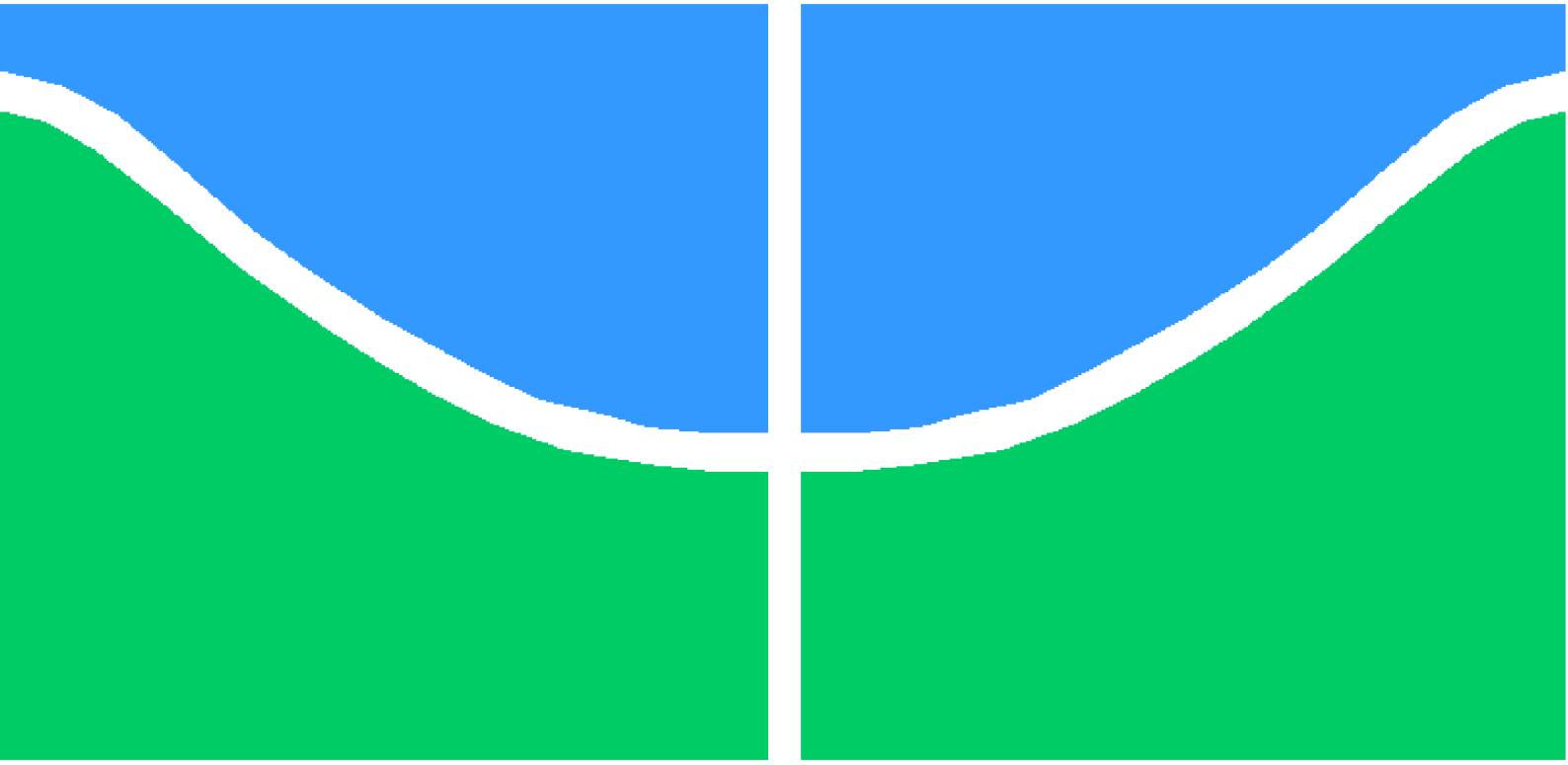

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Engenharia de software

Proposta de uma arquitetura integrativa baseada em serviços para um ambiente virtual

Autora: Beatriz Ferreira Gonçalves

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Brasília, DF

2016

Beatriz Ferreira Gonçalves

Proposta de uma arquitetura integrativa baseada em serviços para um ambiente virtual

Monografia submetida ao curso de graduação em Engenharia de softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Brasília, DF

2016

Beatriz Ferreira Gonçalves
Proposta de uma arquitetura integrativa baseada em serviços para um ambiente virtual/ Beatriz Ferreira Gonçalves. – Brasília, DF, 2016-
50 p. : il. (algumas color.) ; 30 cm.

Orientador: Professor Doutor Sérgio Antônio Andrade de Freitas

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2016.

1. Palavra-chave01. 2. Palavra-chave02. I. Professor Doutor Sérgio Antônio Andrade de Freitas. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Proposta de uma arquitetura integrativa baseada em serviços para um ambiente virtual

CDU COLOCAR CDU

AQUI - DADOS DA FICHA CATALOGRAFICA

Beatriz Ferreira Gonçalves

Proposta de uma arquitetura integrativa baseada em serviços para um ambiente virtual

Monografia submetida ao curso de graduação em Engenharia de softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de software.

Trabalho aprovado. Brasília, DF, XX de Junho de 2016:

Professor Doutor Sérgio Antônio Andrade de Freitas
Orientador

Titulação e Nome do Professor Convidado 01
Convidado 1

Titulação e Nome do Professor Convidado 02
Convidado 2

Brasília, DF
2016

ESCREVER AQUI A DEDICATÓRIA

Agradecimentos

ESCREVER AQUI OS MEUS AGRADECIMENTOS

ESCREVER AQUI A MINHA EPÍGRAFE

Resumo

TODO: ESCREVER O RESUMO. DICAS SOBRE COMO ESCREVER O RESUMO ESTÃO LOGO ABAIXO

O resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto. O texto pode conter no mínimo 150 e no máximo 500 palavras, é aconselhável que sejam utilizadas 200 palavras. E não se separa o texto do resumo em parágrafos.

Palavras-chaves: latex. abntex. editoração de texto.

Abstract

TODO: ESCREVER AQUI O ABSTRACT DO TRABALHO This is the english abstract.

Key-words: latex. abntex. text editoration.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SOA	*Service-Oriented Architecture*
SOAP	*Simple Object Access Protocol*
REST	*Representational State Transfer*
ESB	*Enterprise Service Bus*
API	•

Lista de símbolos

Γ	Letra grega Gama

Sumário

1 INTRODUÇÃO

ESCREVER O INICIO AQUI. INTRODUÇÃO DEVE SER ABRANGENTE.

1.1 Objetivos

São apresentados nessa seção os objetivos gerais, objetivos específicos e a questão de pesquisa.

1.1.1 Objetivos Geral

DESCREVER AQUI O OBJETIVO GERAL.

1.1.2 Objetivos específicos

Os objetivos deste trabalho são:

- DESCREVER OS OBJETIVOS ESPECÍFICOS.

1.1.3 Questão de pesquisa

EXPOR QUESTÃO DE PESQUISA.

1.2 Motivação

EXPOR MOTIVAÇÃO PARA A REALIZAÇÃO DO TRABALHO.

1.3 Metodologia

Esss seção apresenta a metodologia que será usada no desenvolvimento desse trabalho:

1.3.1 Classificação da pesquisa

CLASSIFICAR A PESQUISA.

1.3.2 Referencial teórico

DESCREVER COMO A PESQUISA FOI REALIZADA E ONDE O REFERENCIAL TEÓRICO ESTA CONTIDO NO DOCUMENTO.

1.3.3 Proposta

DESCREVER A PROPOSTA.

1.3.4 Engenharia de software

DESCREVER AS PRÁTICAS DE ENGENHARIA DE SOFTWARE ADOTADAS PARA O DESENVOLVIMENTO DO TRABALHO.

1.4 Estrutura da Monografia

DESCREVER COMO O DOCUMENTO ESTA ESTRUTURADO.

