Capítulo 1

Arquiteturas de Software para o Domínio de Saúde

Lina Garcés, Brauner Oliveira, Carolina Arenas

Laboratório de Engenharia de Software - LabES
Instituto de Ciências Matemáticas e Computação - ICMC
Universidade de São Paulo - USP
E-mail: linamgr@icmc.usp.br, brauner@usp.br, carolina.arenas@usp.br

Abstract

Software architecture is a sub-discipline of software engineering focusing on the quality of software products/services at the early stage of software development. Despite its importance, healthcare software systems creators have had invested a few efforts to architect these solutions. Therefore, existing healthcare solutions commonly present problems with quality characteristics such as usability, performance, availability, security, safety, and interoperability. This chapter studies software architecture as a strategy to support the development of quality healthcare software systems. Besides presenting the software architecture theoretical foundation, in this chapter also is explained, in a didactic way, how a healthcare software solution could be architected. We expect the knowledge detailed in this chapter can bring a basic understanding to future architects continuing their learning on software architecture and its future application during healthcare software solutions with quality.

Keywords: *software architecture, software systems, software quality, e-Health.*

Resumo

Arquitetura de software é uma sub-disciplina da engenharia de software que visa a qualidade dos produtos e serviços durante as primeiras fases do processo de desenvolvimento. Apesar de sua importância, pouco se investe no projeto arquitetural de sistemas de software na área de saúde. Consequentemente, é comum encontrar soluções de software que apresentam problemas de desempenho, disponibilidade, segurança e interoperabilidade, dentre outros. Este capítulo tem como objetivo introduzir o conceito de arquitetura de software para apoiar o desenvolvimento de sistemas de saúde de qualidade. Além da fundamentação teórica, também é apresentado um exemplo de construção da arquitetura de

um sistema de saúde. Espera-se que o conhecimento fornecido seja suficiente para que o leitor possa iniciar ou aprimorar seu aprendizado na área de arquitetura e sua aplicação no desenvolvimento de sistemas de saúde de qualidade.

Palavras chave: *arquitetura de software, sistemas de software, qualidade de software, e-Saúde.*

1.1. Introdução

Nos últimos anos tem sido possível notar um aumento significativo no fornecimento de uma grande diversidade de equipamentos, dispositivos, tecnologias, serviços e sistemas de software para a área de saúde. Estima-se que o mercado mundial de "*healthtech*" cresceu mais de 141% nos últimos cinco anos, alcançando um total de mais de 14 bilhões de dólares em investimentos no ano de 2019 [Portal Saúde Business 2020]. Espera-se ainda que em 2025 esse setor econômico alcance um valor de US$ 504 bilhões [Wanderley 2020].

Nesse cenário, o Brasil está consolidado como o maior mercado de *healthtech* na América Latina e o sétimo maior do mundo, gastando cerca de 42 bilhões de dólares em serviços de cuidado de saúde por ano [Portal Saúde Business 2020]. Essa perspectiva é bastante positiva e tem fortalecido empresas nacionais e incentivado a criação de novas *startups* no setor de TI (Tecnologias da Informação) para saúde. No entanto, várias soluções tecnológicas disponíveis no mercado apresentam diversos problemas relacionados à qualidade, o que acaba sendo percebido pelos usuários finais, que em sua grande maioria são profissionais/especialistas na área de saúde, gestores de serviços de saúde e pacientes. Problemas relacionados à usabilidade, segurança, interoperabilidade, disponibilidade, desempenho são frequentes tanto nas tecnologias já estabelecidas no mercado bem como nas que estão surgindo a partir de inovação. Por exemplo, a baixa usabilidade das soluções resulta não apenas em frustração e fadiga dos profissionais, mas pode acarretar em erros que podem ameaçar a segurança do paciente [Ratwani et al. 2015, Caldeira 2017].

Com aparelhos médicos cada vez mais conectados, o aumento no reúso de tecnologias fornecidas por terceiros e o surgimento de novas tecnologias como *IoT (Internet of Things)*, *blockchain*, *big data*, e *cloud computing*, pode aumentar o risco de ataques cibernéticos na área de saúde. A falta de mecanismos responsáveis por mitigar ataques às tecnologias e serviços tem levado a prejuízos bilionários para o setor de saúde [CIO 2019]. Por exemplo, um evento que ficou mundialmente conhecido foi a série de ataques com o *ransonware* WannaCry, que afetou mais de 200 mil computadores e dispositivos em 150 países, "sequestrando" informações das instituições de saúde em troca de pagamentos em criptomoedas, conforme ocorreu com a rede de hospitais do Serviço Nacional de Saúde (NHS) da Inglaterra em 2017 [CBS News 2017].

A necessidade, cada vez mais presente, de interconectar sistemas que foram inicialmente projetados para operar isoladamente e apoiar processos clínicos e hospitalares de maneira "transparente" para os usuários finais, tem demandado a criação e/ou modificação de sistemas para permitir sua interoperação. A falta de interoperabilidade entre diferentes sistemas, dispositivos, tecnologias tem acarretado em perdas significativas nos processo de negócio das organizações de saúde e em problemas de segurança, acarretando em aumento nos custos e baixa qualidade no atendimento integral do paciente [Neto and

Junior 2019].

De forma similar, as soluções tecnológicas para o setor de saúde devem atender a níveis elevados de disponibilidade, já que o atendimento e cuidado dos pacientes deve ocorrer de maneira contínua. A baixa disponibilidade de recursos e funcionalidades de um sistema pode levar a grandes perdas econômicas para hospitais e gestores de saúde pública. Por exemplo, em 2015, uma falha no sistema da empresa *Cover Oregon*, responsável pelo desenvolvimento de um sistema web para a troca de informações entre convênios de saúde nos Estados Unidos acarretou em uma perda de aproximadamente 200 milhões de dólares devido a problemas de disponibilidade do sistema. A equipe técnica não considerou requisitos de capacidade, desempenho e disponibilidade da base de dados que era totalmente centralizada, representando uma vulnerabilidade conhecida por *single-point failure* na arquitetura do sistema [Turner 2015].

Os exemplos apresentados servem de motivação para destacar a necessidade de considerar a qualidade dos produtos e serviços tecnológicos em todas suas fases de desenvolvimento. Dessa forma, a fim de apoiar a criação de soluções software de qualidade, a área de engenharia de software vem contribuindo há mais de 50 anos com métodos, técnicas, ferramentas dentre outras abordagens para apoiar as diferentes atividades do processo de desenvolvimento de software, que é composto, principalmente, pela engenharia de requisitos, arquitetura de software, projeto (ou *design*) de software, codificação, teste, configuração, manutenção, evolução e gerenciamento. É preciso que as empresas do setor de TI, incluindo aquelas com foco no setor de saúde, visem a inclusão de boas práticas oriundas da engenharia de software para a construção de seus sistemas de software.

Neste capítulo, é introduzida a disciplina de arquitetura de software como subdisciplina da engenharia de software, que visa orientar a criação de projetos arquiteturais para que os sistemas de software atendam aos níveis de qualidade requeridos. Além disso, o conceito de arquitetura de software é apresentado neste capítulo como um dos principais artefatos do processo de desenvolvimento, responsável por apoiar a construção ou codificação do software e a comunicação entre diferentes *stakeholders*.

A **Seção 1.2** abre o capítulo apresentando os conceitos fundamentais de arquitetura de software e sua importância para o desenvolvimento de sistemas de alta qualidade, bem como as atividades envolvidas em sua construção. As **Seções 1.3**, **1.4**, **1.5**, **1.6** apresentam a aplicação dos conceitos e atividades para a criação da arquitetura de software do sistema *SIGEAN*, um software para a gestão da divisão de anestesia de um hospital público. O capítulo termina com a apresentação de algumas discussões finais na **Seção 1.7**.

1.2. Arquitetura de Software e Qualidade de Software

O projeto de um sistema de software compreende diversas decisões tomadas por equipes de desenvolvimento para atender um conjunto de requisitos. Parte dessas decisões devem ser tomadas de maneira cuidadosa, pois negligenciar esta atividade de projeto de software pode impactar profundamente todo o processo de desenvolvimento. Esse é o caso das decisões arquiteturais, que em sua totalidade resultam na arquitetura de um sistema de software.

A arquitetura de software de um sistema pode ser definida como o conjunto de

suas estruturas fundamentais, compreendendo elementos de software, suas relações e as propriedades de ambos [Bass et al. 2012]. Essas estruturas representam, por exemplo, as partes nas quais o sistema pode ser dividido para que seja mais fácil compreendê-lo. No nível arquitetural, essas partes são geralmente chamadas de módulos ou componentes, e nem sempre é possível identificá-las analisando o código-fonte de um sistema. É por esse motivo que, comparado à implementação, costuma-se dizer que o projeto arquitetural ou a própria arquitetura está ou ocorre num nível de abstração mais alto, omitindo detalhes de implementação para que seja possível abordar outras questões de projeto de software considerando o escopo total do sistema de software. Por se tratar de um tema abstrato, é possível encontrar várias outras definições de arquitetura de software pela internet. De fato, existem algumas divergências sobre o que pode ser considerado arquitetural ou não no projeto de software. Contudo, todas as definições convergem no sentido que arquitetura de software lida com questões macroscópicas do projeto de software [Fairbanks 2010].

Diferentemente das decisões de projeto, decisões arquiteturais podem afetar drasticamente, de maneira positiva ou negativa, diversas qualidades de um sistema, tais como desempenho, segurança e manutenibilidade. Além disso, reverter esse tipo de decisões em etapas avançadas do processo de desenvolvimento pode exigir muito esforço. Fazendo uma analogia com um prédio de vários andares, é muito mais provável que trocar uma porta ou janela de lugar seja mais fácil do que adicionar um elevador, um novo andar ou novos apartamentos em uma construção já finalizada. Às vezes, tais alterações podem até mesmo ser impossíveis devido às características da estrutura do prédio que por sua vez pode ser, por exemplo, incapaz de suportar a adição de um novo andar. Da mesma forma, a arquitetura de software de um sistema pode permitir que ele exiba certas qualidades mas também pode inibi-las [Bass et al. 2012] e impossibilitar que o requisitos de qualidade sejam atendidos.

Embora o projeto arquitetural de um sistema de software nem sempre seja considerado durante o desenvolvimento, é possível dizer que todo software possui uma arquitetura. Nesse caso, costuma-se dizer que a arquitetura se trata de uma grande bola de lama (em inglês *"big ball of mud"*), resultado indesejado que ainda pode ser encontrado no contexto de desenvolvimento de software profissional nos dias de hoje. Sistemas que sofrem desse tipo de problema se tornam cada vez mais difíceis de se manter à medida em que novas funcionalidades precisam ser incluídas ou se tornam mais complexas. Em contrapartida, arquiteturas bem projetadas são fundamentais para garantir a longevidade dos sistemas e o desenvolvimento de soluções desafiadoras, inovadoras e de qualidade, como é o caso, por exemplo, de serviços de *streaming* como YouTube e Twitch, redes sociais como Facebook e Instagram, e aplicativos de conversa como Telegram e WhatsApp. Além de estarem disponíveis na maioria dos países o tempo todo, esses serviços precisam ser escaláveis para atender eventuais demandas, e ainda desempenhar rápido o suficiente para que suas funcionalidades não sejam prejudicadas. Caso contrário, teriam que arcar com prejuízos como por exemplo a perda de usuários para concorrentes de mercado. Sendo assim, dada a importância do projeto arquitetural, é fundamental compreender como que ele é realizado na indústria de software.

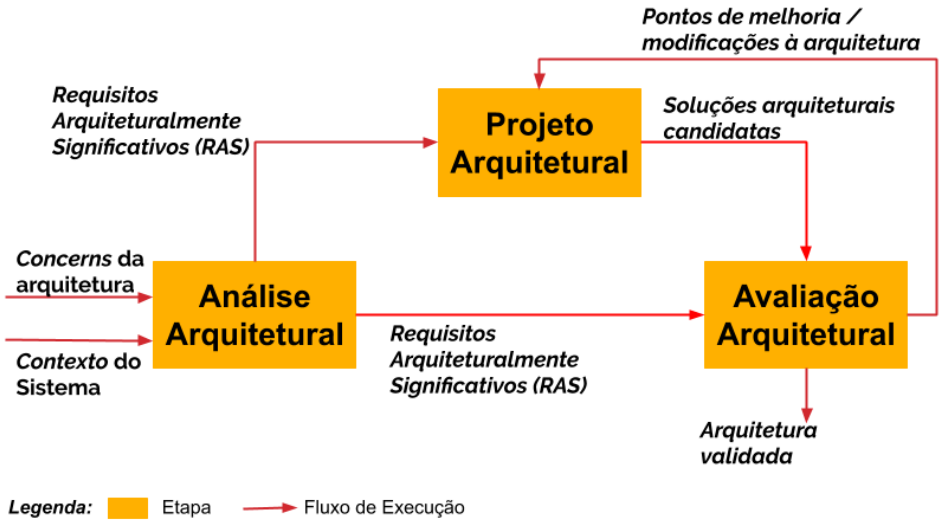

Figura 1.1. Modelo de processo arquitetural identificado a partir da análise de cinco abordagens industriais por [Hofmeister et al. 2007]

1.2.1. Processo Arquitetural

O projeto de uma arquitetura de software ocorre por meio de um modelo de processo geral que envolve três tipos de atividades [Hofmeister et al. 2007], conforme visto na **Figura 1.1**.

A primeira atividade - **análise arquitetural** - tem como objetivo identificar os requisitos que serão significativos para a construção da arquitetura, analisando-se para isso as preocupações arquiteturais e o contexto do projeto. As **preocupações arquiteturais** incluem questões importantes relacionadas ao desenvolvimento e a operação do sistema, bem como aspectos que são críticos ou importantes para alguma parte interessada (*stakeholder*). Em geral, tais preocupações são expressas por meio de requisitos de qualidade do sistema, tais como aqueles relacionados a desempenho, segurança e confiabilidade, mas também podem incluir requisitos oriundos da adoção de padrões e regulamentações. Já o **contexto do projeto** representa as circunstâncias de desenvolvimento, operação, políticas, além de outras que influenciam o sistema de alguma forma. Exemplos oriundos do contexto do projeto incluem, por exemplo, objetivos de negócio, características da empresa que irá desenvolver o sistema e o estado atual da tecnologia.

Assim que os **requisitos arquiteturalmente significativos** (RAS) são identificados, é possível dar inicio ao projeto arquitetural, que é a principal atividade do processo demonstrado na **Figura 1.1**. Durante essa atividade são tomadas as decisões arquiteturais como por exemplo a aplicação de um ou vários estilos ou padrões arquiteturais (e.g., camadas, cliente-servidor, *peer-to-peer*, orientação a serviços, etc.). O conjunto de decisões arquiteturais compreende as possíveis **soluções arquiteturais candidatas** para um conjunto de RAS. Tais soluções podem apresentar outras soluções alternativas que foram consideradas durante a atividade e também podem apresentar soluções para apenas uma

parte dos RAS. Além disso, as soluções incluem justificativas (*rationale*) para cada decisão tomada, ou seja, o porquê de cada decisão, quais decisões foram consideradas e rejeitadas, e a quais RAS essas decisões estão ligadas (rastreabilidade).

Finalmente, as soluções candidatas e os RAS são confrontados durante a **avaliação arquitetural** para garantir que a arquitetura final será adequada para cumprir com os requisitos do projeto. Sendo assim, o resultado dessa atividade é uma **arquitetura validada**.

Embora possa parecer que cada atividade seja realizada apenas uma vez seguindo a sequência que tem inicio na atividade de análise, passando pelo projeto e se encerrando na avaliação da arquitetura, o que ocorre na verdade é que a arquitetura é desenvolvida em vários momentos nos quais os arquitetos alternam entre essas três atividades [Hofmeister et al. 2007]. Isto ocorre pois as informações necessárias para construir uma arquitetura do zero nem sempre estão totalmente disponíveis a priori, como quando o processo de desenvolvimento segue alguma metodologia ágil, por exemplo SCRUM ou XP (eXtreme Programming). Além disso, diferentes forças podem atuar sobre o projeto, ocasionando em mudanças nos requisitos e tornando necessária a recondução de atividades arquiteturais.

1.2.2. Documentação e Representação Arquitetural

Embora todo sistema possua uma arquitetura, nem sempre ela é documentada. Por se tratar de um conceito abstrato, em muitos casos é impossível compreender a arquitetura de um sistema a partir de sua implementação (código-fonte). É por esse e outros motivos que arquitetos e engenheiros de software podem optar por documentar e representar a arquitetura do sistema enquanto este está em seu ciclo de vida útil, que se inicia na fase em que o sistema começa a ser desenvolvido até o momento em que se torna obsoleto e deixa de ser utilizado. A documentação da arquitetura pode incluir diversos itens que possuem alguma relação com as atividades do modelo apresentado na **Figura 1.1**. Informações sobre o contexto, restrições, RAS, resultados de avaliações e a representação da arquitetura são, por exemplo, alguns desses itens. A representação, em especial, é um tipo de documentação no qual a arquitetura de um sistema é representada por meio de diagramas ou modelos, permitindo que a arquitetura seja visualizada.

Há diferentes maneiras de representar uma arquitetura, havendo até mesmo um padrão internacional que define o conceito de representação arquitetural [ISO/IEC/IEEE 2011]. Contudo, o que há de comum entre várias dessas maneiras é o conceito de visão (do Inglês *views*). Como a arquitetura de um sistema geralmente possui várias estruturas e cada estrutura pode apresentar diferentes características e relações com outras estruturas, é impossível representar a arquitetura de um sistema com apenas um modelo ou diagrama. Fazendo analogia com o desenho arquitetônico de edificações, é possível representar uma casa a partir de diferentes perspectivas (ou visões), como pode ser visto na **Figura 1.2**, que apresenta como apartir dessas visões pode-se ter um entendimento "completo" da edificação. De maneira mais clara, não é possível entender completamente o edificio utilizando somente uma visão pois a informação representada é limitada.

