi refinado na fase Elaboração aplicando-se proposta presente neste trabalho.

6.3.2 Atividade Desenvolver incremento da solução

A atividade Desenvolver incremento da solução consiste em realizar projeto (*design*), implementação, teste e integração da solução para os requisitos do sistema do software. Nessa atividade foi realizada tarefa e construído artefato. Elementos dessa atividade são apresentados na Figura 6.9. A seguir, tarefa e artefato associados à atividade:

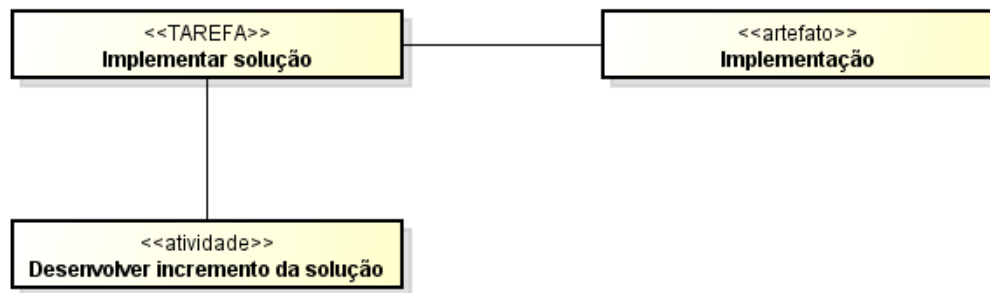


Figura 6.9: Elementos da atividade Desenvolver incremento da solução (Fonte: [19]).

- Tarefa: Implementar solução.
 - Artefato: Implementação.

A tarefa Implementar solução tem como propósito implementar o código fonte a fim de prover novas funcionalidades para o sistema de software ou corrigir defeitos que foram encontrados. Como artefato construído, tem-se a Implementação composta por códigos desenvolvidos, arquivos de dados e arquivos de suporte utilizados no desenvolvimento.

6.3.3 Atividade Planejar e gerenciar iteração

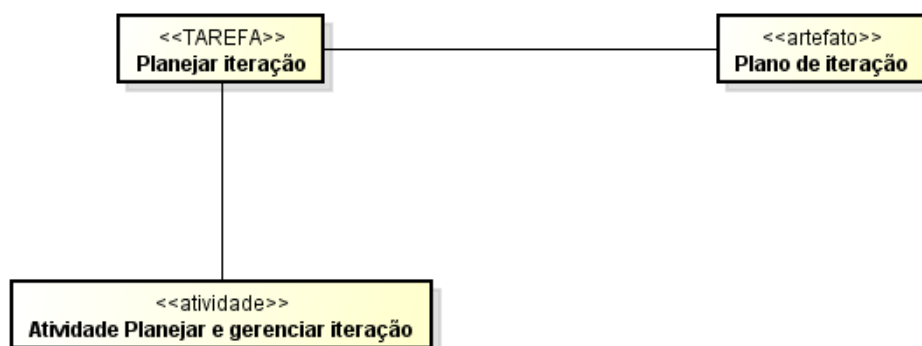


Figura 6.10: Elementos da atividade Planejar e gerenciar iteração (Fonte: [19]).

A atividade Planejar e gerenciar iteração consiste de atividade que promove comunicação em relação às atividades que estão sendo executadas. Além de permitir que o estado do projeto seja atualizado para as partes interessadas. Como produto dessa atividade tem-se o artefato Plano de iteração, que foi criado na fase Concepção e que, conforme evoluiu o projeto, foi implementado. Elementos dessa atividade são apresentados na Figura 6.10. A seguir, tarefa e artefato associados à atividade:

- Tarefa: Planejar iteração.
 - Artefato: Plano de iteração.

6.4 Requisitos funcionais

O sistema de software desenvolvido com o propósito de exemplificar o uso do processo de desenvolvimento configurado no capítulo anterior, tem como objetivo permitir que sejam realizadas consultas acerca de medicamentos disponibilizados ou não em posto de saúde; além de permitir o controle de estoque dos medicamentos em posto de saúde. A seguir, se encontra relação de nomes de atores que foram identificados:

- Cidadão;
- Gestor do posto de saúde.

Cidadão é usuário que necessita informação sobre medicamento que utiliza ou que precisa utilizar, e que deseja saber se esse medicamento é disponibilizado ou não pelo seu posto de saúde. Quem realiza a atualização de informação acerca de medicamento no posto de saúde é o gestor do posto de saúde, que através desse sistema controla estoque de medicamentos no posto de saúde no qual está alocado. A Figura 6.11 consiste de diagrama que mostra atores e relacionamentos entre os mesmos.

As funcionalidades correspondem aos requisitos funcionais do sistema. As funcionalidades serão relacionadas a seguir e depois descritas para melhor compreensão. As seguintes funcionalidades devem ser providas pelo sistema de software para ambos os atores: Autenticar usuário, Visualizar perfil. As seguintes funcionalidades devem ser providas pelo sistema de software para o ator Cidadão: Consultar medicamentos registrados no sistema, Visualizar o gestor responsável pelo seu posto de saúde. Por fim, as seguintes funcionalidades devem ser providas pelo sistema de software para o ator Gestor do posto de saúde: Realizar controle do estoque de medicamentos no posto de saúde (Cadastrar medicamento; Editar registro do medicamento; Apagar registro do medicamento), Solicitar reposição de medicamentos para o posto de saúde.