2 Referencial Teórico

"Este capítulo tem como objetivo apresentar o referencial teórico que embasa a pesquisa deste trabalho, incluindo conceitos que serão utilizados para a construção da proposta de trabalho a ser realizado."

2.1 Arquitetura de Software

Existe na literatura, e também é de senso comum da área de tecnologia da informação, a assertiva de que o produto de software é construído com base em uma oportunidade de negócio ou uma necessidade de usuário(s) identificada. Estes produtos de software possuem uma arquitetura associada à sua construção, que é uma composição de estruturas de um ou mais sistemas que exibem não apenas as características visíveis de elementos que o compõem, mas também o relacionamento entre estes elementos (BASS; CLEMENTS; KAZMAN, 2003).

A arquitetura de software pode ser vista como uma ponte que conecta as necessidades de usuário ou as oportunidades de negócio identificadas ao produto de software construído. Tal arquitetura representa uma abstração do sistema de software a ser desenvolvido e exibe os detalhes que o arquiteto de software julga como necessários. Desta forma, quaisquer produtos de software possuem uma aruiteura definida, independentemente de terem passado pelos processos de desenho, documentação e análise ou não (BASS; CLEMENTS; KAZMAN, 2003).

Bass, Clements e Kasman(BASS; CLEMENTS; KAZMAN, 2003) definem arquitetura de software como "a estrutura ou conjunto de estruturas de um sistema que comprime os elementos de um software, as propriedades externamente visíveis de tais elementos e os relacionamentos entre eles". Desta definição é possível inferir que sistemas podem ser construídos utilizando-se mais de uma estrutura; que os elementos que compõem o sistema, mas que não interagem diretamente, são omitidos na arquitetura; que também faz parte da arquitetura o comportamento e interação dos elementos; e que, como mencionado anteriormente, todo sistema ou produto de software possui uma arquitetura (BASS; CLEMENTS; KAZMAN, 2003).

Ainda de acordo com as ideias expostas por (BASS; CLEMENTS; KAZMAN, 2003), autores importantes na área de arquitetura de software, a definição formal da arquitetura de um software tem sua importância quando o assunto é a comunicação entre envolvidos e decisões importantes: colabora na comunicação entre as partes envolvidas e na tomada de decisões ainda no início do projeto de software, permitindo que outros

sistemas possam utilizar abstrações semelhantes.

2.1.1 Estilos Arquiteturais

De acordo com Pressman (PRESSMAN, 2006), estilos arquiteturais são utilizados para guiar o desenvolvimento de software e podem ser combinados a fim de obter um estilo próprio para cada produto de acordo com os requisitos e restrições identificadas. A seguir, estão descritos alguns dos estilos arquiteturais existentes e descritos na literatura.

2.1.1.1 Arquitetura Baseada em Camadas

A arquitetura baseada em camadas é caracterizada pela divisão de elementos em grupos que possuem responsabilidades semelhantes. Estes grupos compõem camadas da aplicação e estas conversam entre si através de um protocolo estabelecido pelo estilo arquitetural, onde, geralmente, uma camada interage apenas com camadas mais próximas (PRESSMAN, 2006). A figura a seguir ilustra este estilo.

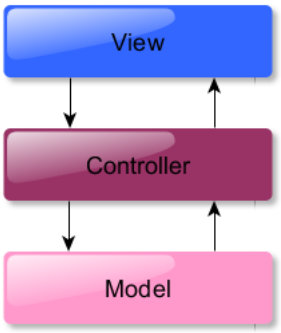

Figura 1: Padrão MVC - Arquitetura baseada em camadas.

A figura acima exibe o padrão arquitetural MVC - *Model View Controller* -, que é uma implementação do estilo arquitetural baseado em camadas. No MVC, a camada de modelo (*Model*) interage apenas com a camada de controle (*Controller*). Esta, por sua vez, é responsável por promover a interação entre as camadas *View* e *Model*.

2.1.1.2 Arquitetura Orientada a Serviços

Este é um estilo que promove a interoperabilidade de um dado sistema, permitindo troca de dados e interação entre diversas aplicações independentemente das plataformas em que são executadas ou tecnologias utilizadas para a sua construção (O. . . , 2010)

Na arquitetura orientada a serviços, as funcionalidades ou módulos de um sistema são definidos como um serviço. Os serviços disponibilizados são expostos através do estabelecimento de contratos e interfaces de acesso e requisitados por meio do envio e recebimento de mensagens pelas aplicações (O. . . , 2010).

2.1.1.3 Arquitetura Cliente-Servidor

O estilo arquitetural cliente-servidor é utilizado como um modelo para a implementação de sistemas distribuídos. Neste estilo, os clientes são responsáveis por realizar requisições a um conjunto de servidores que disponibilizam serviços. Geralmente, os clientes se comunicam de maneira direta com os servidores e possuem conhecimento apenas dos servidores disponíveis, desconhecendo os outros clientes existentes. Um exemplo de implementação deste estilo é a rede de uma organização onde os usuários (clientes) têm conhecimento acerca de impressoras (servidores) disponíveis. Ao acionar o serviço de impressão, o cliente estebelece uma comunicação direta com a impressora (SOMMERVILLE et al., 2008).

2.1.1.4 Arquitetura Baseada em Eventos

A arquitetura baseada em eventos é um estilo arquitetural onde ocorrências importantes no sistema são identificados pelo software ou hardware que o compõe. É composta por elementos que criam um evento, que apenas sabem que um evento ocorreu e o anuncia aos demais elementos do sistema, e por aqueles que consomem os eventos anunciados. Os elementos consumidores necessitam dos eventos para realizar o processamento de uma determinada operação ou mudança de estado (ROUSE, 2011).

2.2 SOA - Arquitetura Orientada a Serviços

Sendo tratado como um conceito evolucionário, a orientação a serviços é uma abordagem que foi criada para a construção de sistemas de software distribuídos, a fim de promover a integração com baixo acoplamento entre aplicações e facilitar a manutenção corretiva, adaptativa ou evolutiva das mesmas (LINTHICUM et al., 2007). Em outras palavras, a arquitetura orientada a serviços, também conhecida como SOA - do acrônimo em inglês *Service-Oriented Architecture* - é "um paradigma para a construção e manutenção de processos de negócio que conecta sistemas distribuídos"(JOSUTTIS, 2007).

Este modelo arquitetural é utilizado para o desenho, construção, implantação e gerenciamento de sistemas de software, onde as funcionalidades deste sistema são providos por serviços que possuem interfaces de acesso bem definidas (LEWIS, 2010). Desta forma, podemos afirmar que SOA não é um tipo de tecnologia, ferramenta ou processo a serem utilizados para a construção de software, mas uma abordagem utilizada para a definição e construção da arquitetura de determinadas aplicações (OLIVEIRA; NAVARRO,).

De acordo com Josuttis (JOSUTTIS, 2007), SOA é um recurso a ser utilizado para construir uma arquitetura de software concreta e tem como objetivo melhorar a flexibilidade de um sistema de software, baseando-se em três conceitos técnicos principais: serviços, interoperabilidade promovida por um barramento de serviços e baixo acopla-

mento. SOA é uma abordagem adequada para a implementação de sistemas distribuídos em que sistemas heterogêneos são aceitos, ou seja, aplicações desenvolvidas em plataformas diferentes e em linguagens de programação distintas são capazes de interagirem formando um sistema único (JOSUTTIS, 2007).

A indústria de software frequentemente implementa o modelo SOA utilizando Web Services que são, de acordo com a W3C (HAAS; BROWN, 2004), sistemas de software construídos para dar suporte à interação ponto-a-ponto de maneira interoperável através da rede. Contudo, a orientação a serviços é algo independente de tecnologias e padrões, justificando sua implementação quando sistemas legados devem ser incorporados à arquitetura de um sistema de software (LINTHICUM et al., 2007). A implementação deste modelo pode combinar diversas tecnologias, APIs, diferentes composições de infra-estrutura, constituindo sempre uma arquitetura única (ERL, 2009).