No contexto de software, as visões também são criadas para destacar partes especificas do sistema ou de sua arquitetura. A partir delas é possível compreender de

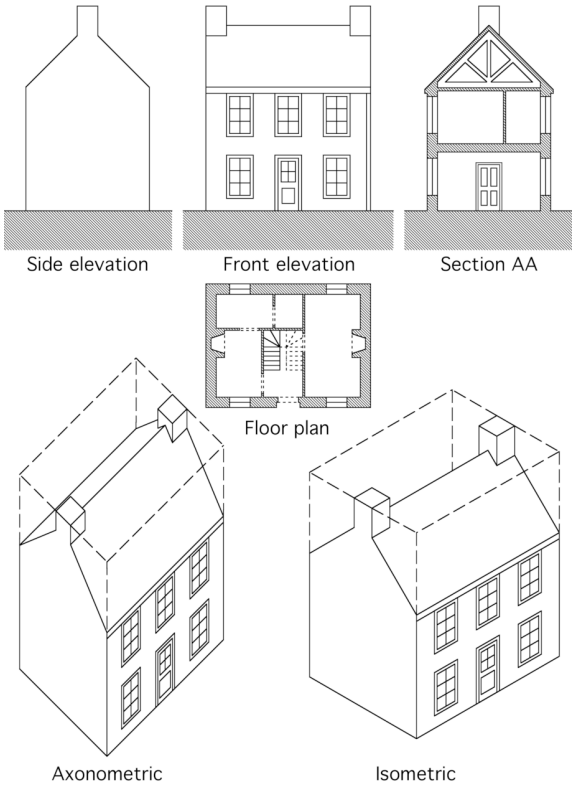

Figura 1.2. Visões comumente utilizadas no desenho arquitetônico de edificações. Fonte: Wikimedia Commons (Licença: CC BY-SA 3.0)

maneira mais clara quais são as estruturas que fazem parte da arquitetura, qual é a relação que existe entre elas e como tal organização pode favorecer o processo de avaliação da arquitetura sob a perspectiva de um atributo de qualidade, por exemplo. Na literatura de arquitetura de software existem várias propostas de visões arquiteturais que podem ser empregadas para a representação arquitetural. Dentre essas abordagens estão o modelo "4+1" [Kruchten 1995], "Views and Beyond" [Clements et al. 2010] e o conjunto de visões de "Rozanski and Woods" [Rozanski and Woods 2011]. As **Figuras 1.3**, **1.4** e **1.5** apresentam três exemplos de visões que são comuns em várias das abordagens existentes para obter um conhecimento (quase) "completo" da arquitetura de um sistema de software.

Como o termo módulo é um dos mais utilizados para indicar elementos que compõe uma arquitetura, nada mais justo do que apresentá-los em uma visão. A **Figura 1.3** apresenta o que é talvez um dos tipos de visão mais populares para descrever arquiteturas de software. A **visão de módulos** apresenta a decomposição de um sistema em partes menores denominadas módulos. Cada módulo representa um conjunto de estruturas de software como programas menores, classes de uma linguagem de programação orientada a objetos, *scripts*, vários arquivos de código-fonte, etc., que possuem uma determinada responsabilidade no contexto do sistema. Tais módulos podem ser agregados em outros maiores ou decompostos em módulos menores. Visões de módulo são úteis para [Clements et al. 2010] (i) a construção do sistema já que definem suas partes menores e permite que o desenvolvimento seja realizado de maneira paralela e distribuída; (ii) aná-

lise de impacto e rastreabilidade, já que é possível identificar quais módulos podem ser afetados caso seja necessário adicionar alguma funcionalidade ou realizar algum procedimento de manutenção, além de permitir o mapeamento entre requisitos e módulos que irão fornecer as funcionalidades para satisfazê-los; e (iii) comunicação entre *stakeholders*, já que é possível explicar as funcionalidades do sistema para novo integrantes do time de desenvolvimento.

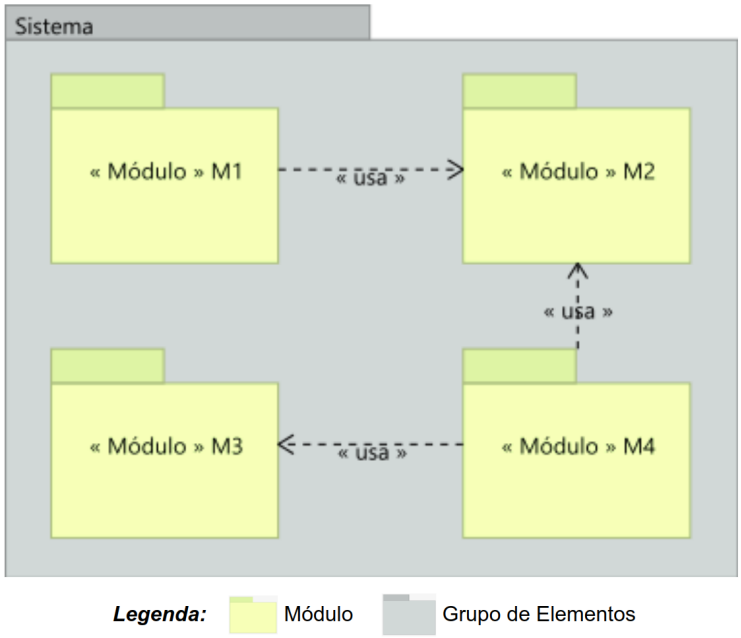

Figura 1.3. Exemplo de Visão de Módulos: o Sistema é decomposto em quatro módulos que apresentam relações de uso entre si. Alterações no módulo M2 podem afetar M1 e M4, enquanto que alterações em M3 pode afetar apenas M4.

Outro tipo bastante conhecido de visão arquitetural é a **visão de módulos em camadas**. Nesse tipo de visão os módulos são agrupados em diferentes camadas que representam algum critério definido pelo arquiteto. A **Figura 1.4** representa um sistema que possui sete módulos distribuídos em três camadas. Os módulos M1 e M2 compartilham alguma característica que os faz serem agrupados na camada C1. O mesmo é válido para os módulos de M3 a M7. Embora possa haver distinções da definição original, um sistema cuja arquitetura segue o padrão de camadas deve respeitar uma restrição importante: camadas superiores só podem acessar as funcionalidades providas pela camada imediatamente abaixo (ou acima, embora menos comum). Isso quer dizer, por exemplo, que os módulos da camada C1 só podem acessar funcionalidades da C2, e os da C2 só possuem acesso aos da C3. Essa relação entre camadas também deve ocorrer de maneira unidirecional, ou seja, a camada C1 pode acessar a C2, mas o contrário não é permitido. Essas regras são importantes para que as propriedades de uma arquitetura em camadas, tais como modificabilidade e portabilidade sejam preservadas. Isso ocorre pois cada camada é responsável por ocultar detalhes de sua implementação, fornecendo às camadas superiores apenas uma interface de suas principais funcionalidades. Deste modo, mudanças numa camada inferior não irão impactar as camadas superiores caso as regras tenham sido respeitadas [Clements et al. 2010].

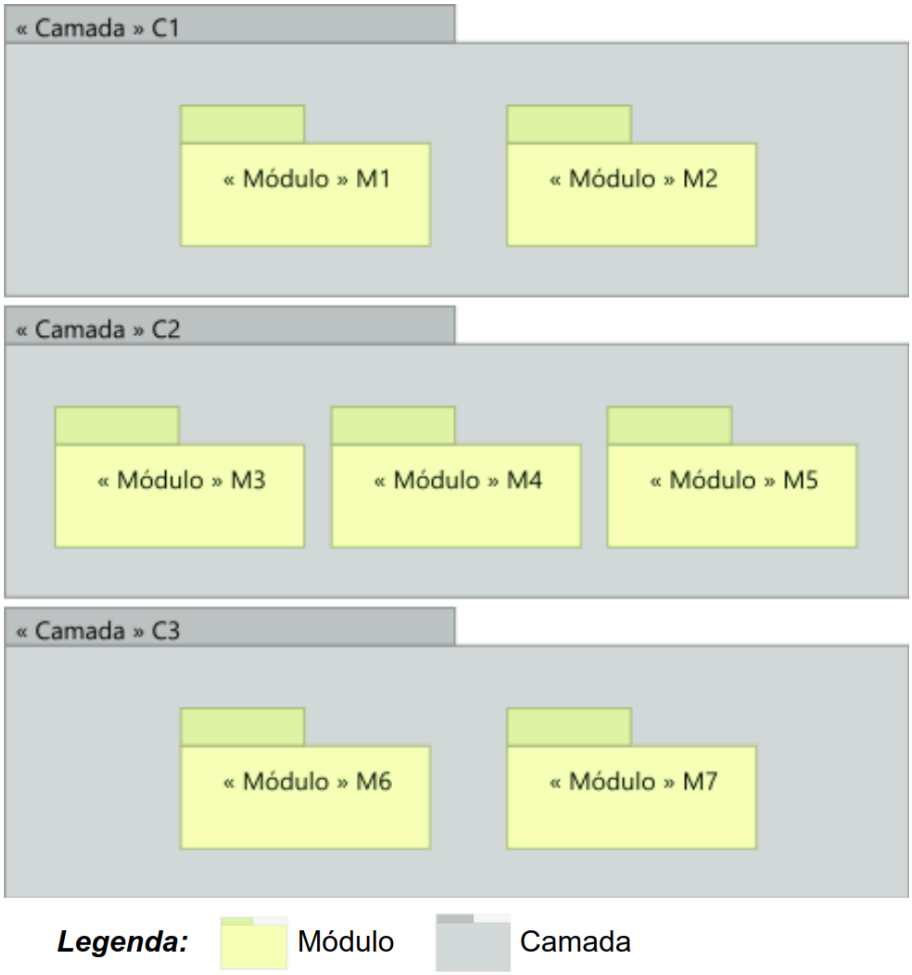

Figura 1.4. Exemplo de visão de módulos em camadas: O Sistema é decomposto em três camadas que agregam módulos de acordo com algum critério. As restrições de acesso entre as camadas garantem maior modificabilidade e portabilidade do sistema.

Por fim, um último exemplo de visão é a **visão física**. Nesse tipo de visão diferentes partes do sistema — sejam elas módulos, componentes, serviços ou outras — são alocadas às unidades físicas que serão responsáveis por sua execução e podem representar, por exemplo, servidores de aplicação, contêineres de computação em nuvem, além de outros tipos de *hardware* como *switchs* e roteadores. Essas unidades são denominadas nós e podem estar conectadas entre si de acordo com a comunicação que deve ocorrer entre as diferentes partes do sistema. A **Figura 1.5** apresenta um exemplo de visão física de um sistema que é distribuído em três nós principais. Nesses, são alocadas uma aplicação e três serviços, representando unidades de software em tempo de execução que fornecem algum tipo de funcionalidade. As conexões entre os nós demonstram que há comunicação entre eles por meio de um ou mais protocolos como HTTPS ou SFTP (*Secure File Transfer Protocol*). Esse tipo de visão é útil para realizar a análise da arquitetura em relação à atributos de qualidade como desempenho, disponibilidade, confiabilidade e segurança [Clements et al. 2010]. No exemplo da figura, a alocação do serviço S3 a um nó individual pode ser justificada, por exemplo, pela necessidade de recursos extras

(e.g., *hardware* dedicado ou específico) para sua execução. Os protocolos utilizados para a comunicação entre os nós não estão representados na figura, mas poderiam ser úteis para analisar o desempenho e a segurança do sistema. Além disso, caso fosse necessário aumentar a disponibilidade do sistema por conta de um aumento da demanda pelos serviços (por exemplo, vários usuários novos irão começar a utilizar o sistema), os serviços poderiam ser replicados entre mais nós para dar conta de atender à nova demanda.

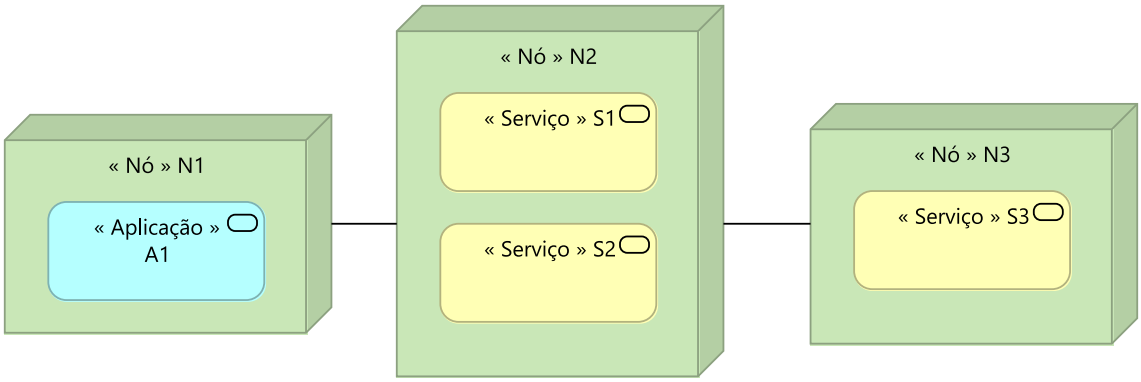

Figura 1.5. Exemplo de visão física: os Serviços do sistema são alocados a três nós físicos. O Nó N1 é capaz de se comunicar por meio de algum protocolo ao N2, que por sua vez é capaz se comunicar com o N3.

1.2.3. Qualidade de Software

Cada uma dessas visões pode ajudar a compreender como que a arquitetura de um sistema permite ou não que os requisitos de qualidade sejam atendidos. Contudo, há outras qualidades além das que foram discutidas nos exemplos anteriores, e para entendê-las é importante compreender o conceito de qualidade de software.

Existem diversos modelos na engenharia de software que objetivam definir o que é qualidade de software, já que essa tem sido um objetivo importante da disciplina desde sua criação. Em geral, esses modelos definem um conjunto de características (ou atributos) de qualidade que em sua totalidade descrevem o que é a qualidade de um sistema de software. Seria como definir, por exemplo, que um possível conjunto de características de um carro são conforto, eficiência, estabilidade e segurança, e que diferentes carros se diferem quando comparados em relação a esses atributos. Contudo, sabemos que nesse caso há outras características que devem ser considerados. No contexto de software os modelos visam descrever todos os atributos de qualidade possíveis ou pelo menos boa parte deles. Um dos modelos mais recentes é a norma ISO/IEC 25010 [ISO/IEC 2011], que decompõe o conceito de qualidade em duas perspectivas, que por sua vez são compostas por características e sub-características de qualidade. A primeira perspectiva, denominada qualidade de produto, define as qualidades relacionadas ao produto de software e é composta por oito características — *functional suitability, performance efficiency, compatibility, usability, reliability, security, maintainability*, e *portability* —, sendo cada um delas decompostas noutras várias sub-características. A segunda perspectiva é denominada qualidade em uso, e define características de qualidade que refletem o grau com que usuários específicos do sistema conseguem alcançar objetivos específicos em contextos de uso específicos. Essa perspectiva é composta pelas características *effectiveness, efficiency,*

satisfaction, freedom from risk, e *context coverage*, sendo cada uma decomposta em sub-características, com exceção de *effectiveness* e *efficiency*. Cada uma das características e sub-características são formalmente[1] definidas pela norma.

Alguns desses atributos, que serão abordados na **Seção 1.5**, são os seguinte: *performance*, *availability*, *interoperability*, *security* e *usability*:

Desempenho (*Performance efficiency*) é definido pela norma como o desempenho relativo à quantidade de recursos utilizados sob determinadas condições, e possui três sub-caracteristicas: *time-behaviour*, *resource utilization* e *capacity*. *Time-behaviour* está relacionado ao tempos de resposta e processamento do sistema durante sua operação, *Resource utilization* com à quantidade recursos utilizados pelo sistema e *Capacity* com os limites de operação do sistema.

Interoperabilidade (*Interoperability*) se refere à capacidade de dois ou mais sistemas ou componentes de trocar informação e utilizar essa informação que foi trocada de maneira correta para um propósito específico.

Usabilidade (*Usability*) pode ser referir à diversos fatores, como por exemplo a facilidade com que um usuário consegue aprender a utilizar o sistema (*learnability*) bem como operá-lo (*operability*), a capacidade do sistema de impedir que o usuário cometa erros por engano durante sua operação (*user error protection*) e também permite que o usuário use o sistema de maneira agradável e satisfatória (*user interface aesthetics*).

Disponibilidade (*Availability*) representa a capacidade de um sistema de estar acessível para uso sempre que necessário.

Segurança (*Security*) representa a capacidade de um sistema de proteger seus dados e informações, ao mesmo tempo em que garante acesso aos seus usuários ou outros sistemas autorizados.

1.2.4. Decisões Arquiteturais

Para atingir tais características de qualidade num determinado grau e dessa forma atender aos requisitos do sistema, é necessário que a arquitetura apresente as estruturas adequadas. Para isso, os arquitetos contam com diferentes estratégias que são empregadas para otimizar determinadas características, como no caso das táticas arquiteturais. Também podem adotar determinados padrões arquiteturais, que definem de maneira mais ampla como a arquitetura será organizada. Contudo, melhorar uma das características de qualidade de um sistema pode implicar em piorar outra característica. Essa consequência é conhecida dentro do projeto arquitetural como *trade-off* e ocorre como resultado de uma decisão tomada. Por fim, de maneira ainda mais ampla, é possível se basear em arquiteturas de referência para garantir que determinadas qualidades estejam presentes na arquitetura. Todas essas estratégias fazem parte do projeto arquitetural e podem ser consideradas decisões arquiteturais e são descritas a seguir.

Táticas arquiteturais são decisões de projeto que influenciam no cumprimento de um atributo de qualidade específico, afetando positivamente na resposta do sistema

[1] A definição de todas as características e sub-características podem ser consultadas em https://www.iso.org/standard/35733.html

quando sujeito a um estímulo relacionado com tal atributo [Bass et al. 2012]. Em geral, táticas arquiteturais podem ser classificadas de acordo com diferentes objetivos relacionados ao atributo de qualidade.

Considerando a características de segurança, por exemplo, é comum empregar táticas para detectar, resistir e reagir a ataques, invasões ou acessos não autorizados aos dados do sistema ou para executar operações não permitidas. Algumas das táticas para evitar tais ações são identificação, autenticação e autorização de usuário, limitação do acesso e exposição dos dados, além do emprego de técnicas de encriptação de dados [Bass et al. 2012]. Para aplicar essas táticas, é importante que o arquiteto identifique quais são os dados que possuem diferentes níveis de sensibilidade para que eles possam ser considerados na definição da arquitetura, além de garantir direitos de acesso a eles apenas para usuários propriamente autenticados e autorizados.

Para interoperabilidade, algumas das táticas existentes visam determinar a sintaxe e a semântica das principais abstrações de dados (conceitos) que deverão ou poderão ser trocadas entre os sistemas ou partes do sistema de maneira interoperável. Deve-se garantir que essas abstrações de dados sejam consistentes com os dados que foram trocados, caso contrário, a ausência de padronização pode tornar necessária a transformação dos dados antes que eles sejam trocados e empregados para algum propósito [Bass et al. 2012].

Bass et. al [Bass et al. 2012] listam diversas táticas que podem ser aplicadas por arquitetos para garantir outras características de qualidade além de segurança e interoperabilidade, como disponibilidade, desempenho, testabilidade, manutenibilidade e usabilidade.

Os **padrões arquiteturais**, por outro lado, definem estruturas de projeto arquitetural (soluções arquiteturais) bem estabelecidas na prática e que podem ser facilmente reutilizadas. Um padrão arquitetural especifica [Buschmann et al. 1997]: (i) sua utilidade (RAS que resolve), (ii) tipos de elementos (componentes, serviços, módulos) que compõem a solução; (iii) tipos de relações (dependência, colaboração, controle); (iv) propriedades dos elementos (responsabilidades, funcionalidades); e (v) restrições sobre como os elementos se relacionam (comunicação unidirecional, bidirecional, eventos, mensagens).