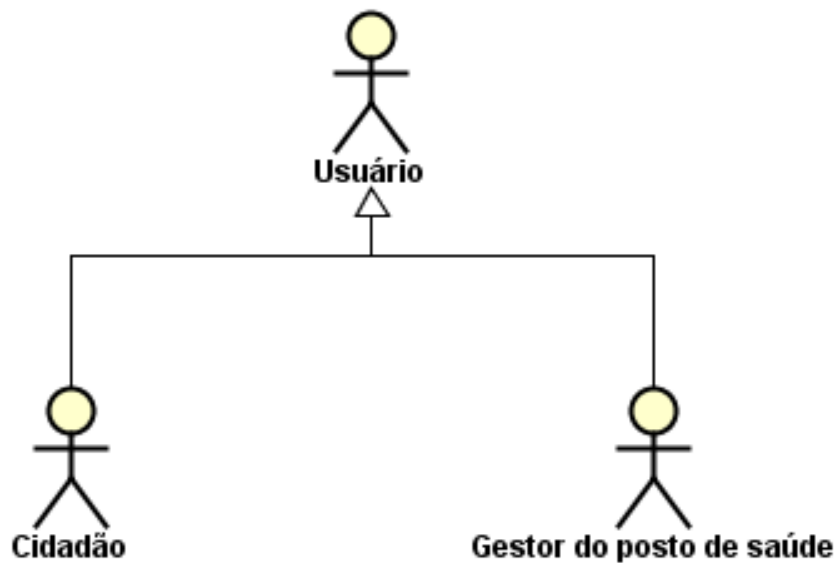


Figura 6.11: Atores e relacionamento entre atores.

6.4.1 Funcionalidade: Autenticar usuário

O usuário pode realizar login no sistema, desde que ele já esteja cadastrado no sistema. Ao se cadastrar, o usuário deve optar por ser cidadão ou gestor de posto de saúde. Dependendo do tipo de usuário, os acessos são distintos, visto que as funcionalidades providas pelo sistema dependem do tipo de usuário que acessa o sistema.

6.4.2 Funcionalidade: Visualizar perfil

O perfil de usuário depende do tipo de usuário cadastrado. Usuário cadastrado como cidadão possui perfil com acesso ao sistema de consulta de medicamentos, acesso ao seu perfil e a qual posto de saúde está registrado. O cidadão pode acessar lista de medicamentos que usa e consultar se o seu posto de saúde tem esse medicamento. O cidadão pode acessar seu perfil e visualizar dados pessoais, como nome, sobrenome e e-mail. Usuário cadastrado como gestor pode visualizar seu perfil. Seu perfil possibilita acessar sistema de gestão de estoque de medicamentos, sendo assim, ele pode atualizar o sistema de estoque do posto de saúde pelo qual é responsável. Na Figura 6.12 tem-se

diagrama de caso de uso contendo representações de atores e de relacionamentos entre atores e casos de uso.