Como qualquer modelo, SOA apresenta benefícios para a construção de software e características que podem ser ditas como desvantajosas quando este modelo é utilizado. A integração com outros serviços, aplicativos e sistemas legados, além de prover um investimento de retorno elevado, a reutilização, flexibilidade, intereoperabilidade e governança de um serviço caracterizam algumas das vantagens relacionado ao uso este modelo (O. . . , 2010) (VANTAGENS. . . , 2013). As desvantagens identificadas estão relacionados a segurança de acesso, complexidade devido à quantidade de serviços (quanto mais robusta, ou seja, quanto mais serviços, mais complexa será a arquitetura construída), performance do servidor que afeta a disponibilidade do sistema de software e a testabilidade deste (O. . . , 2010) (VANTAGENS. . . , 2013).

2.2.1 Conceitos Principais

2.2.1.1 Serviços

Serviço pode ser definido como uma aplicação de software que interage com outras aplicações por meio da troca de mensagens (LINTHICUM et al., 2007). Além disso, um serviço é uma coleção de capacidades (ERL, 2009), "é um valor entregue para outro (serviço) através de uma interface bem definida e disponível e resulta em um trabalho provido de um para outro"(Adaptive Ltd et al., 2009).

Um serviço é uma aplicação independente e deve ser composto por duas partes principais: a interface, que permite a comunicação com os usuários do serviços e define a estrutura das mensagens a ser utilizada para a comunicação entre um serviço e seu usuário; e a implementação, que consiste do núcleo do serviço, desconhecido pelo usuário, mas a parte responsável pela execução do serviço (LINTHICUM et al., 2007) . As principais características de um serviço, de acordo com Jossutis (JOSUTTIS, 2007) e Erl (ERL, 2009), são:

- Um serviço deve ser **autônomo**, capaz de controlar seu ambiente e recursos disponibilizados. Isto implica na não interferência de fatores externos à aplicação que implementa o serviço, dependendo apenas de parâmetros fornecidos pelos usuários de um dado serviço.
- Um serviço deve estar sempre **visível e disponível** para que possa ser descoberto e utilizado por seus usuários.
- Um serviço deve possuir uma **alta abstração**, de modo que os detalhes de implementação sejam ocutados e apenas a interface seja acessível.
- Um serviço **não deve guardar** informações sobre o **estado** de requisições anteriores. Informações acerca do estado de um serviço deve ser mantida apenas quando necessário.
- Um serviço deve ser construído de modo que possa ser **reutilizado** por outras aplicações.
- Um serviço pode ser construído a partir da **composição de outros serviços**.
- Um serviço deve ser **idempotente**, isto é, ao ser utilizado um serviço deve retornar sempre o mesmo resultado quando os recursos disponibilizados para a sua execução forem os mesmos.
- Um serviço deve possuir um **contrato de serviço padronizado**, que expressa o objetivo e a capacidade que o serviço implementa.

Exibir a imagem tres ponto quatro do arquivo "Services Oriented Architecture from Adobe".

2.2.1.2 Baixo Acoplamento

2.2.1.3 Interoperabilidade

usar esta referencia "Interoperabilidade em SOA Desafios e Padrões"

2.2.2 Modelos de integração

A integração entre os subsistemas de software que compõen a arquitetura distribuída de um sistema construído com base no modelo arquitetural SOA pode ser estabelecida usando-se diferentes estatégias. A estratégia utilizada para promover tal integração é um aspecto que deve ser cuidadosamente analisado, pois o impacto de tal decisão é importante e irá perpertuar-se durante a existência do produto final de software construído. Desta forma, de acordo com Bianco et. al. (BIANCO; KOTERMANSKI; MERSON, 2007)

existem duas principais abordagens para a integração entre os sistemas que compõem a implementação deste modelo:

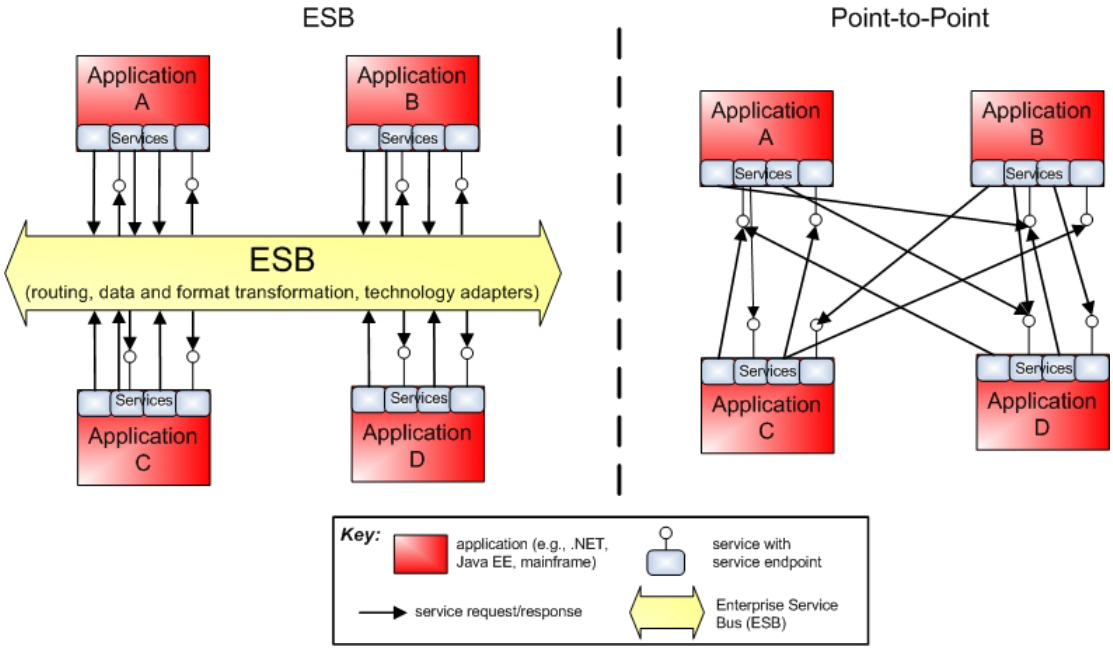

Figura 2: Ilustração dos modelos de integração em SOA. Fonte: (BIANCO; KOTERMANSKI; MERSON, 2007).

- **Direto (Ponto-a-Ponto)**: a interface de comunicação é única entre usuários e provedores de serviço; as questões relacionadas a conectividade entre os sistemas devem ser de reponsabilidade compartilhada entre tais aplicações.

- ***Hub-and-Spoke***: a comunicação entre usuários e provedores de serviço é mediada por um terceiro software chamado ESB (*Enteprise Service Bus*) ou EAI (*Enterprise Application Integration*); nesta abordagem, as aplicações estebelecem uma comunicação com o ESB (ou EAI), responsável por gerenciar as mensagens enviadas pelas aplicações.

2.2.3 ESB - Enterprise Service Bus

- O que é?

- Como funciona?

- Por que usar?

- Qual a utilidade?

2.2.3.1 WSO2 ESB

2.2.3.2 JBoss ESB

2.2.3.3 ErlangMS

2.3 Protocolos de Comunicação

- O que são? Para que servem?

- Quais são os protocolos existentes? Como eles funcionam?

2.3.1 SOAP

2.3.2 REST

2.4 Implementações do modelo SOA existentes

2.5 Engenharia de Software

2.5.1 Conceitos de ESW adotados

- RUP - Scrum

3 A PROPOSTA

Este capítulo apresenta a proposta do trabalho de conclusão de curso, exibindo detalhes da implementação a ser realizada acerca da arquitetura bem como o protocolo de comunicação dentro desta.

3.1 Introdução

Avanços tecnológicos e a criação de linguagens de programação, diferentes técnicas e paradigmas e outros conceitos relacionados ao desenvolvimento de software contribui para que a necessidade de interação entre estes elementos seja emergente, de modo a viabilizar a construção de sistemas cada vez mais robustos e inteligentes. Esta interação entre elementos de software não consistem de aplicações robustas que executam todas as suas atividades de forma independente de outras aplicações, pois cada vez mais diversos sistemas de software interagem com outros, podendo ser estes desenvolvidos tomando como base outros paradigmas ou escritos em outras linguagens de programação e utilizando-se diferentes técnicas.