Um padrão arquitetural possui uma utilidade específica, ou seja, atende a um conjunto pré-estabelecido de RAS, e são geralmente compostos por várias táticas arquiteturais. Para isso, descreve elementos arquiteturais (e.g., módulos, componentes, e serviços) que compõem a solução, bem como suas relações (dependência, colaboração, controle), propriedades (responsabilidades, funcionalidades) e restrições (comunicação unidirecional, bidirecional, eventos, mensagens) [Buschmann et al. 1997].

Alguns dos principais exemplos de padrões arquiteturais são: Camadas (*Layers*), Pipes e Filtros (*Pipes-and-Filters*), Cliente-Servidor (*Client-Server*), Modelo-Visão- Controle (*Model-View-Controller – MVC*) e algumas variações como, Arquitetura Orientada a Serviços (*Service Oriented Architecture – SOA*), Microserviços (*Microservices*), *Publish-Subscribe*, *Data Model*, *Blackboard*, *Broker*, *Shared Data*, *Peer-to-Peer* e *Blockchain*.

A aplicação de um padrão ou outro vai depender da utilidade que ele terá para ajudar o arquiteto a atender os RAS de sua arquitetura. Dessa forma, é de vital importância que os arquitetos tenham um bom conhecimento sobre quando ou não aplicar cada padrão.

Para entender melhor os padrões arquiteturais, é possível consultar as seguintes publicações na área de arquitetura de software orientada a padrões: [Richards 2015, Buschmann et al. 1997, Avgeriou and Zdun 2005, Gorton 2011].

Por fim, é possível reusar conhecimento sobre como arquitetar sistemas de software em um dado domínio de aplicação por meio de uma **arquitetura de referência**, que é um tipo de arquitetura de software abstrata [Nakagawa et al. 2014, Bass et al. 2012, Cervantes and Kazman 2016]. Entre os benefícios da adoção desse tipo de arquitetura tem-se [Nakagawa et al. 2014, Martínez-Fernández et al. 2013, Angelov et al. 2013]: (i) o impacto positivo na produtividade da equipe de desenvolvimento, já que o conhecimento arquitetural é compartilhado e reusado; (ii) a criação de soluções padronizadas e de qualidade para os sistemas de software do domínio; e (iii) aprimoramento da interoperabilidade e capacidade de evolução desses sistemas.

Uma arquitetura de referência pode definir um conjunto coerente de padrões e táticas que ao ser aplicados fornecem um projeto inicial para as arquiteturas de software de um tipo específico de sistema, como é o caso dos sistemas de saúde. As táticas podem ser utilizadas como complemento às soluções mais gerais propostas pelos padrões ou arquiteturas de referência, pois são uma estratégia para refinar o projeto arquitetural. Dessa forma, ao utilizar uma arquitetura de referência, o arquiteto pode agilizar o processo de tomada de decisões pois reusa as soluções propostas por essa arquitetura mais abstrata.

1.2.5. Trade-offs das Decisões Arquiteturais

Em algumas ocasiões, atender certos RAS empregando as estratégias descritas (táticas, padrões e arquitetura de referência) pode resultar em impacto negativo sobre outras características de qualidade. Por exemplo, para aumentar a qualidade de um sistema de software, mais recursos são necessários, aumentando o custo final do projeto. Durante o projeto de uma arquitetura de software, *trade-offs* representam uma situação comum na qual é necessário balancear as perdas em um ou outro atributo de qualidade (o que é causado pela introdução de uma decisão arquitetural) para obter ganhos em outros atributos que, de acordo com os RAS, são mais importantes. Decisões arquiteturais como arquiteturas de referência, padrões e táticas detalham seus impactos positivos em atributos de qualidade específicos e, às vezes, também especificam suas limitações em relação a outros atributos.

Durante a etapa de projeto arquitetural, cada vez que o arquiteto seleciona um padrão ou uma tática para definir a arquitetura de um sistema de software, ele está mudando a arquitetura como resultado da necessidade de alcançar um RAS [Bass et al. 2012]. Nessa perspectiva, os arquitetos devem determinar, avaliar e mitigar os *trade-offs* que são introduzidos pelas diferentes decisões arquiteturais realizadas. No entanto, tomar múltiplas decisões aumenta a dificuldade de compreensão e análise do impacto dessas decisões sobre os níveis de qualidade do sistema de software como produto final.

1.3. Construção de Arquiteturas de Software na Área de Saúde

Esta seção ilustra como que o processo arquitetural apresentado na seção anterior foi aplicado para projetar a arquitetura do *SIGEAN*, um sistema de software para gestão da divisão de anestesia do Hospital das Clínicas da Faculdade de Medicina da Universidade

de São Paulo, HCFMUSP.

O HCFMUSP é um complexo hospitalar público, inaugurado em 1944, com enfoque no aprimoramento da formação de estudantes e no fornecimento de assistência médica gratuita à população através do Sistema Único de Saúde (SUS). Atualmente, o HCFMUSP é formado por 2 hospitais e 7 institutos (i.e., Instituto Central - IC, Instituto de Ortopedia e Traumatologia - IOT, Instituto do Coração - INCOR, Instituto da Criança - ICR, Instituto de Psiquiatria - IPQ, Instituto de Radiologia - INRAD e Instituto do Câncer do Estado de São Paulo - ICESP).

A divisão de anestesia da FMUSP é responsável pela coordenação de: (i) procedimentos pré-operatórios, operatórios e pós-operatórios realizados nos 9 centros cirúrgicos do HCFMUSP; (ii) aproximadamente 4500 anestesias e 3000 consultas por mês; (iii) aproximadamente 200 leitos de UTI (Unidades de Terapia Intensiva), 90 salas de cirurgia e 40 salas para procedimentos diagnósticos sobre anestesia; e (iv) consultas e procedimentos realizados por mais de 300 especialistas médicos nos centros cirúrgicos.

A **Figura 1.6** apresenta o processo que foi executado para a definição da arquitetura do *SIGEAN*. Em comparação ao processo introduzido na **Figura 1.1**, foi conduzida uma etapa (análise do sistema) antes de realizar a análise arquitetural. Além disso, esclarecemos que a etapa de avaliação arquitetural está em processo de planejamento e, portanto, não será apresentada em detalhes neste capítulo.

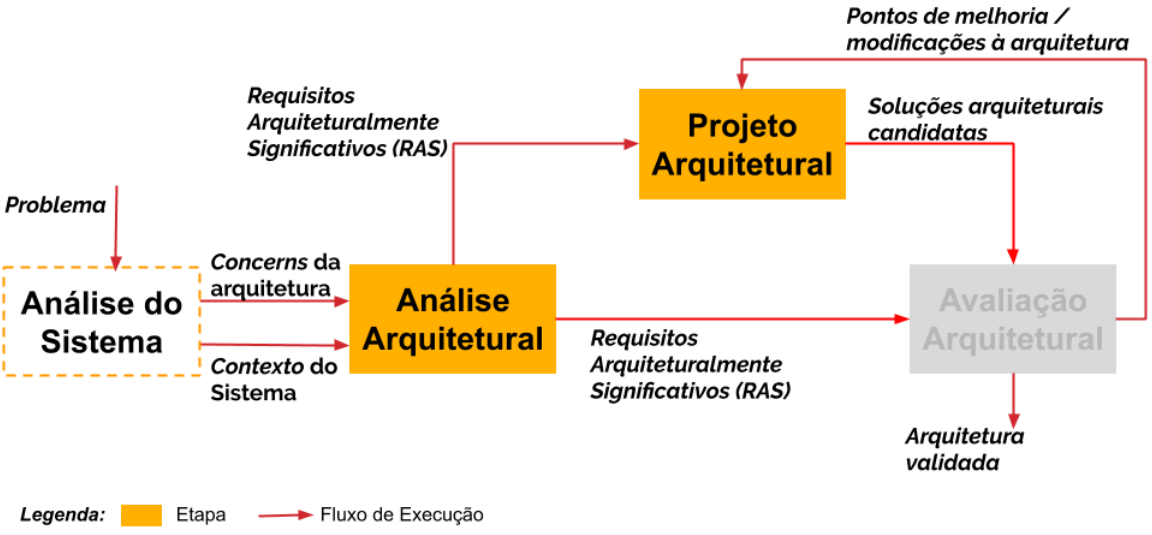

Figura 1.6. Processo de Construção de Arquiteturas de Software. Adaptado de [Hofmeister et al. 2007, Gorton 2011].

No restante do capítulo, iremos explicar como essas atividades foram conduzidas para a produção da primeira versão da arquitetura de software do sistema *SIGEAN*. Destaca-se que a execução das atividades arquiteturais apresentadas neste capítulo podem auxiliar à construção de diferentes arquiteturas de sistemas na área da saúde. Porém, cada arquiteto deve adaptar o processo para o sistema de saúde em questão.

Adicionalmente, ressalta-se que durante a apresentação do processo arquitetural no contexto do *SIGEAN*, apresentam-se diversos diagramas que foram realizados utilizando a ferramenta *Archi*®[2], que utiliza a linguagem *Archimate*® para realizar a descrição arquitetural e permite a modelagem semi-formal de arquiteturas de software.

1.4. Etapa 1 - Análise do Sistema

Esta primeira atividade teve como objetivo identificar e analisar o problema que o sistema deve resolver. Exemplos de problemas na área de saúde estão relacionados à execução de processos de planejamento, programação, regulação, controle, avaliação, auditoria da rede de atenção à saúde, como também desafios apresentados na prestação de serviços de saúde para cada especialidade (e.g., diagnóstico clínico, pronóstico, tratamento, intervenção, internação, observação e prevenção). Além disso, durante a atividade de análise de domínio devem ser estudados os *concerns* (i.e., preocupações ou necessidades) e a motivação que leva a organização a adquirir o sistema. Assim, se faz necessário investigar as restrições da organização como estrutura organizacional (hierarquias de setores e coordenações), infraestrutura física e de TI, recursos econômicos e humanos, legislações, protocolos de atendimento de saúde, entre outras. Além disso, é importante estabelecer os requisitos de operação e informação dos *stakeholders*, e as necessidades da organização para alcançar os objetivos de negócio (e.g., melhorar a qualidade no atendimento dos paciente, otimizar a utilização de recursos, minimizar custos e gastos, e aumentar a oferta de serviços de saúde).

1.4.1. Elicitação de Requisitos do Sistema de Software

Para conseguir entender claramente o domínio do problema, é importante que o arquiteto participe da identificação e especificação de requisitos do sistema, descrevendo:

- **Requisitos Funcionais (RF)** que descrevem como que o sistema deve funcionar, ou seja, as funções que o sistema deve executar, assim como os serviços que o sistema deve fornecer aos usuários e outros *stakeholders*; e

- **Requisitos Não Funcionais (RNF)** que caracterizam as restrições operacionais, ou seja, o comportamento do sistema em diferentes situações, como por exemplo o tempo de resposta, a disponibilidade de recursos, usabilidade, orçamento e cronograma.

Para identificar os requisitos do sistema a ser projetado, algumas técnicas de elicitação de requisitos podem ser aplicadas, como por exemplo:

- ***Estudos observacionais***, nos quais o arquiteto observa, sem intervir, usuários finais na realização das atividades e execução dos processos que apresentam algum problema, podendo identificar sistemas de software e outras tecnologias que são utilizadas pelos usuários durante as atividades;

[2]Ferramenta disponível em `https://www.archimatetool.com/`

- ***Entrevistas*** com usuários finais e outros *stakeholders*, nas quais são realizadas discussões ativas para identificar processos, atividades, tecnologias, sistemas, necessidades e preocupações dos usuários finais.

- ***Análise de documentos*** como protocolos, diretrizes, legislações necessárias para a execução dos processos e atividades e identificação de documentos utilizados e gerados pelos usuários.

Algumas diretrizes para o planejamento, execução e análise de resultados em estudos observacionais e entrevistas podem ser consultadas em [Wohlin et al. 2012, Sommerville 2011].

Na fase de elicitação de requisitos para o sistema *SIGEAN*, foram conduzidos estudos observacionais nos centros cirúrgicos do ICESP e IC do HCFMUSP. Em cada instituto, o arquiteto do sistema acompanhou durante um dia as anestesistas na realização de suas atividades rotineiras no centro cirúrgico. Destaca-se que em casos como esse em que se tem acesso a processos com pacientes, o arquiteto deve lidar com informações sigilosas e manter a privacidade das informações do paciente e dos profissionais de saúde envolvidos no estudo observacional. Em muitas situações, pode ser necessário fazer um planejamento detalhado do estudo observacional para ser previamente analisado e aprovado por um comitê de ética [3]. Durante os estudos observacionais, o arquiteto dedicou-se à identificação de problemas na geração e uso de informações e utilização de tecnologias, equipamentos, e sistemas já implantados nos institutos. Também atentou-se a situações que poderiam estar gerando frustração e sobrecarga de trabalho nos profissionais no momento de executar suas atividades, como por exemplo no uso de tecnologias difíceis de manipular, preenchimento exaustivo de fichas em papel, replicação de informação em diferentes documentos, além de outros problemas. As dificuldades encontradas foram relatadas pelo arquiteto em um documento mantendo o sigilo e privacidade dos pacientes e profissionais. Adicionalmente, o arquiteto identificou alguns ***stakeholders*** do sistema *SIGEAN*: coordenadores de centro cirúrgico, coordenador da divisão de anestesia do HCFMUSP, residentes de anestesia, chefes de anestesia, residentes de cirurgia, chefe de cirurgia, enfermeiros, técnicos de enfermagem, técnico de radiologia e alunos, que são modelados na **Figura 1.7**. O estudo observacional serviu para compreender os problemas *in situ* que os *stakeholders* enfrentam em seu dia a dia.

Um segundo passo para melhorar o entendimento sobre os problemas foi o planejamento e execução de entrevistas com alguns dos *stakeholders* identificados. No planejamento das entrevistas, foram elaborados questionários para cada perfil de *stakeholder*, sendo cada um estruturado em três conjuntos de perguntas para obter informação sobre: (i) a rotina de cada *stakeholder* desde seu ingresso ao centro cirúrgico até o término do plantão; (ii) problemas, dificuldades e frustrações que ele vivencia no dia a dia para cada atividade que realizada; (iii) como que a partir da perspectiva de cada *stakeholder* os problemas podem ser solucionados e como que cada atividade pode ser melhorada. Um exemplo de formulário utilizado nesta fase pode ser consultado no **Anexo A**. Também foi elaborado um termo de confidencialidade para manter o sigilo dos entrevistados. O

[3] No caso do sistema *SIGEAN*, o foco do arquiteto não era o paciente mas sim o profissional, dessa forma não foi necessária a aprovação no comitê de ética.

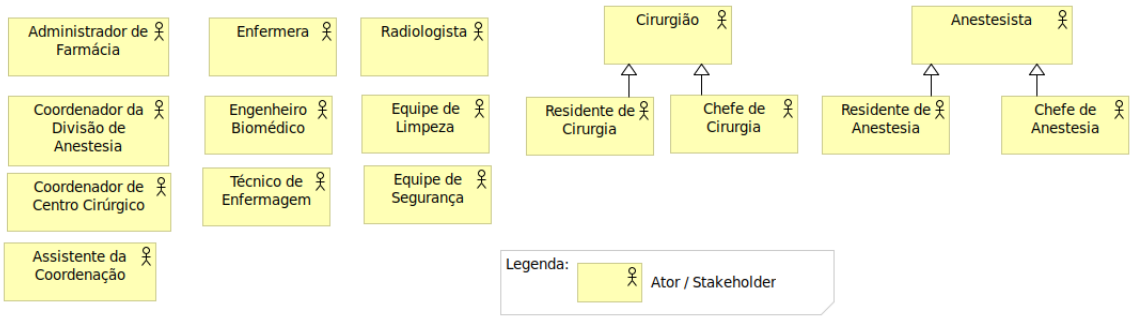

Figura 1.7. *Stakeholders* identificados para o sistema *SIGEAN*.

termo se encontra no **Anexo B**. Em seguida, foram agendadas entrevistas e com prévia autorização dos entrevistados. As entrevistas foram gravadas em áudio e posteriormente transcritas para texto.

Visando aprimorar o conhecimento do arquiteto sobre o domínio, foram identificados e solicitados aos *stakeholders* uma série de documentos que exigiram um estudo aprofundado dos processos, atividades, e informações do centro cirúrgico. Exemplo desses documentos são fichas de avaliação pré-operatória, anestesia, ocorrências durante o processo cirúrgico e de procedimentos pós-operatórios, além da legislação vigente relacionada ao processo cirúrgico. Cada documento apresenta um conjunto de informações que foram consideradas durante o projeto arquitetural do sistema *SIGEAN*.

1.4.2. Especificação de Requisitos do Sistema

Com o conhecimento obtido sobre o domínio através dos estudos observacionais, entrevistas e análise de documentos, deu-se inicio a especificação dos requisitos do sistema a ser projetado. Os requisitos podem ser formalizados através de diferentes modelos como os apresentados em [Pressman 2011]:

- *Modelos baseados em cenários,* definem os possíveis usos do sistema por parte dos *stakeholders*. Exemplos desses modelos são os modelos de casos de uso, histórias de usuários e PERSONAS;

- *Modelos de classes,* definem objetos do domínio que representam conceitos, dados e informações importantes para o funcionamento sistema. Esses modelos ajudam a especificar operações dos objetos e as relações e colaborações entre eles. Diagramas da UML como o de colaboração e o de classes podem ser utilizados para representar objetos do domínio;

- *Modelos comportamentais,* definem a maneira como os elementos externos podem mudar o estado do sistema. Alguns modelos usados para especificar o comportamento do sistema são os diagramas de estado e de sequência da UML; e

- *Modelos de fluxo,* estabelecem como as informações e dados são transformados durante a operação do sistema. Os diagramas de de fluxo de dados (DFD) e modelos de dados podem ser utilizados para entender a transformação das informações.

Pode não ser necessário utilizar todos esses tipos de modelos para especificar os requisitos do sistema. Devem ser escolhidos aqueles que melhor se adequem à natureza do projeto de desenvolvimento de software adotado pela equipe técnica.

Especificação de Requisitos de Usuário

No sistema *SIGEAN*, optou-se por utilizar o modelo PERSONAS para especificar os objetivos de cada tipo de ***stakeholder***. PERSONAS é uma abordagem vinda da área de Design Orientado a Usuários (UCD, em Inglês) e visa modelar usuários finais com necessidades e objetivos reais por meio de abstrações. As diretrizes fornecidas em [Adler 2005] foram seguidas para a especificação das PERSONAS com base nos resultados das entrevistas.

As **Figuras 1.8** e **1.9** apresentam dois modelos PERSONAS de usuários responsáveis pela coordenação do centro cirúrgico de um instituto e pela coordenação da divisão de anestesia do HCFMUSP, respectivamente. No modelo PERSONAS é possível destacar as motivações, necessidades e objetivos que esses usuários revelaram durante as entrevistas. Destaca-se que por ser uma abstração, os modelos PERSONAS não descrevem um usuário em particular, mas especificam um conjunto de usuários com os mesmos objetivos e necessidades. Dessa forma, o nome, a foto[4] e a descrição do usuário modelado na PERSONAS é abstrata, ou seja, não existe na vida real embora as necessidades e objetivos sejam reais.