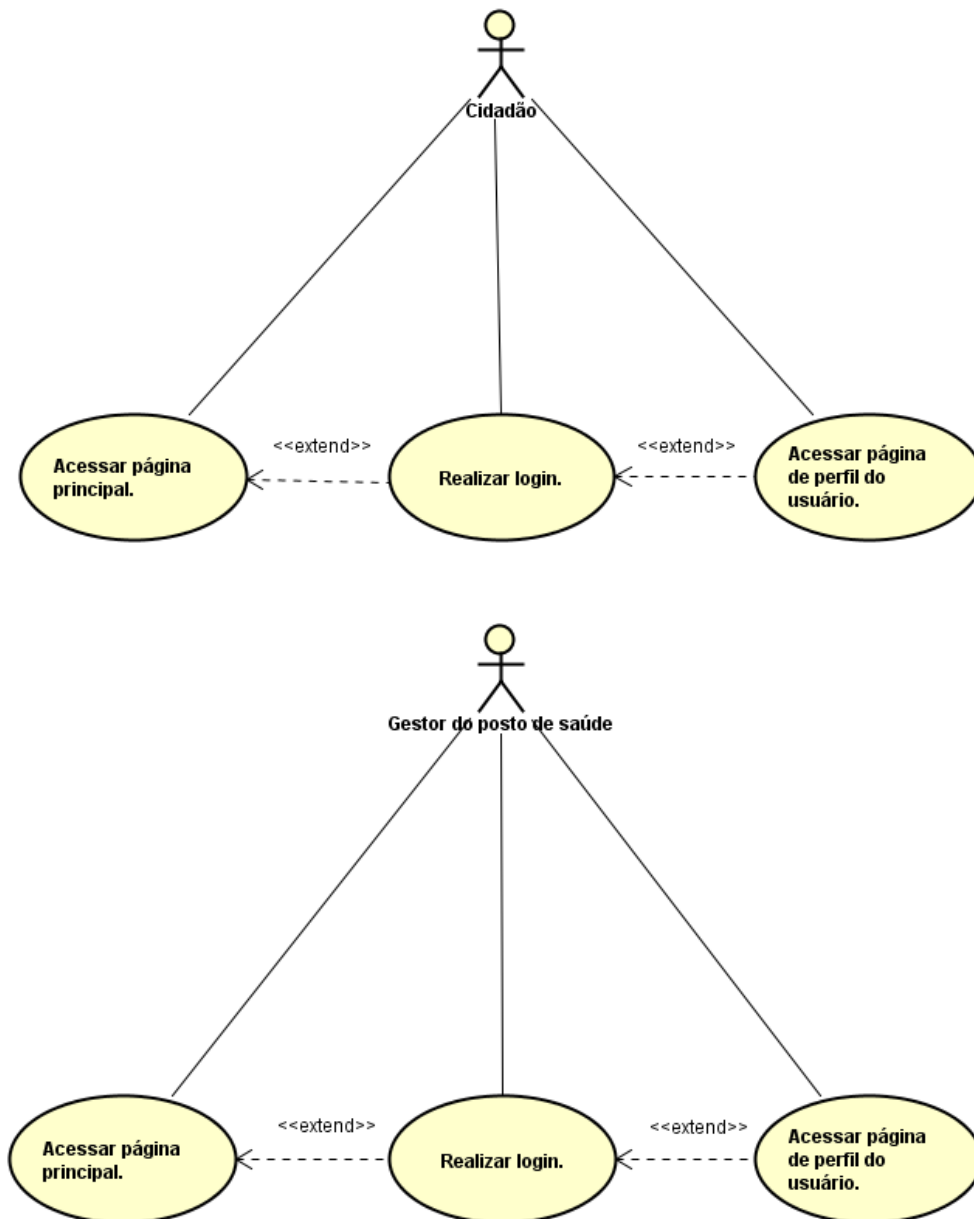


Figura 6.12: Diagrama de caso de uso para visualizar perfil.

6.4.3 Funcionalidade: Consultar medicamentos registrados no sistema

O cidadão pode consultar no sistema, se o medicamento que ele precisa está registrado no sistema. Em caso afirmativo, pode obter informação acerca do medicamento. Por

exemplo, para que é destinado o seu uso, e como deve ser feito o seu uso. Além disso, também é possível verificar se esse medicamento é disponibilizado pelo seu posto de saúde. Caso contrário, é listado se esse medicamento está disponibilizado na farmácia. Na Figura 6.13 tem-se diagrama de caso de uso contendo representações de atores e de relacionamentos entre atores e casos de uso.

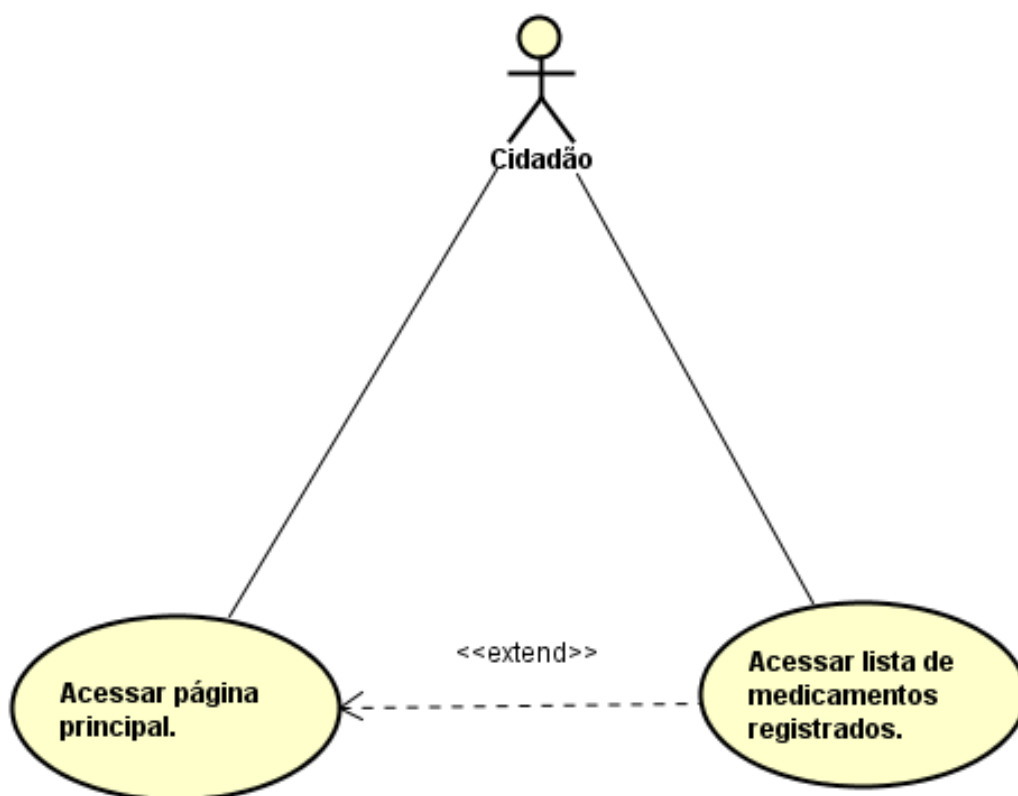


Figura 6.13: Diagrama de caso de uso para consultar medicamentos registrados no sistema.

6.4.4 Funcionalidade: Realizar controle do estoque de medicamentos

O gestor do posto de saúde deve realizar o controle do estoque dos medicamentos do posto de saúde sob sua responsabilidade. Para isso, o gestor pode cadastrar, editar e apagar o registro de medicamento no sistema. O sistema deve ser atualizado de forma que o sistema disponibilize informações atualizadas para os seus usuários. Na Figura 6.14 tem-se diagrama de caso de uso contendo representações de atores e de relacionamentos

entre atores e casos de uso, no contexto dessa funcionalidade. A seguir, tem-se descrições resumidas dos casos de uso.

Caso de uso: Cadastrar medicamento

Gestor do posto de saúde pode registrar medicamento, fornecendo as informações que forem solicitadas.

Atualizar informações de medicamento

Gestor do posto de saúde pode alterar registro de medicamento e este deve atualizar registro de medicamento.

Apagar medicamento

Caso o medicamento pare de ser disponibilizado pelo posto de saúde, gestor pode apagar registro de medicamento.

Solicitar reposição de medicamentos

Gestor de posto de saúde ao constatar que medicamento está em falta e precisa de reposição, ou quer solicitar novos medicamentos para posto de saúde, deve preencher formulário de solicitação de medicamentos. Isso permite que o posto de saúde controle o estoque de modo a prevenir a falta de medicamento no posto de saúde.