A fim de suprir esta necessidade de interação entre sistemas diversos, foi criado um modelo arquitetural conhecido como Arquitetura Baseada em Serviços (ou *Service-Oriented Architecture* - SOA). Este modelo arquitetural utiliza o conceito de serviço como uma unidade que representa uma funcionalidade do sistema, além de trazer consigo os conceitos de interoperabilidade, flexibilidade, extensibilidade e baixo acoplamento entre os diversos sistemas ou serviços.

Para este trabalho de conclusão de curso, a proposta consiste em desenvolver uma arquitetura baseada no modelo SOA para um ambiente virtual, propiciando que diversos trabalhos já desenvolvidos e também aqueles em desenvolvimento não sejam mais "engavetados". Por meio do uso do modelo arquitetural baseado em serviços, será possível integrar tais aplicações, ou serviços, de modo que estas possam trocar dados e fazer uso do serviço disponibilizado por outras, indepentemente das tecnologias utilizadas para o desenvolvimento das mesmas.

Também faz parte da proposta, o estabelecimento de um protocolo de comunicação entre as aplicações, bem como o padrão de comunicação a ser utilizado, uma vez que as aplicações produzidas por outros podem se comunicar de modo a se tornarem mais robustas e completas enquanto ferramentas.

Desta forma, será viável a disponibilização à sociedade, interna e externa à Universidade, de uma plataforma virtual que conterá os trabalhos realizados dentro da mesma,

sejam oriundos de projetos de conclusão de curso, disciplinas ou projetos de extensão e pesquisa.

3.2 Proposta de arquitetura

3.2.1 Requisitos

A partir da necessidade identificada de disponibilizar aplicações que foram desenvolvidas, bem como aquelas que estão em desenvolvimento e que serão desenvolvidas, através de uma plataforma virtual único, algumas das principais características arquiteturais deste ambiente que influenciaram na escolha do modelo arquitetural para a construção da plataforma são:

- A comunicação entre as aplicações deve permitir a troca de dados independentemente das tecnologias utilizadas para seu desenvolvimento.
- O acoplamento entre aplicações deve ser o mínimo possível.
- Extensibilidade, permitindo que novas aplicações/componentes sejam inseridas à plataforma.
- Escalabilidade, fornecendo suporte para que diversas aplicações (ou componentes) sejam aderidas à plataforma.
- Flexibilidade, possibilitando a extensão da plataform sem que a arquitetura original seja modificada drasticamente.

A partir das características acima, foi proposto o uso do modelo arquitetural SOA - *Service-Oriented Architecture* -, pois desta forma a plataforma virtual terá conhecimento sobre as aplicações por meio das interfaces disponibilizadas, mas não precisará ter conhecimento sobre como ou quais tecnologias foram utilizadas para o desenvolvimento das aplicações. As aplicações neste contexto também podem ser denominadas serviços ou funcionalidades da plataforma virtual.

3.2.2 A arquitetura

A proposta de arquitetura a ser implementada faz uso da abordagem de implementação de SOA chamada "*Hub-and-spoke*", onde a interface de comunicação entre os serviços é única e pode ser realizada com o uso de um barrameneto de serviços ou um Enterprise Service Bus (ESB).

O barramento de serviços é um recurso a ser utilizado na implementação da arquitetura baseada no modelo SOA para facilitar a troca entre mensagens entre as aplicações

- ou serviços. Este barramento é uma ferramenta que implementa funcionalidades que roteiam as mensagens entre os usuários e provedores de um determinado serviço, transformam as mensagens e os dados para o formato aceito pelas aplicações e aceita protocolos múltiplos de comunicação através de adaptadores. Na arquitetura proposta, o protocolo de comunicação será padronizado e unificado e apenas as outras características do barramento de serviços serão utilizadas.

Sendo interoperabilidade um dos requisitos relevantes para a escolha do modelo arquitetural, o barramento de serviços é visto como um recurso que pode ser utilizado para ajudar a promover a interoperabilidade na arquitetura definida e na validação de políticas e critérios de segurança a serem definidas em trabalhos posteriores.

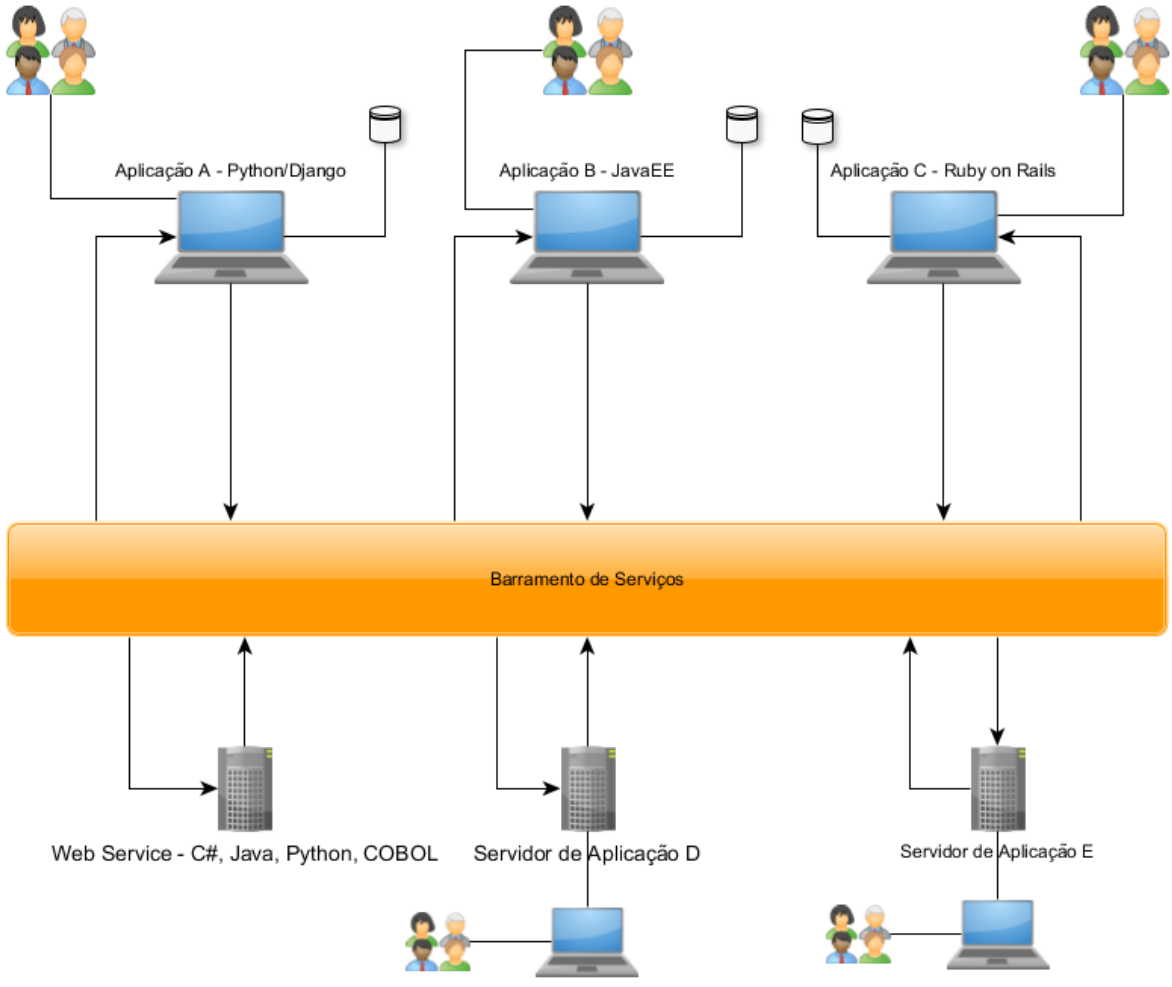

Figura 3: Interoperabilidade em uma arquitetura baseada no modelo SOA.