Sofia (Coordenadora do Centro Cirúrgico)

Motivação: Otimizar recursos no centro cirúrgico

Descrição:
Sofia, 45 anos, casada, é uma médica anestesista com 19 anos de experiência. Atualmente é coordenadora de um centro cirúrgico do complexo HCFMUSP. Realiza atividades como a alocação de salas e distribuição de recursos dentro do centro cirúrgico.

Objetivos:

- Ter um sistema integrado para o trânsito de informações entre setores que interagem com o centro cirúrgico
- Ter um sistema de software que armazene todos os dados gerados durante os processos do centro cirúrgico, excluído o uso de fichas físicas e evitando o registro manual e a perda de fichas.
- Ter um sistema de software com interface mais amigável e maior facilidade de inserção de informações que não atrapalhe as atividades dos profissionais
- Facilitar a gestão do centro cirúrgico (escalamento de profissionais, gestão de leitos, etc.)

Figura 1.8. Exemplo de modelo de PERSONAS para um usuário do sistema *SIGEAN* com perfil de *Coordenação de Centro Cirúrgico*.

Como esses modelos são muito abstratos, é necessário refinar esses objetivos em requisitos de sistema mais detalhados que permitam alcançar os objetivos definidos para cada PERSONAS. Uma abordagem para derivar requisitos é perguntar ***DE QUE FORMA*** o objetivo pode ser atingido. Ao responder essa pergunta, o arquiteto obtém mais informações sobre como o objetivo pode ser cumprido, e pode assim derivar requisitos do sis-

[4]Fotos artificialmente geradas utilizando o site `https://generated.photos/`.

Paulo (Coordenador Geral)

Motivação: Otimizar recursos da divisão de anestesia do complexo HCFMUSP

Descrição:
Paulo, 65 anos, casado, é um médico anestesista com 30 anos de experiência.
Atualmente é coordenador geral da divisão de e anestesia no complexo HCFMUSP, precisando gerenciar seus 9 centros cirúrgicos.
Realiza atividades como a administração e distribuição de recursos entre os diversos institutos do complexo.

Objetivos:

- Ter um sistema integrado entre os institutos do complexo HCFMUSP para possibilitar o acesso às fichas de todos os pacientes.
- Ter um sistema com informações de recursos humanos e equipamentos para facilitar a gestão da divisão de anestesia do complexo HCFMUSP como um todo.

Figura 1.9. Exemplo de modelo de PERSONAS para um usuário do sistema *SIGEAN* com perfil de *Coordenador Geral*.

tema que sejam [Sommerville 2011]: (i) *corretos*, descrevendo atividades e tarefas sem erros; (ii) *precisos*, sem ambiguidade, evitando confusões ou interpretações errôneas; (iii) *completos*, especificando todas as necessidades importantes dos *stakeholders*; (iv) *consistentes*, evitando uso de diferentes termos para o mesmo requisito; e (v) *verificáveis*, possibilitando testar se o requisito é atendido ou não pelo sistema.

Por exemplo, para cumprir o segundo objetivo de Sofia (ver **Figura 1.8**), "*Ter um sistema de software que armazene todos os dados gerados durante os processos do centro cirúrgico, excluído o uso de fichas físicas e evitando o registro manual e a perda de fichas*", este deve ser refinado para entender melhor quais dados são gerados, em quais processos, e em quais fichas esses dados são registrados. Alguns requisitos derivados desse objetivo são:

- Durante o processo de avaliação pré-anestésica, o sistema *SIGEAN* deve permitir a captura digital e armazenamento das informações requeridas na ficha de avaliação pré-operatória estabelecida no protocolo de avaliação pré-operatória do HCFMUSP.
- Durante o processo intraoperatório, o sistema *SIGEAN* deve permitir a captura digital e armazenamento das informações requeridas no relatório de anestesia estabelecido nos protocolos para monitorização de procedimentos anestésicos intraoperatórios do HCFMUSP.
- Durante o processo intraoperatório, o sistema *SIGEAN* deve permitir a captura digital e armazenamento das informações requeridas no relatório de eventos adversos intraoperatórios da divisão de anestesia do HCFMUSP.
- Durante o processo pós-operatório, o sistema *SIGEAN* deve permitir a captura digital e armazenamento das informações requeridas no registro de procedimentos de recuperação pós-anestésica do HCFMUSP.

É sugerido que os requisitos derivados possuam identificadores para facilitar seu rastreamento nos modelos arquiteturais. Ou seja, é necessário identificar quais partes da arquitetura de software abordam um determinado requisito e verificar se de fato o objetivo do usuário está sendo considerado na arquitetura. Para isso, modelos de objetivos e derivação de requisitos como o apresentado na **Figura 1.10** podem ser elaborados pelo arquiteto.

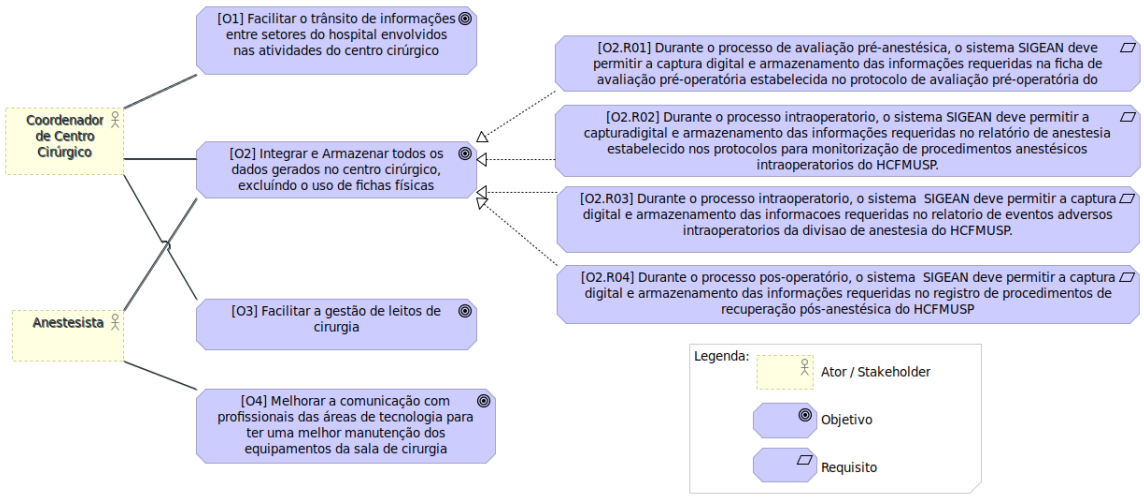

Figura 1.10. Modelo de derivação de requisitos com base nos objetivos dos usuários.

Especificação de Requisitos de Informação:

Adicionalmente, no sistema *SIGEAN*, o arquiteto utilizou modelos de classes da UML para especificar quais objetos precisam colaborar na construção de informações que são necessárias para executar corretamente os processos do centro cirúrgico. Por exemplo, assim que a cirurgia termina, o paciente é transferido para uma sala de recuperação pós-anestésica (RPA). Nessa sala, a equipe médica que é composta por dois profissionais, um anestesista e um enfermeiro, realiza procedimentos de avaliação pós-operatória do paciente. Para melhor compreender as informações envolvidas nesse processo, o arquiteto modelou as entidades de dados que precisam colaborar para construir o registro de recuperação pós-anestésica. O modelo é apresentado na **Figura 1.11**.

Como se observa no modelo de classes, uma unidade RPA faz parte de um único centro cirúrgico, que por sua vez pertence a um instituto específico. Ou seja, uma unidade RPA não pode estar associada a vários centros cirúrgicos nem a múltiplos institutos. Porém, um instituto pode possuir vários centros cirúrgicos e um centro cirúrgico pode ser composto por várias unidades RPA. A unidade RPA possui várias salas onde são executados procedimentos diagnósticos sob anestesia.

Segundo o modelo, podemos deduzir que o registro RPA deve conter informações sobre: (i) os profissionais de saúde (anestesistas e enfermeiros) que realizam o procedimento diagnóstico; (ii) a cirurgia que foi realizada no paciente, informações fornecidas pelo registro de cirurgia; (iii) avaliação do estado de saúde do paciente contida no registro de avaliação RPA; (iv) possíveis intercorrências acontecidas durante a avaliação RPA; e (v) dependendo do estado favorável de saúde, o registro de alta do paciente da unidade

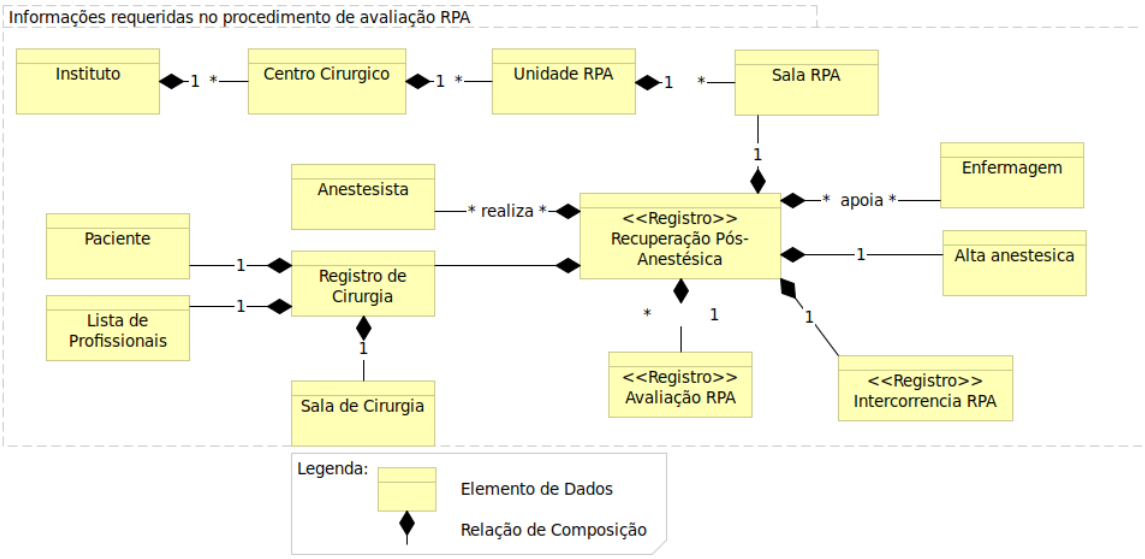

Figura 1.11. Uso do modelo de classes para especificar as informações necessárias para construir o registro de RPA.

RPA.

Destaca-se que o modelo apresentado na **Figura 1.11** foca-se nas informações que devem ser consideradas para o cumprimento do requisito [O2.R04] apresentado na **Figura 1.10**. Nesse sentido, esse modelo de classes apresenta somente as informações do registro RPA. Entretanto, para o sistema *SIGEAN*o arquiteto identificou mais registros e entidades de dados que são necessários para procedimentos pré-operatórios, durante a cirurgia, e depois do paciente receber alta da unidade RPA. Assim, arquitetos de sistemas de saúde devem considerar todas as entidades de dados relevantes para o cumprimento dos requisitos e objetivos do usuário e outros *stakeholders*.

1.5. Etapa 2 - Análise Arquitetural

Os requisitos identificados na etapa de análise do sistema precisam ser estruturados em grupos ou elementos de software como por exemplo, módulos, componentes, serviços ou até mesmo outros sistemas. Cada elemento de software deve agrupar (ou atender) conjuntos de requisitos relacionados a um objetivo específico. Tais agrupamentos podem ser feitos, por exemplo, de acordo com objetivos como: (i) a coordenação de atividades de um determinado processo; (ii) o gerenciamento das informações geradas na execução dos processos e que são necessárias para efetuar outros procedimentos ou atividades; (iii) a gestão de informações de usuários; (iv) o gerenciamento das informações do paciente; ou (v) a coordenação das atividades de um setor específico do hospital, por exemplo farmácia ou divisão de enfermagem.

Para o sistema *SIGEAN*, foram especificados como elementos de software treze serviços que são fornecidos por sete sistemas diferentes dentro do complexo HCFMUSP, conforme mostrado na **Figura 1.12**.

Para cada uma das partes do sistema *SIGEAN*, diversos requisitos foram alocados. A **Figura 1.13** oferece um exemplo de alocação dos requisitos previamente definidos no

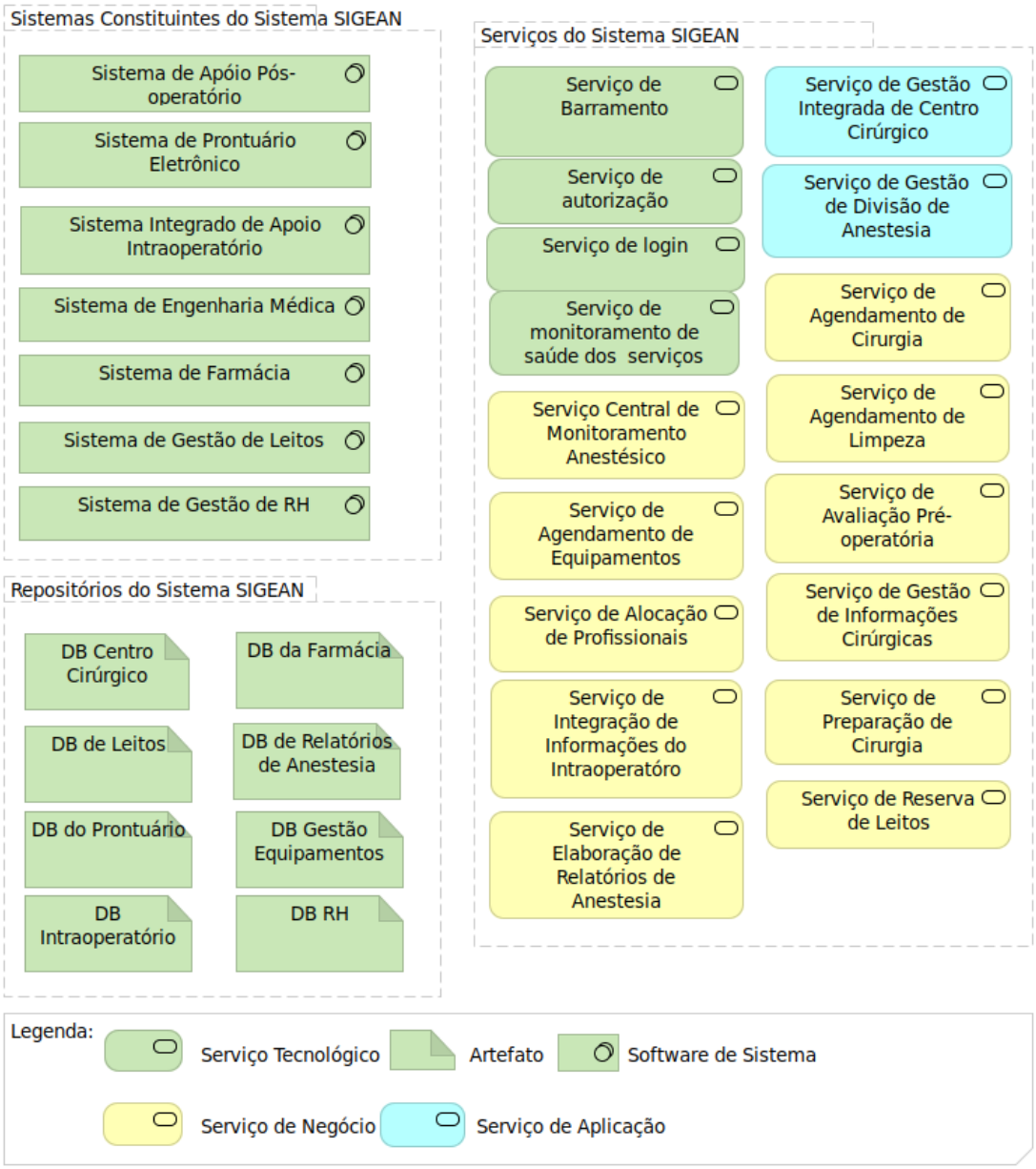

Figura 1.12. Sistemas, Serviços e Repositórios que Constituem o Sistema *SIGEAN*.

modelo da **Figura 1.10**.

Na **Figura 1.13** é possível observar que são necessários três sistemas para abordar os quatro requisitos derivados do objetivo de usuário [O2]: o sistema de prontuario eletrônico, o sistema integrado de apoio intraoperatório, e o sistema de apoio pós-operatório. Além desses, é necessário um serviço adicional responsável por elaborar todos os relatórios que devem estar disponíveis no centro cirúrgico e assim atingir o objetivo [O2].

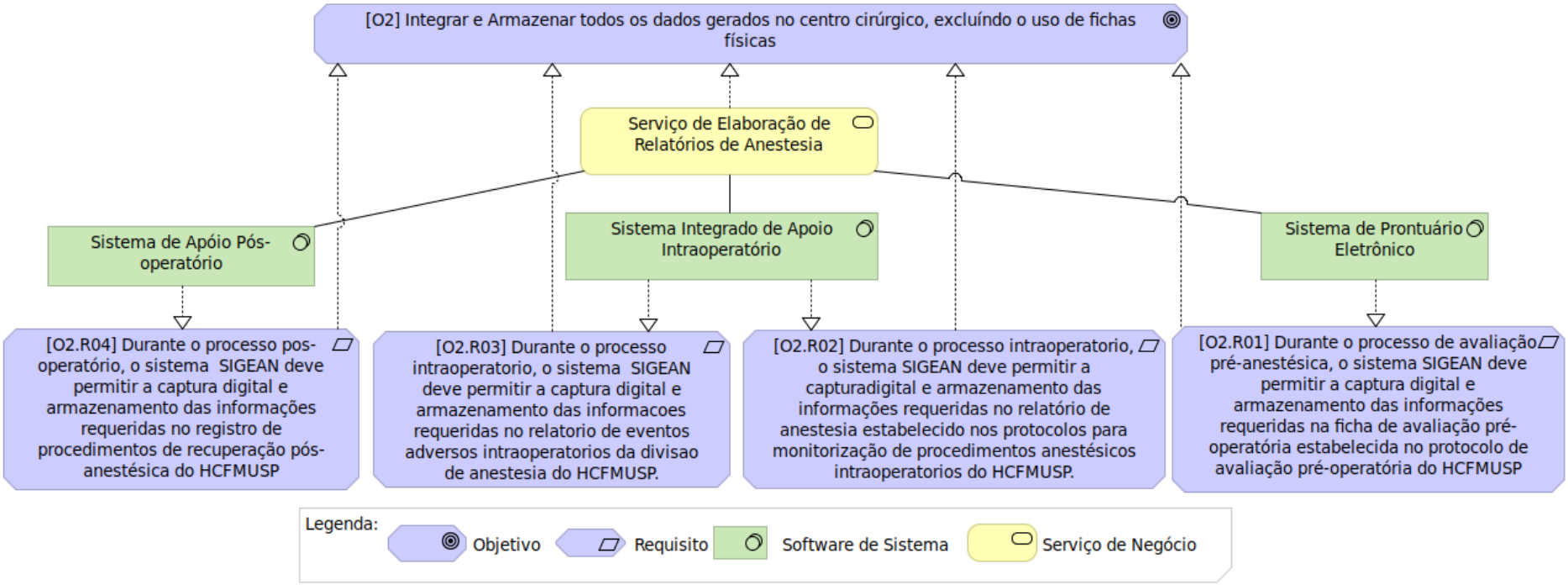

Figura 1.13. Exemplo de modelo de alocação de requisitos a elementos de software.