Na Figura 6.14 tem o diagrama de caso de uso do controle de estoque que deve ser realizado pelo gestor do posto de saúde.

6.5 Cenários Funcionais

Antes de realizar login, o cidadão deve se cadastrar. Uma vez realizado login, o cidadão tem acesso ao seu perfil e à consulta de medicamentos. A seguir, relação de áreas às quais o cidadão tem acesso ao usar o sistema de software: área de login, área de cadastro, área do perfil e área de busca dos medicamentos. A visão do gestor é diferente da visão do usuário, visto que ele controla o estoque de medicamentos do posto de saúde. Após realizar login, o gestor tem acesso à busca por medicamentos. Esse usuário também possui acesso ao controle de medicamentos. No controle de medicamentos, pode cadastrar, editar ou excluir medicamento. Cada ação tem uma tela e uma configuração a ser feita. Caso o estoque de medicamentos esteja com uma quantidade de medicamentos abaixo do necessário, ele pode solicitar medicamentos para o posto a fim de evitar que o posto fique com estoque abaixo do necessário. O gestor tem acesso a relatório mensal referente aos

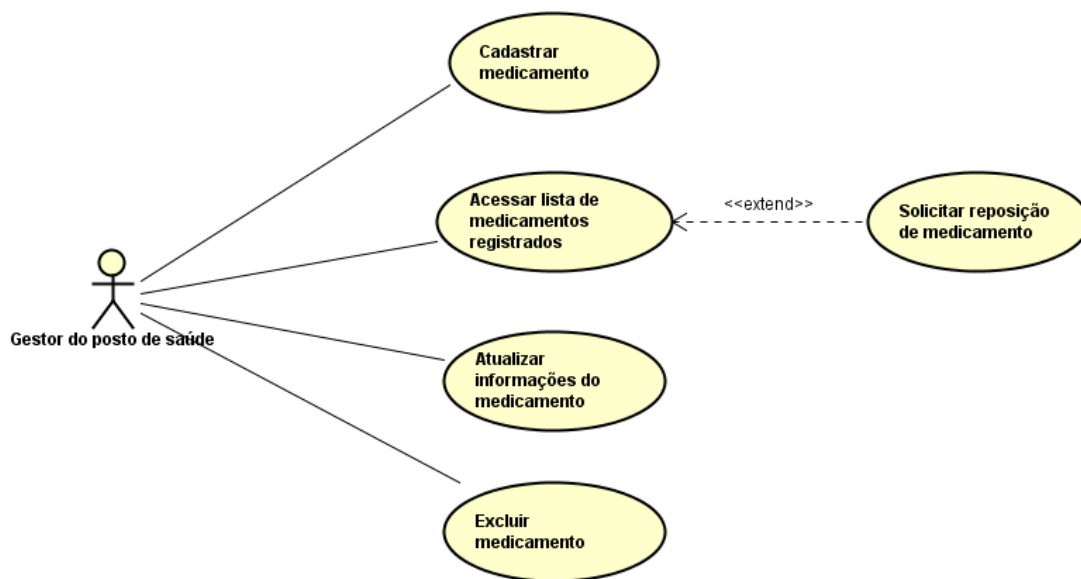


Figura 6.14: Diagrama de caso de uso para controle de estoque do medicamento.

medicamentos do posto de saúde. Além desse relatório mensal, também possui acesso a relatório gerencial do posto de saúde. A seguir, relação de áreas às quais o gestor tem acesso ao usar o sistema de software: área de login, área de cadastro, área do perfil, área de busca dos medicamentos, área de cadastro do medicamento, área de atualização do medicamento, área de deleção do medicamento, área de solicitação do medicamento, área de relatório mensal e área de relatório gerencial.

6.6 Arquitetura do software

6.7 Projeto da interface com o usuário

Para realizar login no sistema de software é necessário informar e-mail e senha. O login ocorrerá se o endereço de e-mail e a senha estiverem cadastrados. Para isso, foi construído código que realiza a validação do e-mail e da senha no banco de dados, vide Figura 6.15. Caso seja encontrado um registro no banco de dados, o usuário tem acesso ao sistema. Caso contrário, o usuário recebe mensagem alertando que foi encontrado erro. Dessa forma, apenas usuário cadastrado pode acessar determinadas informações no sistema de software. Para dar sequência ao desenvolvimento, após a integração com o banco de dados, foi desenvolvido modelo (template) para a tela acessada em seguida pelo

```

<?php
include('connection.php');

if (isset($_POST['email']) || isset($_POST['senha'])) {
    if (strlen($_POST['email']) == 0) {
        echo "Preencha seu e-mail";
    } else if (strlen($_POST['senha']) == 0) {
        echo "Preencha sua senha";
    } else {
        $email = $mysqli->real_escape_string($_POST['email']);
        $senha = $mysqli->real_escape_string($_POST['senha']);

        $sql_code = "SELECT * FROM cidadao WHERE email = '$email' AND senha = '$senha'";
        $sql_query = $mysqli->query($sql_code) or die("Falha na conexão:" . $mysqli->error);

        $qtd = $sql_query->num_rows;
        if ($qtd == 1) {
            $username = $sql_query->fetch_assoc();

            if (!isset($_SESSION)) {
                session_start();
            }

            $_SESSION['id'] = $username['id'];
            $_SESSION['email'] = $username['email'];
            $_SESSION['nome'] = $username['nome'];

            header("Location: index.php");

            echo "Usuário logado com sucesso";
        } else {
            echo "Usuário ou senha inválidos";
        }
    }
}
?>

```

Figura 6.15: Código para realizar validação de login.

usuário. Foi desenvolvido código em HyperText Markup Language (HTML) para a tela na Figura 6.16 e para o layout foi construído código em Cascading Style Sheets (CSS) utilizando o framework Bootstrap.