A figura acima visa demonstrar a interoperabilidade em uma arquitetura baseada no modelo SOA: é possível inferir que as diversas aplicações fazem a requisição dos serviços disponíveis por meio do uso do barramento de serviços, que também pode ser interpretado como um barramento de aplicações. As aplicações podem ser desenvolvidas utilizando-se tecnologias e paradigmas distintos e a troca de dados entre elas se darão de

forma bidirecional via mensagens de requisição e resposta entre as aplicações usuário, que requisitam operações dos serviços, e os serviços, responsáveis por processar as requisições e fornecer a resposta correspondente.

A figura também exibe um fato interessante na arquitetura proposta: não é necessário que um serviço seja apenas um serviço hospedado em um servidor. As aplicações poderão operar tanto em modo *standalone*, sendo executadas de forma independente dos outros serviços ou aplicações, quanto como um serviço para a plataforma virtual ou para outras aplicações que tenham conhecimento da existência e do protocolo de uso deste serviço.

O ESB é uma ferramenta que fornece as funcionalidades do barramento de serviços. Seu uso irá garantir que, sempre que este estiver ativo, as requisições que serão realizadas sempre terão uma resposta, mesmo sendo algo que indique a inatividade do serviço requerido ou a não autorização para acesso à este. Ao adicionar um novo serviço à arquitetura utilizando o ESB, deverão ser especificados tanto os procedimentos quando uma nova requisição for feita quanto aqueles passos que conduzem ao envio de uma resposta de acesso ao serviço especificado, sendo esta resposta advinda do próprio serviço ou uma resposta padrão, também especificada, em caso de falha ou sucesso ao executar a requisição.

Com base nos requisitos essenciais levantados e no estudo realizado sobre o modelo arquitetural SOA, o modelo proposto pode ser visto na imagem abaixo:

A comunicação entre aplicações e serviços deverão seguir o padrão de protocolo também definido durante o desenvolvimento deste trabalho de conclusão de curso, para que seja mantida uma regra de execução na troca de informações. O protocolo também facilitará a adição de um novo serviço à arquitetura no que diz respeito aos procedimentos de transformação dos dados e adaptação entre tecnologias e protocolos de transporte e comunicação adotados.

3.2.2.1 Ferramentas ESB

Existem diversas ferramentas deste tipo disponíveis e em uso por grandes organizações, tais como JBoss ESB, Mule ESB, Zato e WSO2 ESB. Para o conhecimento sobre a viabilidade de execução do trabalho aqui proposto, algumas destas ferramentas já foram levantadas, e, sendo o ESB um elemento importante para a implementação aqui proposta, uma análise prévia destas ferramentas já foi realizada. O levantamento feito mostrou que as ferramentas que podem ser utilizadas para a implementação da arquitetura são o JBoss ESB, WSO2 ESB e ErlangMS. Os critérios utilizados foram:

- Ser uma ferramenta de código aberto e/ou *free*;

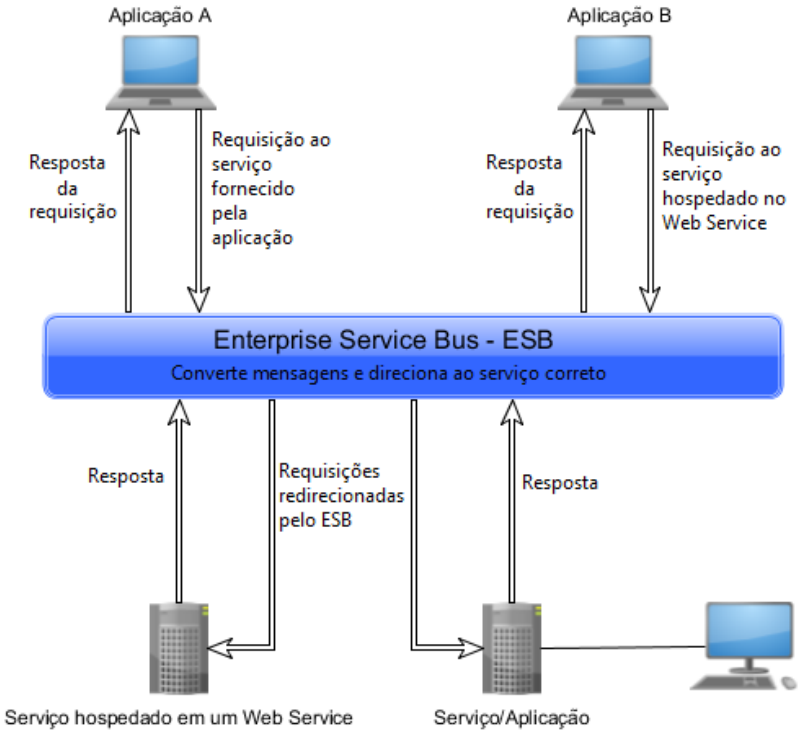

Figura 4: Proposta da arquitetura baseada no modelo SOA com o uso de um ESB.

- Possuir documentação e tutoriais disponíveis;
- Facilidade para implantação;
- Facilidade para uso;
- Possibilida de de uso de conectores (customizados e existentes);
- Suporte ao formato de mensagem escolhido para a implementação do protocolo.

3.2.3 Protocolo de comunicação

O protocolo de comunicação proposto para que ocorra a troca de dados entre usuários e provedores de serviço é um protocolo geral a ser utilizado por estes e adaptados sempre que necessário, porém não fugindo dos padrões estabelecidos. O protocolo proposto utiliza o modelo REST e, portanto, deve ser aderente à arquitetura deste. Este protocolo foi escolhido por ser de fácil uso, podendo ser implementado em diversas linguagens de programação (principalmente aquelas que são destinadas ao desenvolvimento de plataformas para a web) e em diversos sistemas operacionais. Além disso, o modelo REST utiliza o HTTP como protocolo de transporte, contribuindo para que a comunicação entre diversas aplicações seja realizada de maneira mais estável.

A arquitetura do protocolo REST aceita alguns formatos tais como JSON, CSV e texto simples, como mencionado nos capítulos anteriores. O formato definido para uso neste protocolo é o JSON, por ser um formato que permite a composição da mensagem através de chaves e valores. Assim, quando de posse da mensagem, os valores podem extraídos de acordo com a chave. Estas informações deverão ser publicadas pelo serviço na especificação de suas operações e valores de retorno quando forem acessados.

A fim de permitir o acesso ao serviço, as aplicações devem disponibilizar uma API REST para que os recursos sejam manipulados através das operações. Alguns serviços podem necessitar de autenticação e autorização para acesso ao mesmo, porém outros estarão disponpiveis para uso sem a necessidade de que tais recuros de segurança sejam implementados. Os requisitos de segurança na troca de dados deverão ser planejados em trabalhos futuros.

Desta forma, o fluxo básico do protocolo de comunicação entre os provedores e usuários dos serviços pode ser visto na imagem abaixo:

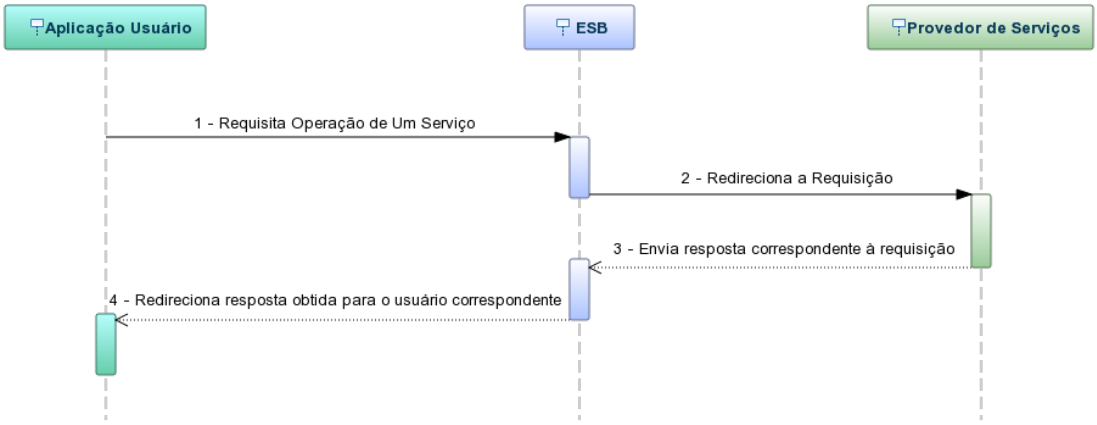

Figura 5: Fluxo básico do protocolo de comunicação.