O procedimento de alocação de requisitos em entidades de software deve ser realizado até que cada requisito especificado para o sistema tenha pelo menos uma entidade responsável por seu cumprimento. É importante ressaltar que uma entidade pode estar comprometida com a execução de vários requisitos, como é o caso do sistema integrado de apoio intraoperatório, responsável pelos requisitos [O2.R02] e [O2.R03].

1.5.1. Análise de Requisitos Arquiteturalmente Significativos (RAS)

Durante a análise do sistema, não basta somente definir ***O QUE*** o sistema deve realizar (requisitos funcionais), ***QUEM*** é o responsável por executar o requisito (entidade de software), nem ***PORQUE*** o requisito deve ser cumprido (objetivo). Ainda é necessário perguntar-se ***COMO*** as partes do sistema (ou elementos de software) devem ser organizadas e quais são os níveis de qualidade que devem ser garantidos durante a execução dos requisitos funcionais. Dessa forma, antes de projetar um sistema de software, é necessário ter um bom entendimento requisitos arquiteturalmente significativos (RAS) [Gorton 2011].

É importante identificar previamente os RAS, idealmente durante a etapa de elicitação de requisitos, para projetar de forma bem sucedida a arquitetura de software de um sistema [Bass et al. 2012]. RAS se diferenciam de outros requisitos no sentido que estes têm efeitos profundos na arquitetura de software de um sistema, ou seja, a arquitetura poderia ser dramaticamente diferente na ausência de tais requisitos [Bass et al. 2012]. Os arquitetos devem ser cautelosos na escolha dos RAS, pois nem todos os requisitos são relevantes para a arquitetura de um sistema [Hofmeister et al. 2007]. A seleção dos RAS é uma tarefa desafiadora, pois esses normalmente não são capturados como requisitos do sistema na etapa de elicitação de requisitos. Os RAS podem surgir implicitamente a partir da análise de outro tipo de característica que terão um forte impacto na arquitetura, como por exemplo [Hofmeister et al. 2007, Richards and Ford 2020, Bass et al. 2012]: (i) necessidades dos stakeholders diferentes daquelas relacionadas às funcionalidades do sistema; (ii) objetivos de negócio, como satisfação de usuário, tempo de comercialização, fusões e aquisições de organizações, vantagem competitiva, valor para a organização, ou conformidade com a legislação ou regulamentação vigente; ou (iii) os contextos nos quais o sistema será desenvolvido e utilizado. Em [Cervantes and Kazman 2016, Bass et al. 2012] os autores oferecem mais detalhes sobre os diferentes tipos de RAS que podem surgir durante a análise arquitetural.

Normalmente, os RAS estão estreitamente relacionados com requisitos de atributos de qualidade [Bass et al. 2012, Gorton 2011, Richards and Ford 2020, Hofmeister et al. 2007] como desempenho, segurança, capacidade de modificação, disponibilidade, usabilidade, interoperabilidade, além de outros requisitos que a arquitetura deve atender. Portanto, quanto maior a dificuldade e importância do requisito de atributo de qualidade, maior a chance de que esse afete significativamente a arquitetura e seja considerado um RAS [Bass et al. 2012]. Os arquitetos podem identificar um grande número de requisitos de atributos de qualidade para suas arquiteturas, porém tentar atender todos esses requisitos adiciona complexidade ao projeto do sistema [Richards and Ford 2020]. Nesse sentido, uma atividade crítica do arquiteto é a escolha da menor quantidade possível de RAS, ou seja ele/ela deve priorizar os RAS conforme o nível de impacto que esses podem ter na arquitetura.

Uma prática para identificar RAS durante a etapa de elicitação de requisitos com os *stakeholders*, é perguntar-se como que as funcionalidades do sistema devem ser entregues aos usuários finais. A **Tabela 1.1** apresenta algumas perguntas interessantes que podem ser consideradas pelo arquiteto durante esta fase de especificação de RAS. Perguntas adicionais para cada atributo de qualidade podem ser consultadas em [Wiegers and Beatty 2013, Richards and Ford 2020].

Tabela 1.1. Exemplos de perguntas para identificar RAS

Atributo de Qualidade	**Possíveis Perguntas**
Disponibilidade	Quanto tempo o sistema precisa estar disponível? Quais partes (elementos) do sistema precisam estar disponíveis 24/7? Qual o tempo máximo que o sistema (ou partes dele) podem ficar indisponíveis? Por quanto tempo as informações, elementos de software, ou uma funcionalidade deve estar disponível para seu uso? Qual o tempo mínimo aceitável para executar uma funcionalidade do sistema? Quais são os elementos que serão mais utilizados pelos usuários?
Capacidade de recuperação	En caso de falha ou desastre, que tão rápido (em ms) o sistema deve estar disponível? Que falhas podem acarretar à organização perda de grande sumas de dinheiro ou afetarão vidas humanas?
Desempenho	Qual o tempo mínimo aceitável para executar uma funcionalidade do sistema? A cada quanto tempo é necessário consultar, salvar, modificar, eliminar informação? Quantas operações são executadas por uma entidade em um tempo determinado? Quais informações serão consultadas com maior frequência? quantos acessos ou solicitações devem ser processadas por unidade de tempo?
Escalabilidade	Qual a estimativa de usuários do sistema a curto, médio e longo prazo?
Capacidade do sistema	Quanta informação é gerada pelo sistema? Quantos dados devem ser armazenados a curto, médio e longo prazo? Por quanto tempo as informações, elementos de software, ou uma funcionalidade deve estar disponível para seu uso? Quais dados precisam ficar arquivados ou podem ser deletados depois de um período de tempo?
Portabilidade	O sistema deverá ser executado em várias plataformas ou sistemas operacionais? O sistema deve interagir com diferentes SGBD (Oracle, Mysql, MongoDB, Firebase)? É necessário suportar múltiple linguagens para que os usuários possam interagir com o sistema?
Reuso	Quais partes do sistema serão utilizadas em outros produtos/serviços/sistemas?
Manutenção	A cada quanto tempo precisa ser modificado um elemento de software ou uma informação?
Facilidade de acesso	Existem usuários com deficiências (visuais, auditivas) que precisem de interfaces de usuário adaptadas?
Proteção contra erros do usuário	Quais informações são sensíveis de serem inseridas com erro pelo usuário?
Autenticação	Como garantir que os usuários são quem eles dizem ser? Qual tipo de autenticação é requerida para acesso remoto de usuários?
Autorização	Quais usuários (perfis / grupos de usuários) estão autorizados a executar as funcionalidades do sistema ou acessar os dados?
Segurança dos dados	É necessário cumprir alguma lei de proteção de dados? Quais leis de proteção de dados devem ser cumpridas? Quais direitos de propriedade intelectual devem ser considerados? Quais dados devem ser encriptados? Quais dados são sensíveis de serem modificados sem autorização? São necessários mecanismos de encriptação end-to-end? É requerido o uso de firewall local ou externo? É necessária a encriptação da rede de comunicação entre sistemas internos?
Coexistência	O sistema deve executar suas funcionalidades enquanto compartilha um ambiente operacional comum a outros sistemas?
Interoperabilidade	É necessário o compartilhamento de informações com outros sistemas? Quais informações são fornecidas por outros sistemas para o correto funcionamento do sistema? Quais informações devem ser compartilhadas com outros sistemas? Qual a semântica e sintaxe das informações a serem compartilhadas?
Corretude funcional	Quais operações ou funcionalidades devem ser 100% corretas para evitar falhas no sistema que possam trazer prejuízos econômicos ou ferir algum usuário? Qual a estimativa de erro aceitável na execução de uma operação no sistema?
Restrições	**Possíveis Perguntas**
Restrição do cliente	Quais as restrições de orçamento e cronograma impostas pelo cliente
Restrições de infraestrutura	Quais as restrições de infraestrutura impostas pelo cliente, organização onde irá ser executado o software ou organização que desenvolve o software?
Restrições da organização desenvolvedora do sistema	Qual o orçamento e tempo para desenvolver um elemento de software?

Podemos observar que essas perguntas estão estreitamente relacionadas ao conceito de atributos de qualidade como capacidade, desempenho, eficiência, disponibilidade, usabilidade, segurança, e manutenção, entre outros. O conceito de atributos de

qualidade foi introduzido na **Seção 1.2**. Adicionalmente, os RAS são diretamente influenciados pelas restrições colocadas pela organização ou cliente do sistema a ser projetado. Alguns exemplos de restrições são o tempo de entrega do sistema, orçamento disponível e o ambiente de produção onde o sistema irá a funcionar.

Algumas definições para os atributos de qualidade listados na **Tabela 1.1** são apresentadas na **Seção 1.2.3**. O arquiteto também pode fazer uso de modelos de qualidade genéricos como o ISO/IEC 25010 - SQUARE [ISO/IEC 2011], ou modelos de qualidade para o domínio de saúde como [Alves 2017, Garcés et al. 2016, Garcés et al. 2017] para ter um melhor entendimento dos conceitos para os diferentes atributos de qualidade que podem ser considerados como RAS.

No estudo de caso do sistema *SIGEAN*, o arquiteto aplicou algumas das perguntas da **Tabela 1.1** para identificar RAS. Exemplos desses requisitos são ilustrados no modelo da **Figura 1.14**. Os RAS foram identificados e associados aos objetivos previdamente especificados para o sistema *SIGEAN*. Para cada parte constituinte do sistema *SIGEAN* foram atribuídos RAS que definem como eles devem executar suas funcionalidades. Na **Figura 1.14**, podemos observar que os três sistemas devem cumprir RAS similares (Grupo 1 de RAS). Porém, o sistema integrado de apoio intraoperatório requer uma especificação detalhada dos RAS (ver Grupo 2 de RAS) devido à criticidade das informações geradas a partir da captura e armazenamento de grandes quantidades de informação sobre o estado de saúde do paciente durante uma cirurgia (e.g., batimento cardíaco, saturação de oxigênio e temperatura).

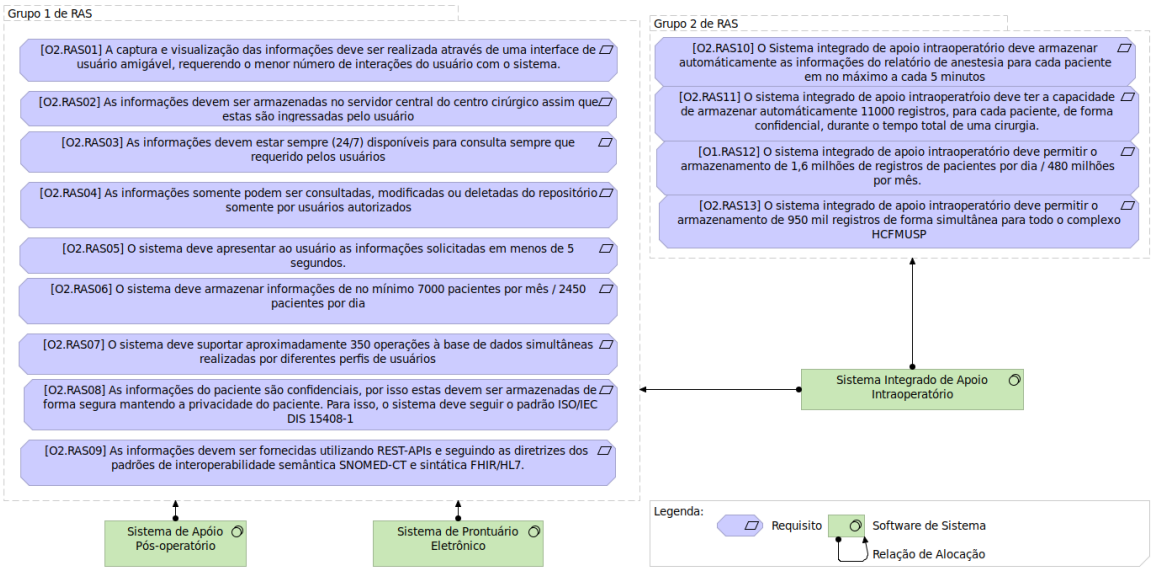

Figura 1.14. Exemplo de modelo de atribuição de RAS a partes que constituem o sistema *SIGEAN*.

A **Tabela 1.2** relaciona os atributos de qualidade considerados para a especificação de RAS para a arquitetura do sistema *SIGEAN*. Observa-se que os três sistemas (como partes do sistema *SIGEAN*) devem considerar atributos de qualidade como usabilidade, desempenho, disponibilidade, segurança, capacidade do sistema e interoperabilidade. Destaca-se ainda que, o sistema de integração de apoio interoperatório deve ser

projetado considerando requisitos de desempenho e capacidade com máxima prioridade.

Tabela 1.2. Atributos de Qualidade Relacionados aos RAS para a Arquitetura do Sistema *SIGEAN*.

Atributo de Qualidade	ID do RAS
Usabilidade	RAS01.
Desempenho	RAS02, RAS05, RAS10, RAS11.
Disponibilidade	RAS03, RAS07.
Segurança	RAS04, RAS08, RAS11.
Capacidade do Sistema	RAS06, RAS07, RAS11, RAS12, RAS13.
Interoperabilidade	RAS09

É importante destacar que, os requisitos para a arquitetura RAS08 e RAS09 foram definidos com base em regulamentação nacional para segurança e interoperabilidade de sistemas de software de saúde. É de grande importância que o arquiteto esteja sempre atualizado sobre esse tipo de regulamentação, pois o cumprimento desses requisitos irá impactar fortemente na certificação destes sistemas. No Brasil, a Sociedade Brasileira de Informática em Saúde (SBIS) é uma das organizações responsáveis pela certificação de Sistemas de Registro Eletrônico em Saúde (S-RES). Qualquer sistema de software que vise o armazenamento e gestão de informações na área de saúde deve cumprir os requisitos de segurança e interoperabilidade propostos pela SBIS em [Silva and Junior 2019] antes de seu uso por profissionais de saúde. Nessa perspectiva, o arquiteto deverá entender essa regulamentação e definir os RAS para que a arquitetura do sistema esteja em conformidade com os padrões impostos pelas diretrizes da SBIS.

1.6. Etapa 3 - Projeto Arquitetural

O objetivo da terceira etapa para a construção de uma arquitetura é gerar um projeto arquitetural que atenda aos RAS [Bass et al. 2012] identificados na etapa de análise arquitetural. A atividade de projeto da arquitetura deve ser planejada, intencional, racional e dirigida [Cervantes and Kazman 2016]. Essa atividade é desafiadora já que o arquiteto começa com "uma folha em branco" que deve ser preenchida com diversas possibilidades visando o cumprimento dos RAS. O preenchimento do projeto ou formação da arquitetura é realizado através da tomada de decisões que o arquiteto realiza durante o projeto. Essas decisões precisam ser modeladas de tal forma que possam ser claramente entendidas pelos diferentes *stakeholders* do projeto. Adicionalmente, é necessário documentar adequadamente o projeto para compartilhar e comunicar todas as informações da arquitetura requeridas por esses *stakeholders*. As atividades envolvidas dentro da etapa de projeto arquitetural são apresentadas na **Figura 1.15** e serão descritas a seguir.

1.6.1. Tomada de Decisões Arquiteturais no Sistema *SIGEAN*

A tomada de decisões é a atividade fundamental de um arquiteto de software, pois define: (i) como que as estruturas de um sistema de software serão organizadas; (ii) como que essas estruturas estarão relacionadas; e (iii) quais são as restrições necessárias para essas estruturas e tais relacionamentos. A tomada de decisões é de vital importância pois é através desse processo de raciocínio que o arquiteto deve garantir que os RAS da arquitetura sejam satisfeitos da melhor forma.

O arquiteto pode fazer uso de sua experiência para definir a arquitetura do sistema.

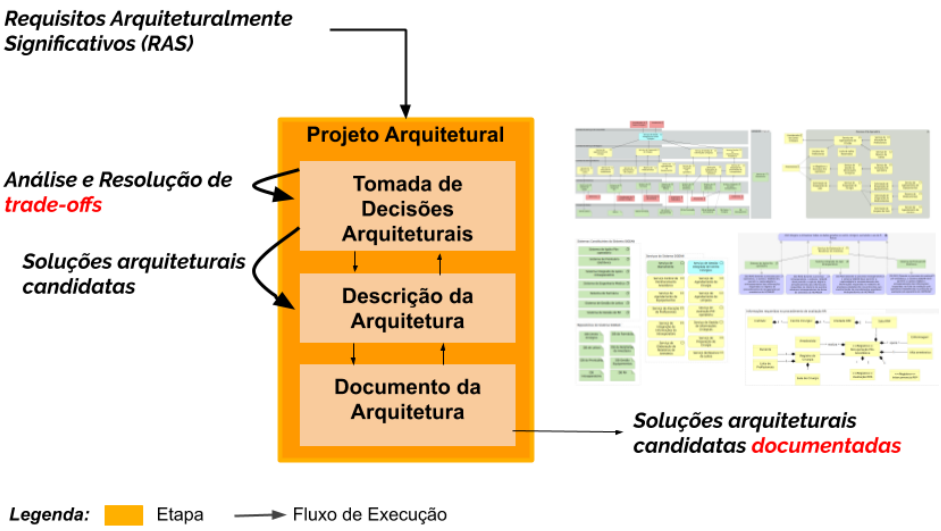

Figura 1.15. Atividades para Definir o Projeto de uma Arquitetura de Software.

Existem "boas práticas" que podem ser seguidas para criar arquiteturas de uma forma mais rápida e com certo nível de confiança de que a arquitetura irá atender os RAS. Essas práticas estão relacionadas ao uso de soluções arquiteturais existentes que já têm sido validadas previamente pela comunidade e que são bem conhecidas para solucionar RAS específicos. Dentre essas soluções destacam-se o uso de táticas arquiteturais, padrões arquiteturais (algumas vezes chamados de estilos) e arquiteturas de referência.

Táticas Arquiteturais

Para a arquitetura do sistema *SIGEAN* foram adotadas táticas para *segurança* (não repúdio, encriptação dos dados, autorização, e autenticação), *desempenho* (replicação e agendamento de recursos), *disponibilidade* (monitoramento do estado das partes do sistema para detectar falhas) e *interoperabilidade* (transformação de dados entre partes do sistema para realizar a troca de informações). Uma descrição detalhada dessas táticas pode ser encontrada nos capítulos 5, 6, 8 e 9 do livro [Bass et al. 2012].