De acordo com a permissão que o usuário tenha, o mesmo pode acessar determinadas funcionalidades. Caso o usuário seja um cidadão, as funcionalidades providas para ele são diferentes das funcionalidades providas para o gestor. Sendo assim, para o cidadão tem-se a tela apresentada na Figura 6.18. Já para o gestor, como são providas diferentes funcionalidades, tem-se a tela apresentada na Figura 6.17.

Consulta ao estoque só pode ser realizada pelo gestor do projeto, dessa forma, para que o gestor tenha acesso a essa área, o gestor precisa estar cadastrado. O gestor pode realizar as seguintes ações: cadastramento de medicamento, atualização de medicamento e exclusão de medicamento, conforme ilustrado na Figura 6.19.

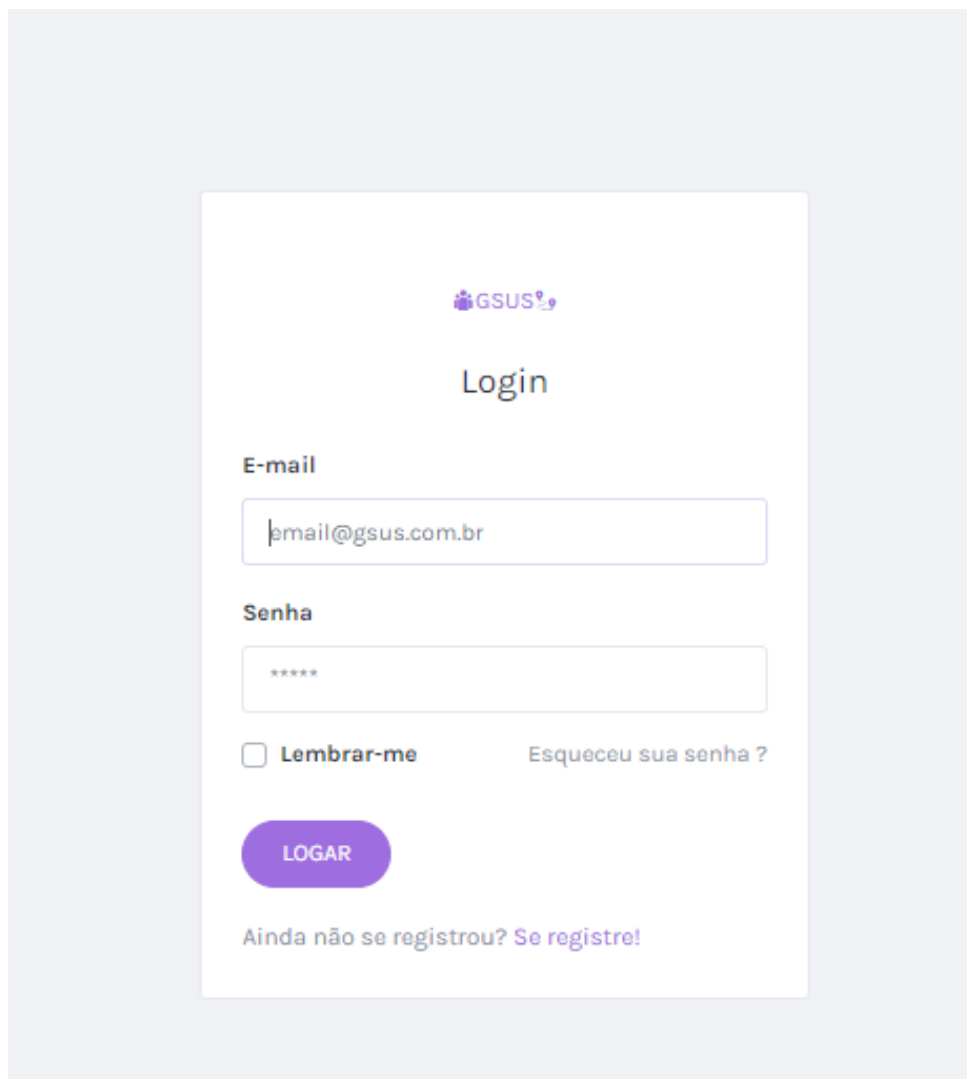


Figura 6.16: Tela de login implementada.

Na Figura 6.20 é apresentada tela para cadastramento de medicamento. Essa funcionalidade só é disponibilizada para o gestor do posto de saúde, logo a tela correspondente só aparece para usuário com permissão de gestor conforme ilustrado na Figura 6.17.

6.8 Projeto de banco de dados

Antes de realizar implementação, foi realizado projeto de banco de dados segundo o paradigma relacional. O modelo desenvolvido engloba entidades, atributos de entidades e relacionamentos entre entidades. Cada entidade é representada por tabela e tem os seus relacionamentos representados por ligações entre tabelas. O modelo desenvolvido

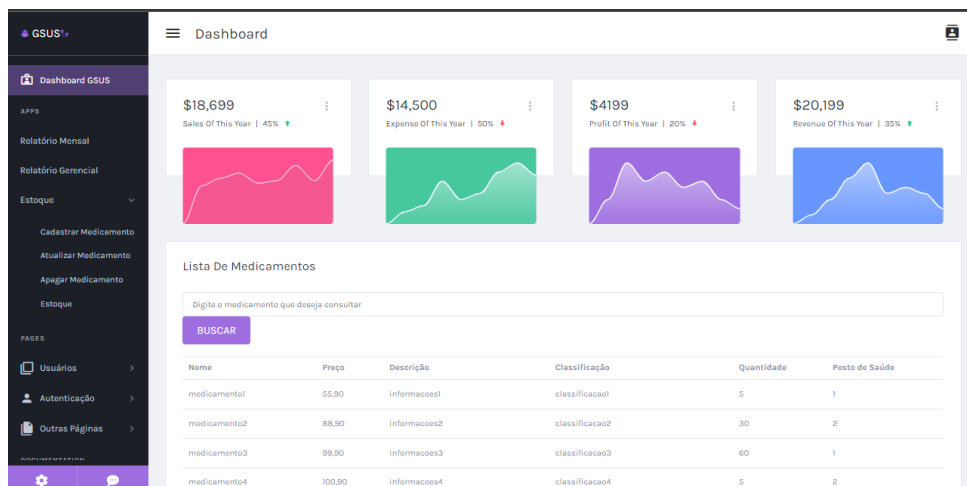


Figura 6.17: Tela principal para o gestor implementada.