A imagem exibe o fluxo geral do protocolo de comunicação estabelecido entre o usuário e o provedor de serviços, sendo esta comunicação intermediada pelo ESB: a aplicação que faz uso do serviço requisita que uma operação disponibilizada pelo serviço seja executada, esta requisição, por sua vez, é tratada pelo ESB e redirecionada à aplicação provedora do serviço; a resposta da requisição feita também será intermediada pelo ESB e redirecionada à aplicação que realizou a requisição.

O ESB é o elemento da comunicação que detêm o conhecimento sobre os servidos providos na plataforma e é o ator responsável por realizar a entrega das mensagens de requisição e de resposta dos serviços. Caso necessário, é também uma responsabilidade do ESB, realizar a transformação/adaptação dos formatos das mensagens e dos protocolos usados.

3.2.3.1 Formato das Mensagens

Como anteriormente mencionado, o formato escolhido para a troca de mensagens entre as aplicações será o JSON. A escolha foi realizada pelo fato de este formato de mensagem ser leve, de fácil entendimento e implementação, além de permitir que não seja difícil a recuperação dos dados em qualquer linguagem de programação, bem como pelo ESB.

O formato JSON é baseado em um esquema de chave-valor, onde a chave identifica um atributo, um dado, e o valor é o dado em si, o valor quantitativo ou qualitativo do atributo indicado pela chave. Este formato de mensagem adotado, pode ser tratado na aplicação como uma mensagem JSON ou como uma cadeia de caracteres, a depender da linguagem de programação adotada na construção da plataforma virtual e dos serviços disponibilizados.

Assim, para realizar uma requisição, a aplicação que executa o papel de usuário de um serviço deverá indicar o serviço e a operação desejada, o formato da mensagem (JSON por padrão) e os valores necessários para que o serviço seja executado corretamente, indicados pela API disponibilizada pelo mesmo. Da mesma forma, a resposta também deverá ser gerada em formato JSON, porém a aplicação que prover serviços retorna apenas a resposta da mensagem no padrão chave-valor. Abaixo podem ser visualizados um exemplo de requisição e outro de reposta, ambos em formato JSON.

```
url = "http://localhost:8280/services/facebookConnector"
headers = {'Action': 'urn:getUserDetails',
           'Content-type':'application/json'}
payload = {'apiUrl':'https://graph.facebook.com','apiVersion':'v2.5',
           'accessToken':access_token,
           'fields':'id,name,email,age_range,birthday'}
```

Figura 6: Exemplo de uma mensagem de requisição de serviço em formato JSON de uma aplicação usuário escrita em Python e Django.

```
{'id': 'user_id', 'name':'user_name',  'email':'user@email.com',
 'age_range':'20-25', 'birthday':'user_birthday'}
```

Figura 7: Exemplo de uma mensagem de resposta de requisição em formato JSON.

A primeira imagem exibe os valores necessários para realizar uma requisição ao serviço fornecido pela rede social Facebook através de sua API. Para a chamada do serviço, são necessários a especificação do endereço do serviço, indicado pela *url*; *headers* guarda os valores da operação a ser executada pelo serviço (’*Action*’) e o formato da mensagem (’*Content-type*’); os valores necessários para a execução da operação requisitada estão contidos no *payload* também em formato *’chave’:’valor’*.

As imagens acima mostram apenas exemplos do uso do formato de mensagem JSON para realizar uma requisição e de mensagem obtida como resposta advinda do serviço. Pode-se ver que os valores são correspondentes à uma chave conhecida por ambas as aplicações, permitindo que as aplicações (provedora e usuária de serviços) possam comunicar-se entre si de forma padronizada e conhecida por ambas as partes.

4 Desenvolvimento da Proposta

Este capítulo apresenta o uso de metodologias e conceitos relacionados à Engenharia de Software que serão aplicados no desenvolvimento da proposta descrita no capítulo anterior.

4.1 Introdução

O projeto de Engenharia de Software a ser desenvolvido como parte do trabalho de conclusão de curso consiste em uma arquitetura para uma plataforma virtual onde as funcionalidades serão tratadas como serviços no contexto arquitetural. Os serviços ou funcionalidades desta plataforma virtual consiste em trabalhos já realizados, em processo de desenvolvimento e também de trabalhos futuros que, quando incorporados a esta arquitetura, poderão ser disponibilizados para uso pela comunidade tanto acadêmica quando externa, deixando que ser "projetos de gaveta".

A proposta descrita no capítulo anterior, será desenvolvida utilizando-se de alguns conceitos de metodologias de desenvolvimento e de gerenciamento de projetos de software, adaptadas conforme as necessidades deste projeto.

4.2 Metodologia de Execução da Proposta

Um projeto de Engenharia de Software deve ser realizado utilizando-se de metodologias, técnicas e ferramentas disponíveis relacionadas à esta área de conhecimento, sendo sempre adaptadas de acordo com o projeto a ser desenvolvido visando, assim, o sucesso na conclusão do projeto.

Entre as diversas metodologias de desenvolvimento de software existentes, foi decidido pela adoção de uma metodologia híbrida para o desenvolvimento deste projeto. Esta metodologia híbrida contém alguns conceitos relacionados ao RUP (Rational Unified Process) e outros relacionados à metodologia ágil de desenvolvimento, mais especificamente o Scrum.

A metodologia híbrida adotada é composta pelas fases do RUP e alguns de seus artefatos. Envolve também o conceito de histórias de usuário, que em sua marioria serão tratadas como histórias técnicas neste projeto. Além disso, o modelo de desenvolvimento é iterativo e incremental, tornando a identificação de falhas e correção das mesmas mais eficiente, assim como a identificação de novas necessidades para que a arquitetura e o protocolo de comunicação propostos sejam implementados.

Foi modelado um processo para a execução das atividades necessárias para a realização do trabalho de conclusão de curso, tanto para que a proposta pudesse ser elaborada, quanto para o planejamento e execução desta. A imagem abaixo ilustra o processo, ainda em andamento, e contém os aspectos relacionados à metodologia adotada e descrita no parágrafo anterior.

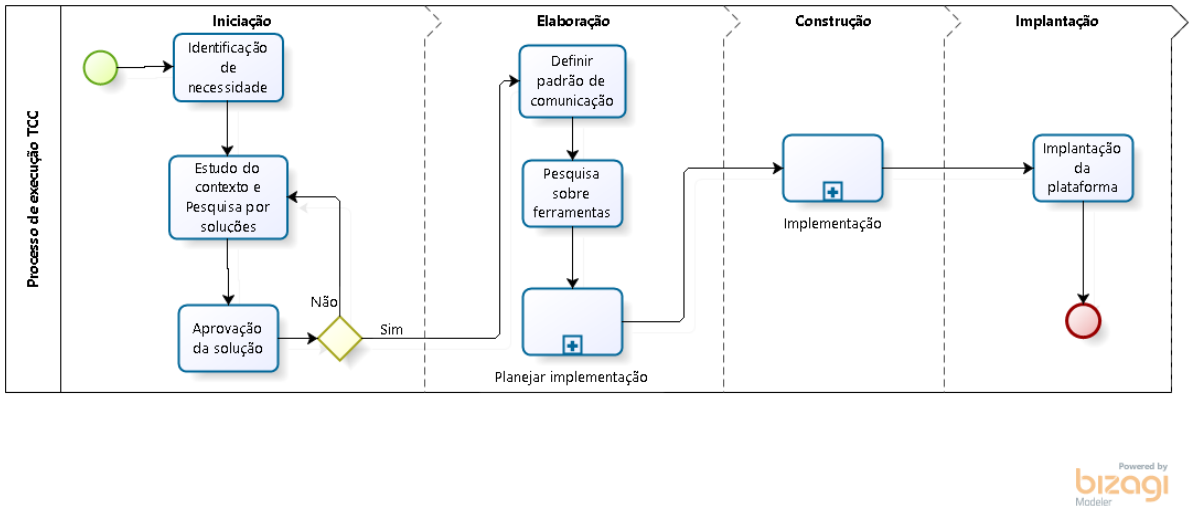

Figura 8: Processo de elaboração e execução do TCC.