Segundo as diretrizes fornecidas pela SBIS, os sistemas de software para saúde que pretendem atuar no âmbito nacional devem adotar os padrões ISO/IEC 27002:2013 [ISO/IEC 2013, ISO/IEC 2015] para segurança de informação, que já aderem às táticas arquiteturais de controle de acesso, encriptação, login e monitoramento de acesso. Nesse sentido, o arquiteto do sistema *SIGEAN* segue a série de padrões ISO/IEC 27002 como estratégia para implementar as táticas de segurança.

Com o intuito de promover a criação de sistemas de saúde interoperáveis, a SBIS recomenda adoção dos padrões HL7[5] e FHIR[6] para definir a sintaxe das informações tro-

[5] http://www.hl7.com.br/
[6] https://hl7.org/FHIR/

cadas entre sistemas, e os padrões LOINC[7], CID-10[8], SNOMED-CT[9] para identificar os conceitos e a terminologia que deverá ser utilizada pelos sistemas que irão trocar informações [Silva and Junior 2019].

Ao reusar os padrões previamente identificados pela SBIS e definidos por organizações internacionais de padronização, o trabalho do arquiteto deve estar focado em identificar e descrever quais elementos (serviços e entidades de dados) da arquitetura deverão implementar as táticas de segurança e interoperabilidade que esses padrões definem para os sistemas de saúde.

Arquiteturas de Referência e Padrões Arquiteturais

Para sistemas de saúde existem diversas arquiteturas de referência que podem oferecer uma estrutura arquitetural inicial em vários tipos de aplicação, como por exemplo, casas inteligentes para o cuidado de saúde [Garcés 2018], prontuários eletrônicos [The openEHR Foundation 2018, Cavalini and Cook 2014, Losavio et al. 2015], saúde conectada [Bandara 2015], ecossistema de e-saúde [Wartena et al. 2010], além de outros. Em [Garcés et al. 2015, Garcés et al. 2020] são apresentadas diversas arquiteturas de referência para projetar sistemas de software de saúde. Dentro desse conjunto de arquiteturas de referência, destaca-se a OpenEHR [The openEHR Foundation 2018] cujo uso é recomendado pela SBSI para arquitetar sistemas de prontuário eletrônico que pretendem certificação nacional [Silva and Junior 2019].

O arquiteto do sistema *SIGEAN* fez uma revisão dessas arquiteturas de referência e não achou alguma que poderia orientar o projeto arquitetural no domínio de salas de cirurgia, centros cirúrgicos ou processos de anestesia. Nesse contexto, o arquiteto optou por utilizar uma arquitetura de referência para sistemas orientados à serviços. Dessa forma, a arquitetura para o sistema *SIGEAN* é baseada na arquitetura de referência "Service-Oriented Solution Stack (S3)" [Arsanjani et al. 2007], que permite criar arquiteturas seguindo o padrão SOA. Entretanto, a S3 precisou ser refinada e adaptada ao contexto do sistema *SIGEAN*, levando o arquiteto a tomar decisões complementares como a adoção de outros padrões arquiteturais como o de camadas (*layers*), SOA, repositório, e barramento.

Resolução de Trade-offs

Como introduzido na **Seção 1.2**, a aplicação de arquiteturas de referência, padrões ou táticas arquiteturais para ajudar a cumprir RAS da arquitetura sob construção leva a necessidade de resolução de conflitos ou *trade-offs* entre atributos de qualidade impostos por essas soluções arquiteturais.

A **Figura 1.16** apresenta um exemplo dos possíveis *trade-offs* existentes entre alguns dos atributos de qualidade considerados RAS para a arquitetura do sistema *SIGEAN*. Esses *trade-offs* foram recuperados de [Wiegers and Beatty 2013]. Por exemplo, observa-

[7] `https://loinc.org/`
[8] `https://www.who.int/classifications/icd/icdonlineversions/en/`
[9] `http://www.snomed.org/`

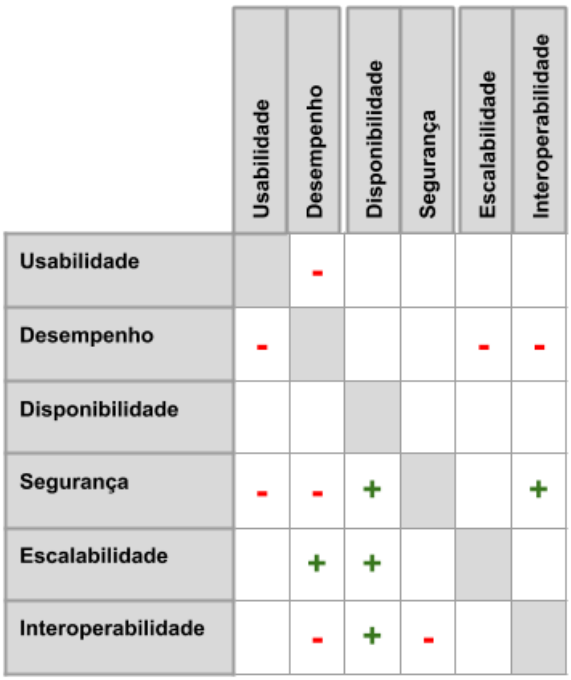

Figura 1.16. Possíveis trade-offs entre RAS para a arquitetura do sistema *SIGEAN*. Extraído de [Wiegers and Beatty 2013].

se que priorizando os RAS relacionados a desempenho o arquiteto terá que mitigar os possíveis efeitos negativos em RAS associados à usabilidade, escalabilidade (capacidade do sistema) e interoperabilidade. Por outro lado, ao priorizar RAS de escalabilidade ou capacidade do sistema, o arquiteto poderá contribuir positivamente também para o cumprimento de RAS de desempenho e disponibilidade do sistema.

A análise de *trade-offs* permitiu que o arquiteto priorizasse o conjunto de RAS associados aos atributos de escalabilidade, interoperabilidade e segurança, ao invés dos atributos de desempenho, disponibilidade e usabilidade. Essa decisão foi baseada no fato de que ao priorizar o primeiro conjunto de RAS, seria possível contribuir parcialmente para o cumprimento dos níveis de desempenho e disponibilidade. Para atingir o grau desejado de usabilidade o arquiteto deverá analisar possíveis alternativas que possibilitem um mínimo cumprimento desses RAS, visando sempre não prejudicar os outros RAS que foram priorizados.

É importante ressaltar que o processo de busca por um padrão, tática arquitetural ou arquitetura de referência adequado deve estar sempre acompanhado de uma análise aprofundada dos *trade-offs* que poderão surgir ao adotar essas soluções na arquitetura do sistema de software. Assim, a tomada de decisões é um processo sistemático, ciente e de grandes desafios para um arquiteto e deve ser realizado cuidadosamente pois decisões não adequadas podem repercutir negativamente no sucesso do sistema durante as etapas de desenvolvimento, teste, implantação, manutenção e evolução, entre outras.

Na próxima seção é explicado como que cada uma dessas estratégias arquiteturais foi utilizada para formar a primeira versão da arquitetura do sistema *SIGEAN*.

1.6.2. Descrição da Arquitetura

Conforme introduzido na **Seção 1.2.2**, o arquiteto precisa descrever de forma clara e coerente as decisões tomadas para construir a arquitetura do sistema. Nesse sentido, faz-se necessário apresentar a arquitetura como um conjunto de visões. Para descrever a arquitetura do sistema *SIGEAN* foram criadas as seguintes visões arquiteturais utilizando a ferramenta *Archi*®:

- **Visão de *Stakeholders*** que descreve os diferentes interessados na construção do sistema e que foi apresentada na **Figura 1.7**;
- **Visão de Objetivos dos *Stakeholders*** que define quais as metas que cada parte interessada espera cumprir ao utilizar o sistema *SIGEAN*, introduzida na **Figura 1.10**;
- **Visão Lógica** que modela as entidades de dados envolvidas na operação do sistema *SIGEAN*como o modelo apresentado na **Figura 1.11**;
- **Visão de Sistemas**, serviços e repositórios que constituem o sistema *SIGEAN*, apresentada na **Figura 1.12**;
- **Visão de Capacidades** que descreve quais partes do sistema serão responsáveis por cumprir cada requisito, que é ilustrada nas **Figuras 1.13** e **1.14**;

Essas visões ainda não apresentam decisões arquiteturais, mas fazem parte do conhecimento que o arquiteto utilizou para propor uma solução arquitetural coerente para o sistema *SIGEAN*. Para representar as decisões arquiteturais, o arquiteto decidiu utilizar as visões em camadas, de processos e física, descritas a seguir.

1.6.2.1. Visão em Camadas

A **Figura 1.17** descreve de forma geral a organização dos sistemas, serviços e repositórios que compõem a primeira versão arquitetura do sistema *SIGEAN*. Diferentemente da visão de sistemas (ver **Figura 1.12**), a visão em camadas explica como os elementos da arquitetura estão organizados seguindo o padrão camadas (*layers*).

Camadas é um padrão arquitetural que orienta a categorização dos elementos da arquitetura com base nas funcionalidades que eles oferecem para outras partes do sistema e aos objetivos dos usuários finais que eles ajudam a cumprir. Esse padrão impõe restrições às maneiras com que os elementos de camadas adjacentes se comunicam. Alguns dos benefícios de adotar camadas na arquitetura do sistema *SIGEAN* é facilitar a modificação ou substituição dos elementos dentro de uma camada sem afetar outros elementos em camadas adjacentes. Isso é possível devido ao baixo acoplamento que existe entre os elementos de diversas camadas [Harrison and Avgeriou 2007, Richards 2015, Gorton 2011]. Adicionalmente, considerando a arquitetura como camadas de elementos é possível adicionar camadas de elementos destinados à segurança do sistema, como por exemplo a camada de *Qualidade de Serviço (QoS)* no diagrama da **Figura 1.17**.

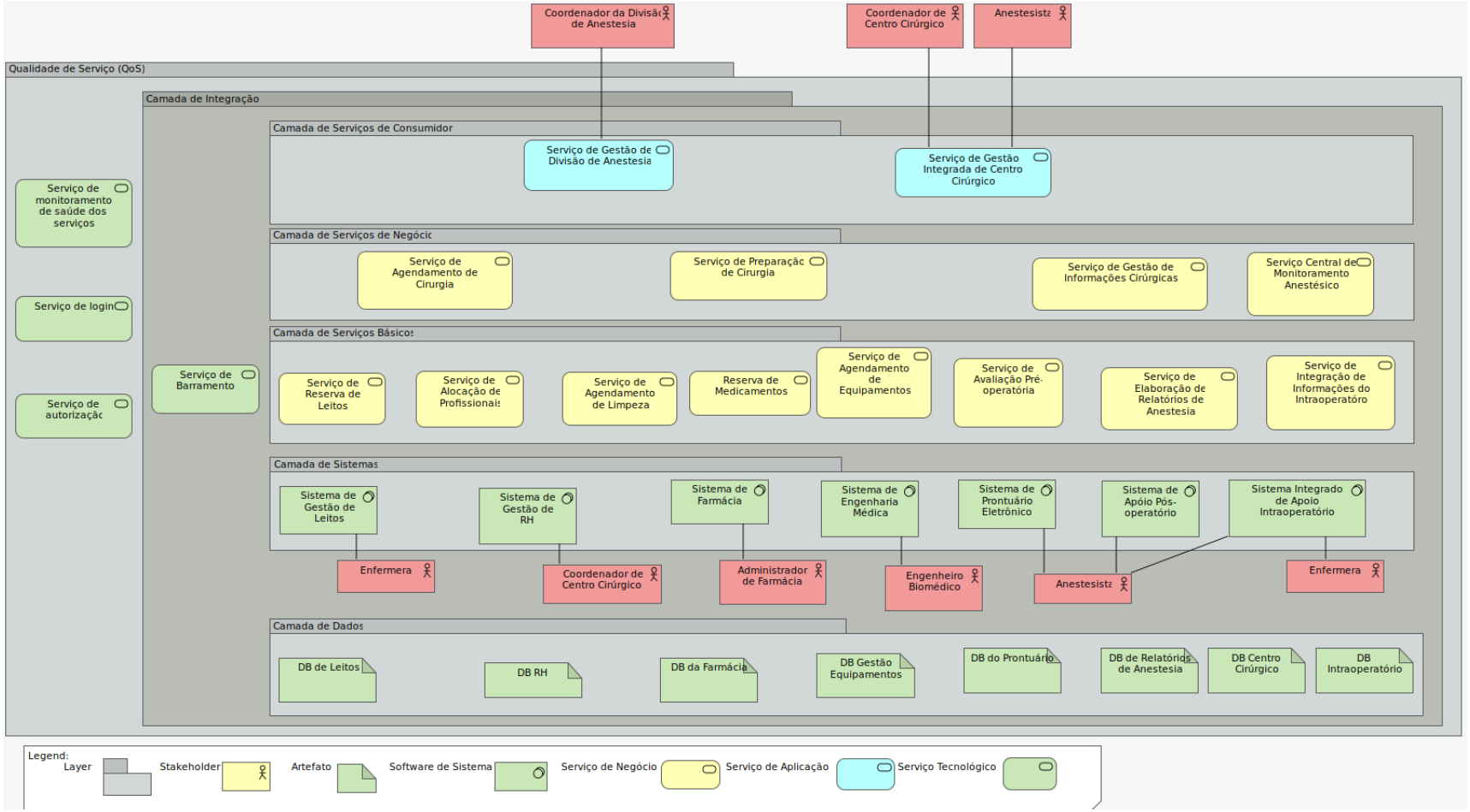

Figura 1.17. Visão em Camadas da Arquitetura do Sistema *SIGEAN* como Instância da Arquitetura de Referência S3.

A arquitetura de referência S3 [Arsanjani et al. 2007] oferece diretrizes para categorizar arquiteturas que seguem o padrão SOA, como é o caso da arquitetura do sistema *SIGEAN*. Ao instanciar S3, foi possível reusar o conhecimento sobre como organizar uma arquiteturas SOA em camadas. Nessa perspectiva, as sete camadas da visão na **Figura 1.17** não foram propostas pelo arquiteto do sistema *SIGEAN* mas reutilizadas a partir da S3 [Arsanjani et al. 2014]. Uma das muitas vantagens da S3 é que a forma de organizar as camadas de software e as relações entre elas ajuda a preencher o *gap* entre os processos de negócio e a infraestrutura de TI em uma organização. Isso é realizado a partir da criação de diferentes tipos de serviços como por exemplo os serviços básicos, de negócio e de consumo, além de serviços dedicados a integrar as comunicações e monitorar a qualidade das operações executadas por todos os serviços na arquitetura.

Outra vantagem de utilizar S3 é que os serviços que participam de um sistema de software podem ser reusados em outros sistemas que pertencem à mesma ou entre diferentes indústrias, agilizando assim o desenvolvimento, implantação (*deployment*) e gerenciamento de sistemas baseados em SOA [Arsanjani et al. 2007, Arsanjani et al. 2014].

As camadas inferiores da **Figura 1.17**, camada de dados e de sistemas, são destinadas a sistemas de software existentes na organização e cujas operações/capacidades serão fornecidas como serviços para o sistema *SIGEAN*. Destaca-se que, esses sistemas operam nos diferentes institutos do HCFMUSP, porém eles são terceirizados, ou seja, não estão sob o controle do HCFMUSP pois existem diferentes empresas de software por trás da manutenção, atualização e gerenciamento desses sistemas. O fato de considerar esses sistemas como partes do novo sistema *SIGEAN* influencia diretamente no custo total para implementar *SIGEAN* como uma nova solução SOA. Nessa visão, também são representados diferentes usuários finais que atualmente podem acessar esses sistemas existentes.

A camada de serviços básicos é composta por todos os serviços que são fornecidos pelos sistemas na camada inferior. Na arquitetura de referência S3, um serviço é considerado como uma especificação abstrata de uma coleção de (um ou mais) funções de negócio expostas por um sistema fornecedor. O sistema deve especificar o serviço através de uma definição abstrata das funcionalidades oferecidas, o que comumente é feito através de APIs REST, ou usando WSDL. Os serviços nessa camada podem ser descobertos e invocados ou possivelmente coreografados/orquestrados para compor os serviços das camadas superiores.

Na camada de serviços de négocio são alocados serviços compostos pelos serviços da camada inferior. Serviços de negócio descobrem, invocam, coreografam ou orquestram serviços básicos. **Coreografia e orquestração** são duas **táticas arquiteturais para interoperabilidade** [Valle et al. 2019, Bass et al. 2012] que possibilitam a coordenação de serviços para executar processos de negócio que precisam da interação de várias funcionalidades de forma organizada. Por exemplo, na **Figura 1.18** pode-se observar os serviços básicos envolvidos e as respectivas informações requeridas pelo serviço composto, o *Serviço de Preparação de Cirurgia*, para atender uma solicitação de preparação de sala.

A camada de consumidor, ou camada de apresentação, fornece as capacidades requeridas para entregar funcionalidades e dados aos usuários finais para eles cumprirem

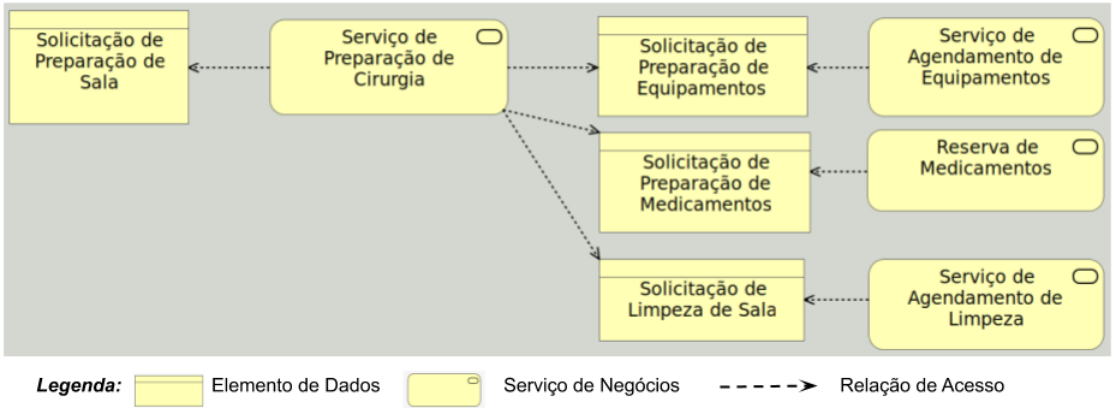

Figura 1.18. *Serviço de Preparação de Cirurgia* coordenando serviços básicos.

suas atividades dentro da organização. Essa camada possibilita a criação rápida do *font-end* dos serviços da camada de negócio. Em *SIGEAN*, os serviços da camada de consumidor oferecerão as interfaces de usuário (UI) para os coordenadores da divisão de anestesia e dos centros cirúrgicos do HCFMUSP.