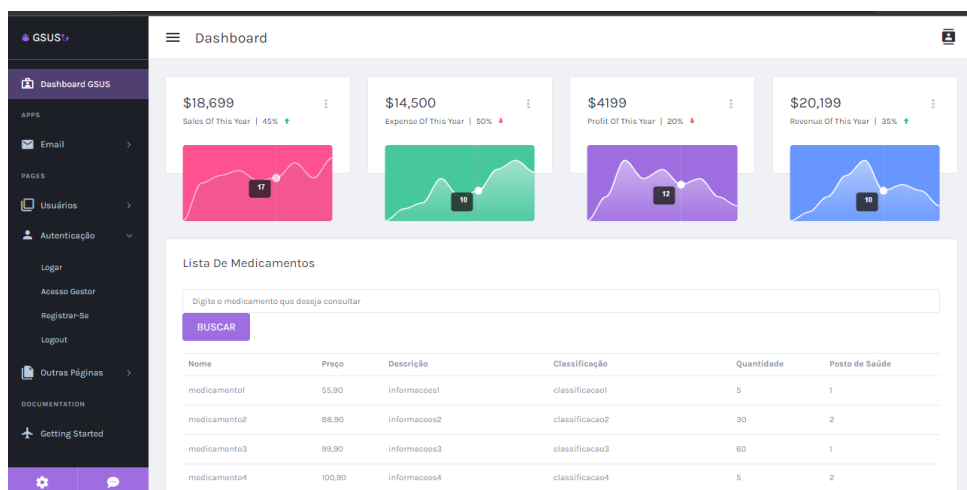


Figura 6.18: Tela principal para o cidadão implementada.

auxiliou na implementação do sistema de software. Esse modelo de banco de dados foi desenvolvido usando-se a ferramenta do MySQL Workbench, e se encontra na Figura 6.21.

6.9 Ferramentas de desenvolvimento

Para desenvolver o sistema de software, foi utilizada uma linguagem de programação popular, a linguagem PHP. As seguintes tecnologias foram também utilizadas: HyperText Markup Language (HTML), Javascript e Cascading Style Sheets (CSS) para desenvolver o frontend do sistema de software. Na construção da interface com o usuário, foi utilizado o framework Bootstrap. O desenvolvimento do sistema de software foi realizado utilizando o editor de texto Visual Studio Code. O sistema gerenciador de banco de dados (SGBD)

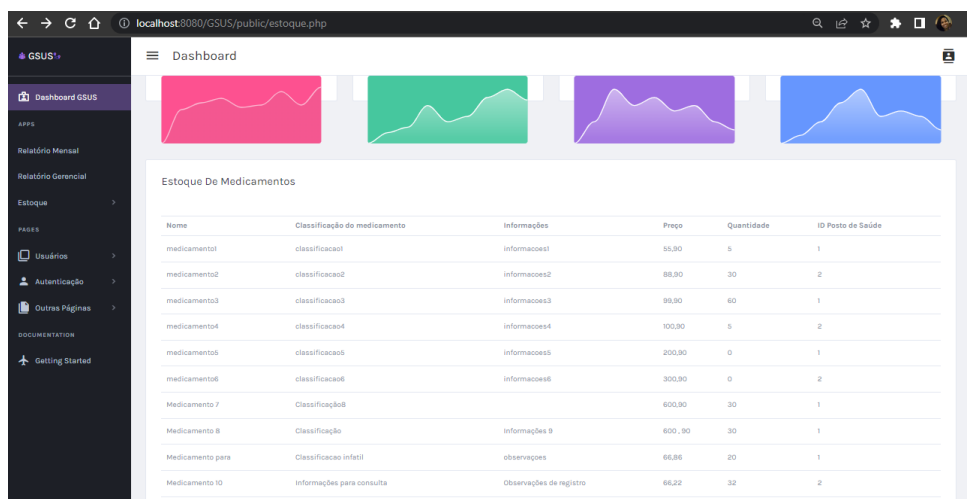


Figura 6.19: Tela de estoque de medicamentos.

The screenshot shows the GSUS Dashboard with the 'Cadastro o medicamento' form. The form includes the following fields and a button:

- ID Posto: 2
- ID medicamento: 3
- Nome:
- Classificação do medicamento:
- Informações:
- Preço: R\$: 55,60
- Quantidade: 60
- CADASTRAR MEDICAMENTO** button

Figura 6.20: Tela de cadastro de medicamentos.

utilizado foi o MySQL, acessado pela ferramenta Phpmyadmin. Para desenvolver o modelo relacional foi utilizado o MySQL Workbench. Para realizar o acesso dessas tecnologias foi configurado um servidor local, sendo ele o XAMPP. O XAMPP possui um pacote que permite realizar a integração das tecnologias que foram necessárias para desenvolver o sistema de software. Ao iniciar o XAMPP são iniciados os serviços do Apache e do MySQL. Tendo esses serviços iniciados para acessar o sistema de software desenvolvido, ele é acessado por meio do endereço <http://localhost:8080/GSUS>. O navegador web utilizado para realizar a navegação no sistema de software foi o Google Chrome. O projeto foi desenvolvido e executado em um notebook com sistema operacional Windows 10, contendo processador 5a geração Intel Core i5 com 8 GB de memória RAM. Para realizar o desenvolvimento desse sistema de software, foi necessário realizar a integração