A imagem exibe o processo que está sendo executado, com suas fases e as atividades correspondentes de cada fase. Como anteriomente mencionado, o processo foi dividido de acordo com as fases do RUP: iniciação, elaboração, construção e implantação. A fase de iniciação contém as atividades de *Identificação de necessidade*, *Estudo do contexto e Pesquisa por soluções* e *Aprovação da Solução*: uma vez que a necessidade foi identificada, foi realizado um estudo sobre o contexto da necessidade e uma pesquisa por soluções arquiteturais adequadas e que suprissem a necessidade; a solução, com o uso do modelo arquitetural SOA, foi proposta e aprovada.

A fase de elaboração contém atividades que procuram refinar conceitos relacionados ao este modelo arquiteural escolhido e definir padrões e ferramentas que serão utilizadas na fase de construção, além de obter um plano de execução para a fase seguinte. Assim, foram realizadas as atividades de definição de um padrão de comunicação para a arquitetura, uma pesquisa sobre ferramentas que podem ser úteis no desenvolvimento da arquitetura da plataforma virtual e o planejamento da fase de construção.

A fase de construção é aquela em que o planejamento realizado na fase anterior será executado. É nesta fase em que adaptações deverão ser feitas, tanto na plataforma virtual quanto em aplicações que fornecerão serviços. A fase de construção é constituída de ciclos iterativos e incrementais e será detalhada mais adiante.

A última fase definida no processo é a fase de implantação. Nesta fase a plataforma

deverá ser implantada e homologada para uso pela comunidade.

Até o final do TCC 1, as fases do processo que foram executadas são as fases de iniciação e elaboração do projeto. O processo definido poderá ser readaptado conforme caso seja necessário durante as próximas etapas de realização do trabalho. É sugerido que os processos definidos para as fases de construção e implantação sejam repetidos sempre que uma nova funcionalidade implementada por um serviço seja incorporada à plataforma virtual criada.

4.2.1 Planejamento da Implementação

Durante a fase de construção, um subprocesso definido foi o de planejamento da implementação (subprocesso da fase de construção do projeto). As atividades e o fluxograma deste subprocesso podem ser visualizados na imagem a seguir.

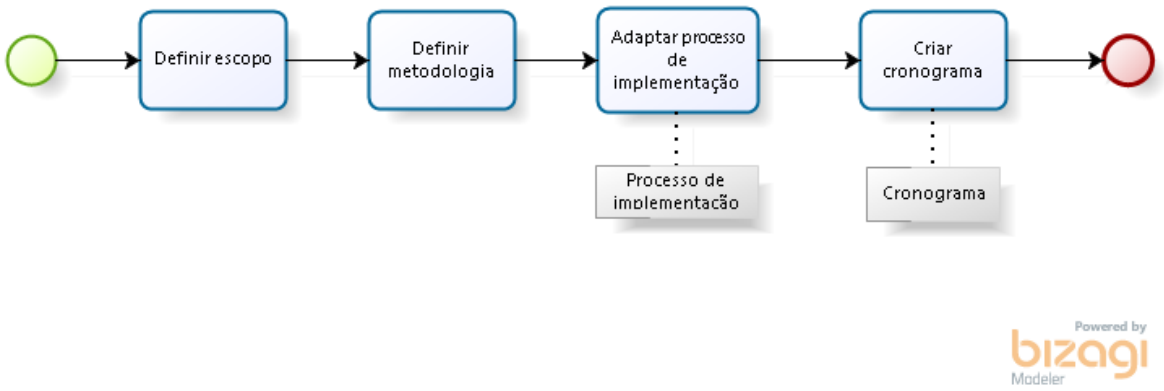

Figura 9: Fluxograma e atividades do subprocesso Planejar Implementação.

A figura acima exibe as atividades realizadas durante a fase de elaboração definida no processo e que já foi realizada durante o TCC 1. As atividades são *Definir escopo*, *Definir metodologia*, *Adaptar processo de implementação* e *Criar cronograma*. No início do planejamento o escopo do que será entregue ao final do trabalho de conclusão de curso foi definido. Esta atividade foi seguida da definição de uma metodologia de desenvolvimento de software para a execução do projeto e, baseando-se na metodologia definida, um processo a ser seguido para a implementação da arquitetura foi adaptado. Por fim, um cronograma de execução da implementação foi criado.

Na atividade de definição do escopo não foi definida a aplicação que será adaptada e incorporada à plataforma como um serviço a fim de demonstrar a proposta feita. Esta escolha deverá ser feita durante na fase de construção.

A metodologia definida foi do tipo híbrida, que utiliza conceitos das metodologias tradicional (RUP) e ágil (Scrum), como mencionado anteriormente neste capítulo.

O cronograma criado durante a última fase deste subprocesso poderá ser visto na seção final deste capítulo.

4.2.2 Fase de Construção

O uso de uma arquitetura baseada no modelo SOA não elimina o trabalho necessário para a inserção de novos serviços: o seu uso visa minimizar os reparos que devem ser feitos para que a nova funcionalidade seja incorporada ao sistema, promovendo extensibilidade e flexibilidade à arquiteura construída.

A fase de construção do precesso definido tem como objetivo a implementação da plataforma virtual de acordo com os requisitos básicos levantados e que foram importantes para a proposta da solução e a adaptação de uma aplicação existente para ser incorporada à plataforma. Em um contexto onde novos serviços serão inseridos na plataforma em um tempo posterior à execução do TCC, esta fase será também executada. Assim, o processo definido para a fase de construção pode ser visualizado na imagem a seguir.

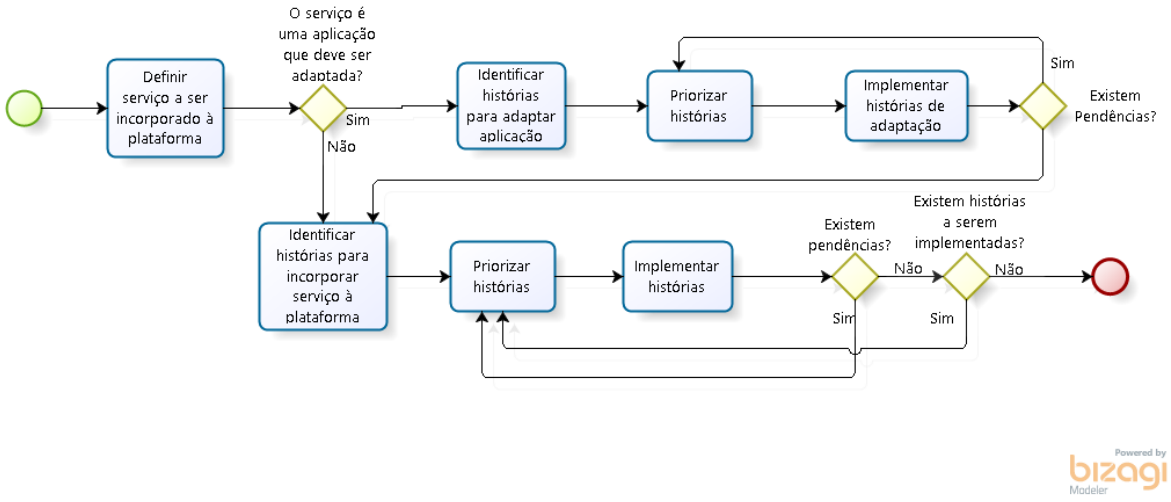

Figura 10: Fluxograma e atividades da fase de construção.