A camada de integração fornece as capacidades para mediar, encaminhar e transportar de forma confiável as requisições de serviços aos serviços fornecedores. A integração é realizada fazendo uso de um barramento de serviços definido em S3 como ESB (Enterprise Service Bus). Tal barramento fornece capacidades de integração ponto-a-ponto, mediação de protocolos de transferência de dados e mecanismos de transformação de protocolos para garantir interoperabilidade técnica. A camada de integração oferece capacidades de comunicação, invocação e qualidade de serviço (QoS) entre as camadas adjacentes em arquiteturas que seguem o padrão SOA. As maiores vantagens de ter um serviço de barramento ou camada de integração é a comunicação ou relacionamento indireto entre o consumidor da funcionalidade e seu fornecedor, como ilustrado na **Figura 1.19**. O serviço consumidor interage com o serviço fornecedor somente através da camada de integração. Como resultado, cada especificação de serviço somente é exposto através da camada de integração (como um ESB), nunca de forma direta. Adicionalmente, essa camada intermediária desacopla ou dissocia consumidores e fornecedores, facilitando a interoperação e integração de sistemas heterogêneos e distribuídos para formar novos sistemas.

A camada de qualidade de serviço (QoS) fornece às arquiteturas baseadas em SOA capacidades para realizar requisitos de atributos de qualidade. Serviços dessa camada capturam, monitoram, registram e alertam o incumprimento desses requisitos por parte dos serviços das outras camadas. A camada de QoS serve como um "observador" das outras camadas e podem emitir sinais ou eventos quando um requisito de atributo de qualidade não é cumprido, ou quando uma condição de possível descumprimento é detectada. Na arquitetura do sistema *SIGEAN*, a camada de QoS fornece os serviços para garantir que o sistema atingirá requisitos de segurança e disponibilidade. Para isso, os serviços dessa camada deverão implementar **táticas de autorização, autenticação (login) e monitoramento** da saúde ou disponibilidade dos serviços pertencentes a outras camadas.

Os leitores que desejem aprofundar o conhecimento sobre como arquitetar siste-

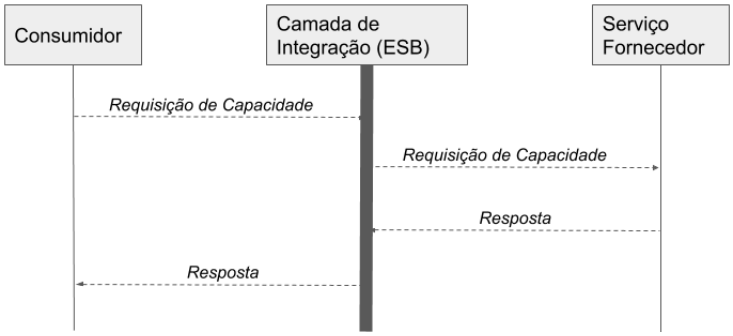

Figura 1.19. Interação indireta entre consumidor e fornecedor de um serviço utilizando a camada de integração.

mas de software baseados em SOA podem consultar o livro de [Josuttis 2007].

1.6.3. Visão de Processo de Negócio

Quando se arquiteta um sistema baseado no padrão SOA, é necessário definir os processos que os serviços de negócio precisam executar. Nessa perspectiva, para a arquitetura do sistema *SIGEAN*, foram criadas três visões que explicam como os processos pré-operatório, intraoperatório e pós-operatório devem ser executados. A **Figura 1.20** apresenta os serviços das camadas de serviços básicos e de negócio precisam interagir para executar as atividades envolvidas no processo pré-operatório, que irão ser acessadas pelos anestesistas e coordenadores dos centros cirúrgicos do HCFMUSP através da UI fornecida pelo *Serviço de Gestão Integrada de Centro Cirúrgico*.

1.6.3.1. Visão Física

Considerando a importância de ter conhecimento sobre os recursos físicos que a arquitetura irá impor durante a implantação do sistema nas instalações da organização, foi importante definir duas visões físicas para o sistema *SIGEAN*, que são apresentadas nas **Figuras 1.21 e 1.22**.

A primeira visão (**Figura 1.21**) apresenta o mapeamento dos diferentes serviços requeridos para executar os processos dentro dos centros cirúrgicos. Segundo esta visão, pode-se observar que o arquiteto definiu a necessidade de ter um servidor central em cada um dos 9 centros cirúrgicos do HCFMUSP. Esse servidor central é responsável por alocar a maioria de serviços das camadas de serviços básicos e de negócio. Dessa forma, é nesse servidor que a orquestração dos serviços para executar todos os processos durante a gestão do centro cirúrgico é realizada. O servidor central também aloca o serviço de *Gestão Integrada de Centro Cirúrgico* com o qual os usuários finais (coordenador de centro cirúrgico e anestesistas) irão interagir para acessar às funcionalidades do sistema, como visto na **Figura 1.20**. O servidor central do centro cirúrgico é considerado um servidor de aplicações e lógica de negócio e não é responsável pela persistência dos dados gerados e consumidos pelos serviços. A persistência das informações está baixo responsabilidade do *Servidor Central de Dados do Centro Cirúrgico*. O desacoplamento dos dados e aplicações é uma implementação da **tática "evitar um único ponto de falha"** para favorecer

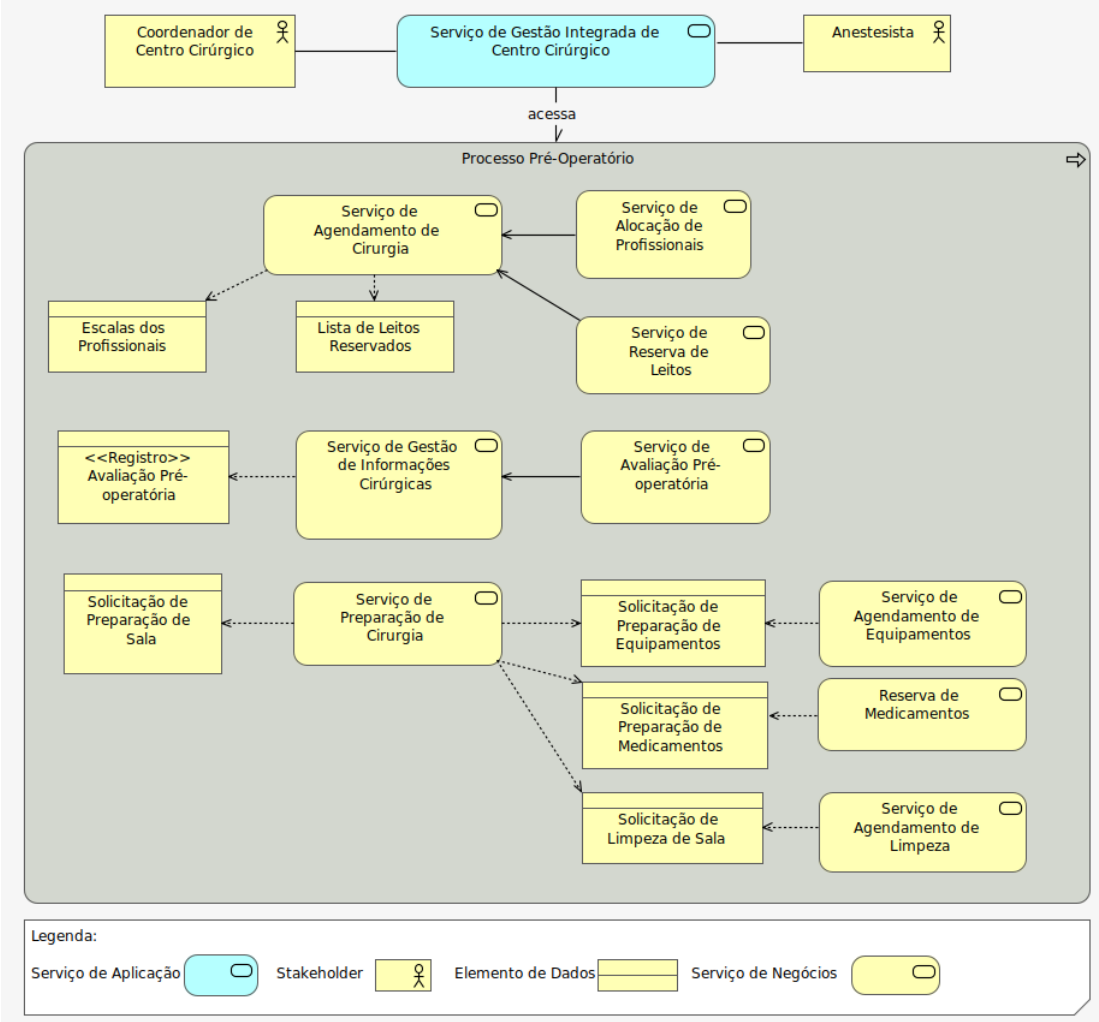

Figura 1.20. Visão de *Processo Pré-Operatório*

a **disponibilidade** dos serviços e dados para evitar falhas generalizadas no sistema no caso de apresentar-se falhas em algum desses servidores.

Devido à alta carga de processamento e à segurança de informações geradas pelos diferentes equipamentos dentro das salas de cirurgia, foi necessário definir um servidor local para cada sala responsável por executar o *serviço de integração de informações do intraoperatório*. Nessa perspectiva, cada sala de cirurgia terá seu próprio servidor, que além de integrar as informações do paciente no relatório de anestesia, deve comunicar a cada 5 minutos o estado atual do paciente tanto ao anestesista dentro da sala quanto à coordenação do centro cirúrgico. Essa comunicação entre o *serviço de integração de informações do intraoperatório* e o *Serviço central de monitoramento anestésico* é realizado através do *Serviço de barramento* alocado num outro nó físico, o servidor de barramento.

Outra decisão importante que pode-se identificar ao analisar a visão física da **Figura 1.21** é a aplicação da tática de redundância de recursos físicos que permite aumentar os níveis de disponibilidade, capacidade (ou escalabilidade) e desempenho do sistema. Por exemplo, foi planejado a redundância do servidor de QoS, central do hospital e de

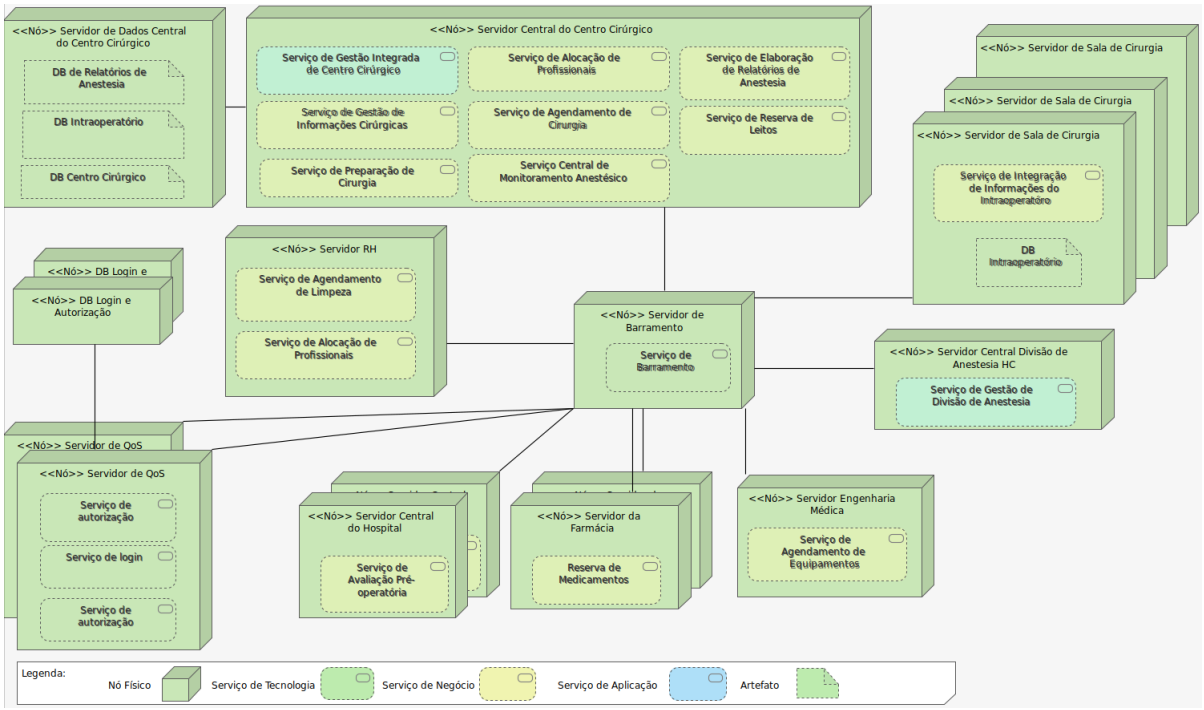

Figura 1.21. Visão Física da Arquitetura do Sistema *SIGEAN* a ser Implantada no Complexo HCFMUSP, parte 1.

farmácia os quais deverão atender grandes quantidades de requisições de diversos usuários de forma concorrente. Ao ter réplicas dos servidores é possível atender o incremento de demanda de requisições dos servidores pois pode ser executado um balanceamento da demanda entre as réplicas, evitando sobrecarregar um único servidor, favorecendo a disponibilidade desse elemento físico e os serviços alocados nele. Igualmente, esta tática evita que os tempos de resposta aumentem devido ao aumento de carga de trabalho nos servidores, pois tal carga será balanceada entre os diferentes nós replicados.

A segunda visão (ver **Figura 1.22**) são representados os recursos físicos necessários para implementar o sistema *SIGEAN* no HCFMUSP considerando os nove centros cirúrgicos dentro dos 7 institutos que compões este complexo hospitalar. Nessa visão, é possível observar a necessidade de ter diversos níveis de integração no sistema *SIGEAN*. Na visão da **Figura 1.21**, foi representado um servidor de barramento para integrar os diferentes nós que contém serviços de diversos setores dentro do HCFMUSP, como RH, ambulatório (servidor central do hospital), farmácia, engenharia médica, divisão de anestesia, centros cirúrgicos e suas salas de cirurgia. Ao fazer uma análise mais rigorosa para integrar os serviços que gestionam as informações relacionadas aos procedimentos anestésicos, foi identificada a necessidade de instanciar o serviço de barramento em diferentes servidores para evitar a possível sobrecarga do trabalho de transferência, comunicação e mediação executado por um só barramento centralizado para todo o sistema *SIGEAN*.

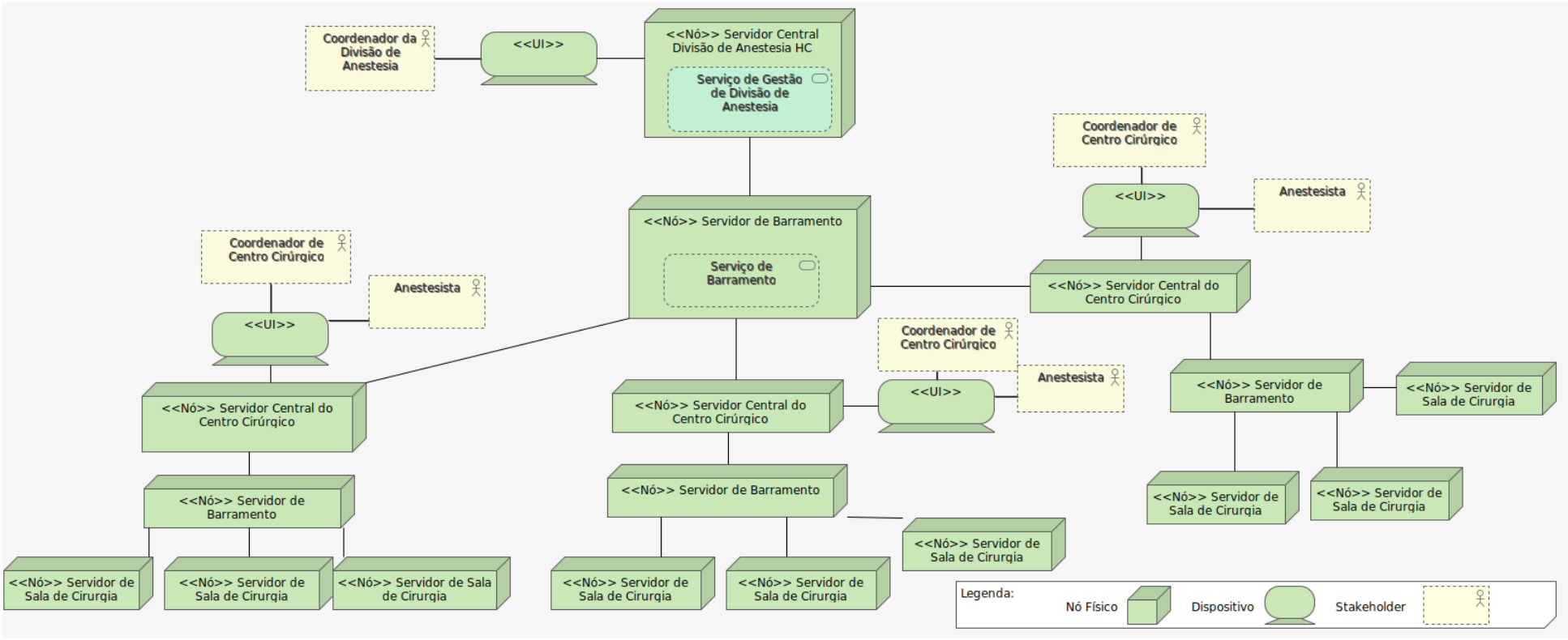

Figura 1.22. Visão Física da Arquitetura do Sistema *SIGEAN* a ser Implantada no Complexo HCFMUSP, parte 2.

Nesse contexto, é necessário ter um servidor de barramento por centro cirúrgico que integre todas as informações geradas pelas diversas instâncias do *Serviço de Integração de Informações do Intraoperatóro* que são executadas nos servidores das salas cirúrgicas em cada centro cirúrgico. Ao utilizar hierarquias de barramentos o arquiteto visou aumentar a escalabilidade do sistema *SIGEAN* devido que novas salas de cirurgia podem ser abertas nos centros cirúrgicos sem afetar a capacidade do barramento central, e ao mesmo tempo é possível conectar gradativamente os centros cirúrgicos sem a necessidade de realizar grandes modificações nos serviços de barramento do complexo hospitalar.

1.6.4. Documentação da Arquitetura

A última atividade da etapa de projeto arquitetural é a documentação da arquitetura do sistema. Documentar uma arquitetura é uma questão de representar e descrever as visões e, em seguida, adicionar informações se aplicam a mais de uma visão [Clements et al. 2010]. É de grande importância traçar os relacionamentos entre visões para garantir um correto entendimento da arquitetura desde diferente visões.

Destaca-se aqui que, as atividades de tomada de decisões, descrição (criação das visões) e a documentação da arquitetura são atividades que devem ser executadas paralelamente e incrementalmente. Ou seja, uma vez uma decisão seja tomada pelo arquiteto para abordar certo RAS, essa decisão deve ser representada nas visões e ao mesmo tempo explicada no documento da arquitetura. Novas decisões irão ser inseridas nas visões e no documento de forma incremental.

Uma das vantagens de utilizar ferramentas de modelagem arquitetural, como *Archi*®, para a criação das visões arquiteturais é a possibilidade de gerar automaticamente a documentação da arquitetura, poupando esforços ao arquiteto para documentar de forma clara e correta os modelos.