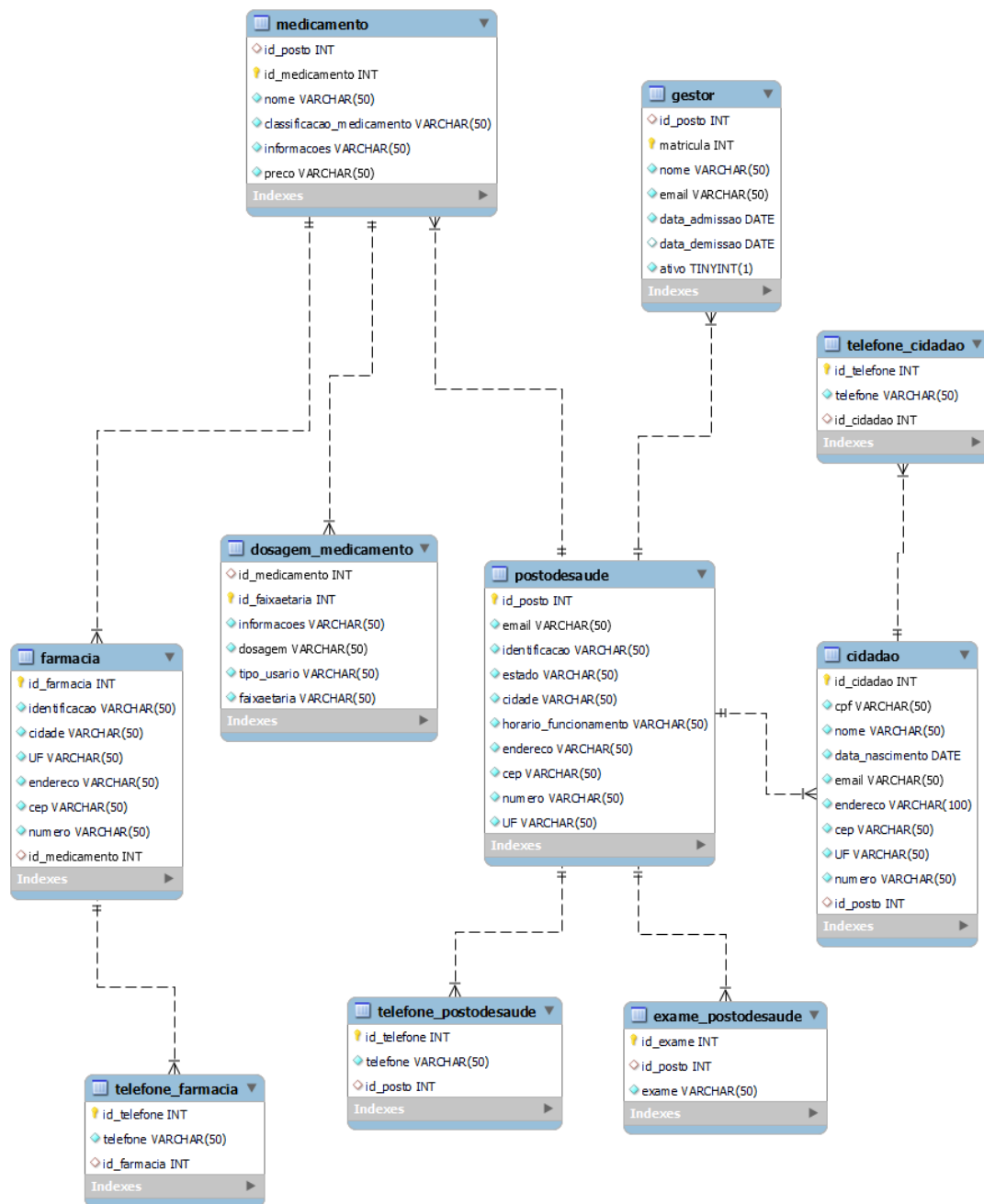
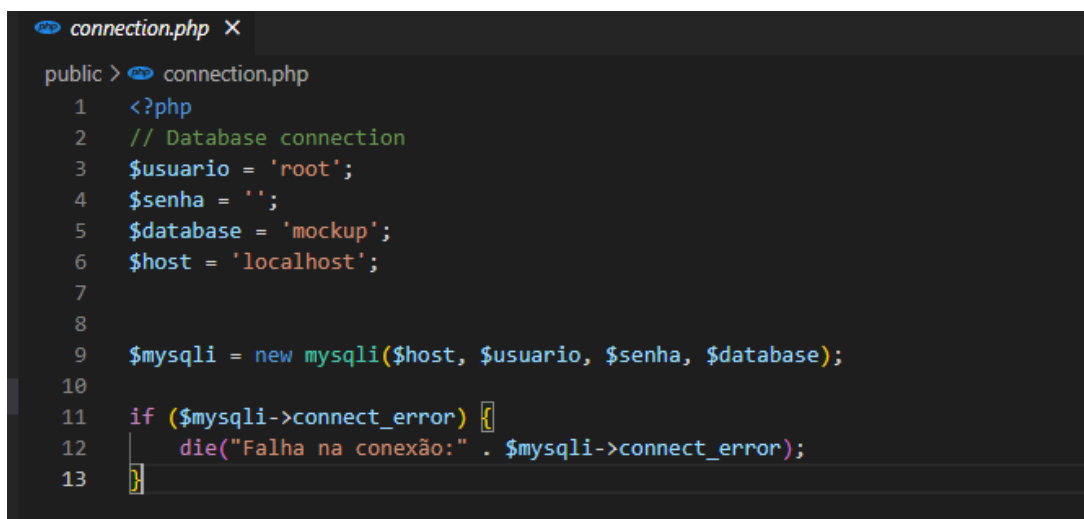


Figura 6.21: Modelo relacional do GSUS .

entre o Phpmyadmin e o sistema de software. Para isso, foi necessário primeiro realizar a configuração do banco de dados no Phpmyadmin e posteriormente foi criado um código em PHP a fim de realizar a conexão do banco com o sistema. Código para conexão com o banco de dados se encontra na Figura 6.22.