A imagem acima exibe o fluxograma de atividades a serem executadas no subprocesso de implementação definido na fase de construção da plataforma virtual.As atividades a serem realizadas são dependentes da definição do serviço que será incorporado à plataforma, pois, caso o novo serviço seja uma aplicação que deve ser adaptada, as atividades referentes à adaptação desta deverão ser executadas. Se o novo serviço, ou funcionalidade, não for uma aplicação a ser adaptada para fornecer suas operações, pode-se executar as atividades e tarefas referentes à construção da plataforma (no caso do trabalho de conclusão de curso) ou adaptação desta para que a nova funcionalidade seja disponibilizada na plataforma.

Um dos conceitos ágeis utilizados no subprocesso de implementação é o de histórias de usuário, que consiste na descrição de funcionalidades que agregam valor ao produto final a ser entregue de forma que possa ser facilmente entendida e escrita de maneira simples. As histórias a serem elicitadas consistirão tanto de histórias que descrevem funcionalidades, quando de histórias técnicas, que tratam de adaptação e implantação de tecnologias e outros aspectos mais relacionados à parte técnica do desenvolvimento de software so que à funcionalidades.

Outro conceito associado à metodologia ágil de desenvolvimento de software é o de *sprints*. Por se tratar de um modelo criado para ser iterativo e incremental, a fase de construção definida para este TCC consistirá de sprints com períodos definidos de 1 (uma) semana. Foi estabelecido que a quantidade de *sprints* será de 8 a 10.

A fase de construção e a subsequente (implantação) serão executadas durante a realização do TCC2.

4.3 Cronograma de Execução do TCC

Para o desenvolvimento da proposta e completude da mesma com sucesso foi criado um cronograma para as fases de construção da proposta aqui feita e de implantação do produto final obtido. O cronograma contém as atividades a serem realizadas, bem como os prazos relacionados a cada uma das atividades e pode ser visualizado na tabela abaixo.

Tabela 1: Cronograma de atividades relacionadas ao TCC 2

Atividade	Jul	Ago	Set	Out	Nov	Dez
Análise de ferramentas a serem utilizadas	X					
Implantação da ferramenta escolhida	X					
Adaptação de uma aplicação já desenvolvida		X	X	X		
Implementação da plataforma virtual		X	X	X		
Implantação da plataforma virtual				X		
Escrita do TCC 2					X	X

- **Análise de ferramentas a serem utilizadas:** como explicitado no capítulo anterior, que trata os detalhes da proposta, será utilizada uma ferramenta que fornece os serviços de roteamento, adaptação de tecnologias e tratamento de formatos de dados e de mensagens. Para que isso ocorra, algumas das ferramentas levantadas serão analizadas com mais rigor para que uma seja eleita. A análise consistirá de testes de implantação da ferramentas, bem como de uma avaliação das funcionalidades e da facilidade de uso e aprendizagem.

- *Implantação da ferramenta escolhida:* após a escolha da ferramenta, esta deverá ser implantada em um servidor que tenha capacidade para tal. A implantação deverá

facilitar o uso da ferramenta durante as atividades posteriores.

- **Adaptação de uma aplicação já desenvolvida:** aplicações já desenvolvidas podem não estar preparadas para serem incorporadas à plataforma virtual como um serviço. Para que isto ocorra, será selecionada uma aplicação para ser adaptada, sendo assim possível de ser disponibilizada na plataforma. Esta atividade também consiste em fazer com que as operações do novo serviço sejam acessadas via o protocolo estabelecido.

- **Implementação da plataforma virtual:** nesta atividade, a plataforma virtual será criada e o serviço que foi adaptado deverá ser incorporado a tal plataforma. Aqui serão implementados o uso do protocolo de comunicação estabelecido, bem como o uso da ferramenta ESB escolhida.

As atividades de adaptação de uma aplicação existente e a incorporação desta aplicação como um serviço à plataforma virtual utilizando a ferramenta ESB escolhida serão realizadas de maneira iterativa e incremental, justificando o uso de um método de desenvolvimento de software capaz de fornecer suporte para que iterações sejam executadas. A intenção é ter pelo menos uma aplicação adapatada ao padrão de protocolo estebelecido e sendo utilizada na arquitetura como um serviço.

Com relação às atividades realizadas durante o desenvolvimento do TCC 1, a tabela abaixo contém o registro das atividades realizadas bem como os seus respectivos períodos de execução.

Tabela 2: Cronograma de atividades relacionadas ao TCC 1

Atividade	Jan	Fev	Mar	Abr	Mai	Jun
Identificação da necessidade	X					
Leituras sobre modelos arquiteturais	X					
Pesquisa sobre o modelo arquitetural mais adequado	X	X				
Pesquisa sobre trabalhos relacionados	X	X				
Levantamento superficial de ferramentas ESB		X	X			
Definição do padrão de comunicação			X			
Organização das referências				X		
Escrita formal da proposta				X		
Estabelecimento de metodologia				X		
Escrita do TCC 1					X	X

5 Considerações finais

Referências

Adaptive Ltd et al. *Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS)*. OMG - Object Management Group, 2009. Disponível em: <http://www.uio.no/studier/emner/matnat/ifi/INF5120/v10/undervisningsmateriale/>. Citado na página 30.

BASS, L.; CLEMENTS, P. C.; KAZMAN, R. *Software Architecture in Practice*. 2. ed. [S.l.]: Addison-Wesley Professional, 2003. ISBN 0-321-15495-9. Citado na página 27.

BIANCO, P.; KOTERMANSKI, R.; MERSON, P. *Evaluating a Service-Oriented Architecture*. [S.l.], 2007. Disponível em: <http://www.sei.cmu.edu/library/abstracts/reports/07tr015.cfm>. Citado na página 31.

ERL, T. Orientação a serviços. In: GAMA, F. C. N. d.; BARBOSA, R. d. C. (Ed.). *SOA: Princípios de Design de Serviços*. São Paulo: Pearson Prentice Hall, 2009. p. 306. ISBN 978-85-7605-189-3. Citado na página 30.

HAAS, H.; BROWN, A. *Web Services Architecture*. 2004. Disponível em: <https://www.w3.org/TR/ws-arch/#whatis>. Citado na página 30.

JOSUTTIS, N. *Soa in Practice: The Art of Distributed System Design*. [S.l.]: O'Reilly Media, Inc., 2007. ISBN 0-596-52955-4. Citado 2 vezes nas páginas 29 e 30.

LEWIS, G. *Getting Started with Service-Oriented Architecture (SOA) Terminology*. Software Engineering Institute Carnegie Mellon University, 2010. Disponível em: <http://www.sei.cmu.edu/library/assets/whitepapers/SOA_Terminology.pdf>. Citado na página 29.

LINTHICUM, D. et al. *Service Oriented Architecture (SOA) in the Real World*. [S.l.]: Microsoft Corporation, 2007. Citado 2 vezes nas páginas 29 e 30.

O que é SOA e por que usá-la? 2010. Disponível em: <http://www.celtainformatica.com.br/noticias/o-que-e-soa-e-por-que-usa-la>. Citado 2 vezes nas páginas 28 e 30.

OLIVEIRA, M.; NAVARRO, R. Interoperabilidade em SOA: Desafios e Padrões. *SOA na prática*. Disponível em: <http://www.univale.com.br/unisite/mundo-j/artigos/37Interoperabilidade.pdf>. Citado na página 29.

PRESSMAN, R. *Engenharia de software*. McGraw-Hill, 2006. ISBN 9788586804571. Disponível em: <https://books.google.com.br/books?id=MNM6AgAACAAJ>. Citado na página 28.

ROUSE, M. *What is event-driven architecture (EDA)? - Definition from WhatIs.com*. 2011. Disponível em: <http://searchitoperations.techtarget.com/definition/event-driven-architecture>. Citado na página 29.

SOMMERVILLE, I. et al. *Engenharia de software*. ADDISON WESLEY BRA, 2008. ISBN 9788588639287. Disponível em: <https://books.google.com.br/books?id=ifIYOgAACAAJ>. Citado na página 29.

VANTAGENS e Desvantagens de SOA. 2013. Disponível em: <http://www.devmedia.com.br/vantagens-e-desvantagens-de-soa/27437>. Citado na página 30.