Como exemplo de um documento arquitetural gerado por *Archi*®, a documentação da arquitetura do sistema *SIGEAN* pode ser consultada no seguinte endereço: `https://linamgr.github.io/SIGEAN/`.

1.7. Considerações Finais

Os autores deste capítulo destacam que a arquitetura do sistema *SIGEAN* aqui apresentada está ainda em sua primeira versão. Dessa forma, as decisões arquiteturais apresentadas aqui ainda precisam ser avaliadas com os respectivos *stakeholders*.

Adicionalmente, algumas decisões arquiteturais ainda precisam ser representadas nas visões, como é o caso da tática de encriptação dos dados nos diferentes repositórios. Outra decisão arquitetural a ser definida é em relação a quais padrões para interoperabilidade semântica e sintática as organizações responsáveis pelos sistemas já existentes que operam no HCFMUSP estão dispostas a implementar para facilitar a troca das informações entre eles e os serviços de barramento. Uma vez as partes interessadas concordem em quais padrões de interoperabilidade utilizar, o arquiteto do sistema *SIGEAN* deverá identificar quais estruturas de dados precisam ser formatadas seguindo os padrões selecionados. Além disso, o arquiteto precisará representar e documentar nos modelos da visão

lógica (por exemplo a **Figura 1.11**) quais termos deverão adotar a semântica do padrão de terminologia médica selecionados. De forma similar, o arquiteto deverá identificar e descrever como as informações trocadas devem ser formatadas sintaticamente para garantir uma correta comunicação entre as partes da arquitetura. Essa especificação deve ser representada na visões lógica e de processos como aquelas apresentadas nas **Figuras 1.11** e **1.20**.

Atualmente, o planejamento da etapa de avaliação está sendo executada. Visa-se que a primeira versão da arquitetura do sistema *SIGEAN* seja avaliada com os diferentes interessados no sistema. Com tal avaliação pretende-se a verificação das decisões arquiteturais em relação aos RAS definidos para a arquitetura. Essa atividade de verificação faz-se necessária uma vez que é preciso ter um nível de certeza sobre se: (i) a arquitetura de software é adequada para o sistema desejado; (ii) os RAS serão alcançados; (iii) o software poderá ser desenvolvido conforme a arquitetura e com os recursos disponíveis; e (iv) seguindo a arquitetura proposta é possível que o sistema apresente problemas ou baixa qualidade durante sua operação.

Por fim, deve-se definir quais tipos de tecnologias serão utilizadas para permitir o desenvolvimento do sistema *SIGEAN* conforme o que foi especificado na arquitetura de software. Para isso, é necessário estimar os riscos associados à adoção de tecnologias e infraestruturas fornecidas por empresas de cloud computing como IBM, Amazon, Google entre outras. Também será necessário entender as possíveis mudanças organizacionais que uma abordagem de gestão de tecnologia local irá trazer para o HCFMUSP. Ou seja, é necessário que o arquiteto estude as implicações de adotar tecnologias de código aberto para a implantação de uma nuvem privada para o HCFMUSP. Nessa perspectiva, a grande pergunta que deve ser analisada durante o processo de avaliação da arquitetura é como manter a integridade da arquitetura sem extrapolar o orçamento e cronograma estimado para o projeto de desenvolvimento do sistema *SIGEAN*.

Acrônimos

ADL Architectural Description Language, ou em Português: Linguagem de Descrição Arquitetural.

API Application Programming Interface, ou em Português: Interface de programação de aplicações.

AQ Atributo de Qualidade

ASR Architecturally Significant Requirements.

bpm Batimentos por minuto

ICD International Classification of Disease, ou em Português: Classificação Internacional de Doenças

ESB Enterprise Service Bus, em Português: Barramento de Serviços Empresariais.

FHIR (Fast Health Interoperable Resources, ou em Português: Recursos Interoperáveis Rápidos em Saúde

HCFMUSP Hospital das Clínicas, Faculdade de Medicina da Universidade de São Paulo

HL7 High Level 7

ICHC Instituo Central do Hospital das Clínicas

ICR Instituto da Criança

ICESP Instituto do Câncer do Estado de São Paulo

INCOR Instituto do Coração

INRAD Instituto de Radiologia

IntOR Sistema Integrado de Sala de Cirurgia

IOT Instituto de Ortopedia e Traumatologia

IPQ Instituto de Psiquiatria

IT Information Technology, em Português Tecnologia da Informação.

LOINC Logical Observation Identifiers Names and Codes, ou em Português: Nomes e códigos de identificadores de observação lógica de laboratórios médicos.

QoS Quality of Service, em Português Qualidade de Serviço.

RAS Requisitos Arquiteturalmente Significativos.

REST Representational State Transfer, em Português Transferência Representacional de Estado

RF Requisito Funcional

RNF Requisito Não Funcional

RPA Recuperação pós-anestésica

S-RES Sistemas de Registro Eletrônico em Saúde

SGBD Sistema de Gerenciamento de Base de Dados

SIGEAN Sistema Integrado de Gestão de Divisão de Anestesia

SNOMED-CT Systematized Nomenclature of Medicine - Clinical Terms, ou em Português: Nomenclatura sistematizada de medicina - Termos Clínicos

SOA Service Oriented Architecture, ou em Português: Arquitetura Orientada a Serviços.

SO2 Saturação de Oxigênio

SBIS Sociedade Brasileira de Informática em Saúde

UI User Interface, em Português: Interface de Usuário

UML Unified Modeling Framework. Em Português: Linguagem de Modelagem Unificada.

UTI Unidade de Terapia Intensiva

WSDL Web Services Definition Language, em Português: Linguagem de Definição de Serviços Web.

A. Questionário de Entrevista

A seguir, apresenta-se o questionário utilizado para realizar a entrevista para um profissional de saúde com perfil de *Coordenador do Centro Cirúrgico*. O questionário foi dividido nas seguintes quatro partes:

A.1. Parte I: Coleta de dados demográficos

- *Nome completo:*
- *Cargo:*
- *Anos de experiência:*
- *Instituto com o qual tem vínculo:*

A.2. Parte II: Como é hoje?

- Qual o papel que você desempenha no centro cirúrgico?
- Você realiza alguns procedimentos antes de uma cirurgia?; Quais?
- Você realiza alguns procedimentos durante uma cirurgia?; Quais?
- Você realiza alguns procedimentos depois de uma cirurgia?; Quais?
- Quais informações devem ser armazenadas?
- Como elas são armazenadas?
- Como são recolhidas essas informações?
- Os procedimentos mudam de acordo com o tipo de cirurgia?

A.3. Parte III: Quais são os problemas?

- Quais problemas ou dificuldades você considera que existem em relação ao armazenamento de informações no centro cirúrgico?
- Quais problemas ou dificuldades você considera que existem em relação à execução das suas atividades antes de uma cirurgia?
- Quais problemas ou dificuldades você considera que existem em relação à execução das suas atividades depois de uma cirurgia?
- Quais problemas ou dificuldades você considera que existem em relação à comunicação entre todos os profissionais envolvidos durante uma cirurgia?
- Quais problemas ou dificuldades você considera que existem em relação aos equipamentos utilizados?
- Qual seu maior medo ou preocupação na hora de usar sistemas de software como apoio na execução das atividades do centro cirúrgico?

A.4. Parte IV: Como pode melhorar?

- Como você imaginaria um sistema ideal para a realização de suas atividades no centro cirúrgico?
- Como você imagina que seria a melhor forma para apresentar (ou você obter) as informações relevantes para suas atividades?
- O que você acha que poderia ser melhorado em relação ao armazenamento de informações do centro cirúrgico?
- Qual a sua opinião em relação a usar sistemas automatizados para apoiar suas atividades no centro cirúrgico?

- Você acredita que um sistema integrado dos equipamentos da sala de cirurgia melhoraria seu dia a dia?; De que maneira?

B. Termo de Confidencialidade

Título do Projeto:

Responsáveis/Coordenadores do Projeto:

Ao assinar o seguinte termo,

1. Eu entendo que todas as informações são confidenciais. Concordo em concluir o questionário para fins de pesquisa e que os dados derivados dessa pesquisa podem ser publicados de forma anônima em periódicos, conferências e publicações em blogs.
2. Entendo que minha participação neste estudo de pesquisa é totalmente voluntária e que recusar participar não envolverá penalidade ou perda de benefícios. Se eu escolher, posso retirar minha participação a qualquer momento. Eu também entendo que, se eu optar por participar, posso me recusar a responder qualquer pergunta que eu não esteja confortável em responder.
3. Entendo que posso entrar em contato com os pesquisadores se tiver alguma dúvida sobre a pesquisa. Estou ciente de que meu consentimento não me beneficiará diretamente. Também estou ciente de que os autores manterão os dados coletados em perpetuidade e poderão utilizar dados para trabalhos acadêmicos futuros.
4. Eu livremente forneço consentimento e reconheço meus direitos como participante voluntário da pesquisa, conforme descrito acima, e forneço consentimento aos pesquisadores para usar as informações fornecidas na condução de pesquisas sobre as áreas mencionadas acima.

Nome completo do entrevistado:

CPF:

Assinatura do entrevistado:

Local e data:

Referências

[Adler 2005] Adler, P. J. (2005). Dealing with interviews when creating personas: a practical approach. In Buur, J. and Mathews, B., editors, *Proceedings of Student Interaction Design Research Conference SIDER05,*, pages 84–88. University of Southern Denmark.

[Alves 2017] Alves, J. M. (2017). The adequate model: software quality evaluation model for telemedicine and telehealth systems. Master's thesis, UFSC.

[Angelov et al. 2013] Angelov, S., Trienekens, J., and Kusters, R. (2013). Software reference architectures - exploring their usage and design in practice. In *European Conference on Software Architecture*.

[Arsanjani et al. 2007] Arsanjani, A., Zhang, L.-J., Ellis, M., Allam, A., and Channabasavaiah, K. (2007). S3: A service-oriented reference architecture. *IT Professional*, 9(3):10–17.

[Arsanjani et al. 2014] Arsanjani, A., Zhang, L.-J., Ellis, M., Allam, A., and Channabasavaiah, K. (2014). Design an soa solution using a reference architecture. improve your development process using the soa solution stack. techreport, IBM.

[Avgeriou and Zdun 2005] Avgeriou, P. and Zdun, U. (2005). Architectural patterns revisited - a pattern language. In *Pattern Languages of Programming Conference*, pages 1–40.

[Bandara 2015] Bandara, N. (2015). Connected health reference architecture. techreport, WSO2.

[Bass et al. 2012] Bass, L., Clements, P., and Kazman, R. (2012). *Software Architecture in Practice*. Addn-Wesley, 3ª edition.

[Buschmann et al. 1997] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., and Stal, M. (1997). *Pattern-Oriented Software Architecture — A System of Patterns — Volume 1*. John Wiley & Sons.

[Caldeira 2017] Caldeira, F. N. (2017). Avaliação situada de usabilidade de bombas infusoras em uma unidade de terapia intensiva. Master's thesis, Universidade Federal do Estado do Rio de Janeiro.

[Cavalini and Cook 2014] Cavalini, L. T. and Cook, T. W. (2014). Use of xml schema definition for the development of semantically interoperable healthcare applications. In Gibbons, J. and MacCaull, W., editors, *Foundations of Health Information Engineering and Systems*, pages 125–145, Berlin, Heidelberg. Springer Berlin Heidelberg.

[CBS News 2017] CBS News (2017). Global cyberattack strikes dozens of countries, cripples u.k. hospitals. on-line: https://www.cbsnews.com/news/hospitals-across-britain-hit-by-ransomware-cyberattack/.

[Cervantes and Kazman 2016] Cervantes, H. and Kazman, R. (2016). *Designing Software Architectures: A Practical Approach*. Addison-Wesley Professional, 1ª edition.

[CIO 2019] CIO (2019). Saúde enfrenta crescente problema de cibersegurança, diz especialista. on-line: https://cio.com.br/saude-enfrenta-crescente-problema-de-ciberseguranca-diz-especialista/. On-line.

[Clements et al. 2010] Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., and Stafford, J. (2010). *Documenting Software Architectures: Views and Beyond*. Addison-Wesley Professional, 2ª edition.

[Fairbanks 2010] Fairbanks, G. (2010). *Just Enough Software Architecture: A Risk-Driven Approach*. Marshall & Brainerd, 1ª edition.

[Garcés 2018] Garcés, L. (2018). *A reference architecture for healthcare supportive home systems from a systems-of-systems perspective.* PhD thesis, Universidade de São Paulo.

[Garcés et al. 2015] Garcés, L., Ampatzoglou, A., Avgeriou, P., and Nakagawa, E. Y. (2015). A comparative analysis of reference architectures for healthcare in the ambient assisted living domain. In *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*, pages 270–275.

[Garcés et al. 2017] Garcés, L., Ampatzoglou, A., Avgeriou, P., and Nakagawa, E. Y. (2017). Quality attributes and quality models for ambient assisted living software systems: A systematic mapping. *Information and Software Technology*, 82:121 – 138.

[Garcés et al. 2020] Garcés, L., Oquendo, F., and Nakagawa, E. Y. (2020). Assessment of reference architectures and reference models for ambient assisted living systems: Results of a systematic literature review. *International Journal of E-Health and Medical Communications (IJEHMC)*, 11(1):20.

[Garcés et al. 2016] Garcés, L., Oquendo, F., and Nakgawa, E. Y. (2016). A quality model for aal software systems. In *2016 IEEE 29th International Symposium on Computer-Based Medical Systems (CBMS)*, pages 175–180.

[Gorton 2011] Gorton, I. (2011). *Essential Software Architecture*. Springer Publishing Company, Incorporated, 2nd edition.

[Harrison and Avgeriou 2007] Harrison, N. B. and Avgeriou, P. (2007). Leveraging architecture patterns to satisfy quality attributes. In *Proceedings of the First European Conference on Software Architecture*, ECSA'07, page 263–270, Berlin, Heidelberg. Springer-Verlag.

[Hofmeister et al. 2007] Hofmeister, C., Kruchten, P., Nord, R. L., Obbink, H., Ran, A., and America, P. (2007). A general model of software architecture design derived from five industrial approaches. *Journal of Systems and Software*, 80(1):106 – 126.

[ISO/IEC 2011] ISO/IEC (2011). ISO/IEC 25010:2011(en) systems and software engineering — systems and software quality requirements and evaluation (square) — system and software quality models. On-line.

[ISO/IEC 2013] ISO/IEC (2013). ISO/IEC 27002:2013(en) Information technology — Security techniques — Code of practice for information security controls. On-line.

[ISO/IEC 2015] ISO/IEC (2015). ISO/IEC 15408-1:2009. Information technology — Security techniques — Evaluation criteria for IT security — Part 1: Introduction and general model. On-line.

[ISO/IEC/IEEE 2011] ISO/IEC/IEEE (2011). Iso/iec/ieee 42010:2011 systems and software engineering — architecture description.

[Josuttis 2007] Josuttis, N. (2007). *Soa in Practice: The Art of Distributed System Design*. O'Reilly Media, Inc.

[Kruchten 1995] Kruchten, P. (1995). The 4+1 view model of architecture. *IEEE Software*, 12(6):42–50.

[Losavio et al. 2015] Losavio, F., Ordaz, O., and Esteller, V. (2015). Quality-based bottom-up design of reference architecture applied to healthcare integrated information systems. In *2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS)*, pages 70–75.

[Martínez-Fernández et al. 2013] Martínez-Fernández, S., Ayala, C. P., Franch, X., and Marques, H. M. (2013). Benefits and drawbacks of reference architectures. In *European Conference on Software Architecture*, pages 307–310.

[Nakagawa et al. 2014] Nakagawa, E. Y., Oquendo, F., and Maldonado, J. C. (2014). *Reference Architectures*, chapter 2. Wiley Online Library.

[Neto and Junior 2019] Neto, J. A. d. A. and Junior, A. E. a. (2019). Atualmente, os diferentes elos na área de saúde não estão integrados. como mudar esse cenário? *GVEXECUTIVO*, 18(1):29–31.

[Portal Saúde Business 2020] Portal Saúde Business (2020). Mercado de healthtech cresce 141% em cinco anos, de acordo com o distrito healthtech report. on-line: https://tinyurl.com/y9gbhlac.

[Pressman 2011] Pressman, R. S. (2011). *Engenharia de Software. Uma Abordagem Profissional.* The McGraw-Hill Companies, Inc., 7ª edition.

[Ratwani et al. 2015] Ratwani, R. M., Fairbanks, R. J., Hettinger, A. Z., and Benda, N. C. (2015). Electronic health record usability: analysis of the user-centered design processes of eleven electronic health record vendors. *Journal of the American Medical Informatics Association*, 22(6):1179–1182.

[Richards 2015] Richards, M. (2015). *Software Architecture Patterns*. O'Reilly.

[Richards and Ford 2020] Richards, M. and Ford, N. (2020). *Fundamentals of Software Architecture*. O'Reilly Media, Inc.

[Rozanski and Woods 2011] Rozanski, N. and Woods, E. (2011). *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Pearson Education, 2ª edition.

[Silva and Junior 2019] Silva, M. L. and Junior, L. A. V. (2019). *Manual de Certificação para Sistemas de Registro Eletrônico em Saúde. Instituído e regido pela Resolução CFM nº 1821/2007 (Versão 4.3)*. Sociedade Brasileira de Informática em Saúde.

[Sommerville 2011] Sommerville, I. (2011). *Engenharia de Software*. Pearson Education do Brasil, 9ª edition.

[The openEHR Foundation 2018] The openEHR Foundation (2018). *openEHR Architecture. Architecture Overview.* The openEHR Foundation. On-line: `https://specifications.openehr.org/`.

[Turner 2015] Turner, G.-M. (2015). Oregon's failed obamacare exchange is a warning for other states. On-line.

[Valle et al. 2019] Valle, P. H. D., Garcés, L., and Nakagawa, E. Y. (2019). A typology of architectural strategies for interoperability. In *Proceedings of the XIII Brazilian Symposium on Software Components, Architectures, and Reuse (SBCARS)*, page 3–12, Salvador, Brazil.

[Wanderley 2020] Wanderley, M. (2020). Setor de health tech cresce no brasil. on-line: https://computerworld.com.br/2020/02/19/setor-de-health-tech-cresce-no-brasil/.

[Wartena et al. 2010] Wartena, F., Muskens, J., Schmitt, L., and Petković, M. (2010). Continua: The reference architecture of a personal telehealth ecosystem. In *The 12th IEEE International Conference on e-Health Networking, Applications and Services*, pages 1–6.

[Wiegers and Beatty 2013] Wiegers, K. E. and Beatty, J. (2013). *Software Requirements 3*. Microsoft Press, USA.

[Wohlin et al. 2012] Wohlin, C., Runeson, P., Hst, M., Ohlsson, M. C., Regnell, B., and Wessln, A. (2012). *Experimentation in Software Engineering*. Springer Publishing Company, Incorporated.