A screenshot of a code editor window titled 'connection.php'. The code is written in PHP and shows the configuration for a MySQL database connection. It includes variables for username, password, database name, and host, followed by the instantiation of a MySQLi object and an error handling block.

```
1  <?php
2  // Database connection
3  $usuario = 'root';
4  $senha = '';
5  $database = 'mockup';
6  $host = 'localhost';
7
8
9  $mysqli = new mysqli($host, $usuario, $senha, $database);
10
11  if ($mysqli->connect_error) {
12      die("Falha na conexão:" . $mysqli->connect_error);
13  }
```

Figura 6.22: Código para realizar conexão com banco de dados.

Capítulo 7

Conclusão

Este capítulo tem como intuito apresentar, tendo como base o referencial teórico e o exemplo desenvolvido, uma síntese dos resultados obtidos no trabalho. Para isso, ele foi dividido em duas partes. A primeira apresenta as considerações finais do estudo e a segunda apresenta sugestões para trabalhos futuros relacionados ao tema abordado.

7.1 Considerações Finais

O principal objetivo deste trabalho foi configurar processo de desenvolvimento de software utilizando elementos de processo de definição da arquitetura de software e exemplificar o uso do processo de desenvolvimento configurado. O processo configurado foi o OpenUp. A configuração do processo de desenvolvimento consistiu 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 do OpenUP, passos (*steps*) para analisar arquitetura de software por meio do método de análise de arquitetura Software Architecture Analysis Method (SAAM).

Com isso, pode ser concluído para este trabalho que realizar a especificação da arquitetura de software seguindo um processo é de grande relevância considerando o tempo de trabalho, visto que os cenários são explorados e analisados antes da sua implementação. Encontrar as possíveis falhas da integração permite ajustar a arquitetura de acordo com a necessidade que é avaliada seguindo o ponto de vista da parte interessada.

7.2 Trabalhos Futuros

Como sugestão de trabalho futuro relacionado a este projeto, tem-se implementar o sistema de software, seguindo o processo de desenvolvimento proposto, com mais desenvolvedores e implementar mais funcionalidades para o sistema de software. O

trabalho futuro pode também considerar ampliar o escopo. Por exemplo, implementar mais postos de saúde e maior população do banco de dados. Também pode ser ampliado o uso do sistema de software para mais usuários ao mesmo tempo. Uma outra sugestão para prosseguir este trabalho é melhorar o requisito segurança, visto que o banco de dados utilizado foi um banco de dados sem ênfase na segurança da informação. Com o aumento do escopo, melhorar a segurança seria relevante, visando assegurar a segurança do sistema frente a ataques maliciosos ou a acidentes.

Referências

- [1] Sommerville, Ian: *Engenharia de Software*. Pearson Education, São Paulo, 2011. 9.ed. 1, 3, 4, 12
- [2] *Guide to the software engineering body of knowledge (swebok v3)*. Disponível em: <https://www.computer.org/education/bodies-of-knowledge/software-engineering>, 2004. Acesso em: 12 de fevereiro de 2022. 1
- [3] *Software and systems engineering vocabulary*. Disponível em: https://pascal.computer.org/sev_display/index.action. Acesso em: 01 de março de 2022. 4, 12
- [4] ISO/IEC/IEEE: *Systems and software engineering — Vocabulary*. Institute of Electrical and Electronics Engineers, 2010. 4
- [5] BASS, Len ; CLEMENTS, Paul ; KAZMAN, Rick: *Software architecture in practice*. Addison-Wesley Professional, 2003. 4, 5
- [6] 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. 4, 5
- [7] CLEMENTS, Paul , et.al: *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2010. 5
- [8] ISO/IEC/IEEE: *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*. Institute of Electrical and Electronics Engineers, 2000. 5, 6, 8, 9, 27, 28, 29, 30
- [9] ISO/IEC/IEEE: *Systems and software engineering - Architecture description*. Institute of Electrical and Electronics Engineers, 2011. 6, 8, 9, 10
- [10] ISO/IEC/IEEE: *Systems and software engineering - Content of life-cycle information items (documentation)*. Institute of Electrical and Electronics Engineers, 2019. 7
- [11] *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
- [12] 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>. 10

- [13] ISO/IEC/IEEE: *Systems and software engineering-Life cycle management*. Institute of Electrical and Electronics Engineers, 2020. 11, 12
- [14] ISO/IEC/IEEE: *Systems and software engineering-Software life cycle processes*. Institute of Electrical and Electronics Engineers, 2017. 12, 13, 14, 15
- [15] 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, 20, 30
- [16] 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
- [17] 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. 18, 19
- [18] KAZMAN, Rick, et.al.: *Scenario-based analysis of software architecture*. IEEE Software, 1996. 20, 21, 22, 30
- [19] *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. 23, 24, 25, 26, 27, 31, 32, 33, 34, 35, 36, 37, 38, 39