NADER GHODDOSI

UM MÉTODO DE APOIO À DECISÃO NA ESTIMAÇÃO DE SOFTWARE BASEADO EM SERVIÇOS EM UMA PERSPECTIVA SOA

Tese submetida ao Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina para a obtenção do Grau de Doutor em Engenharia de Automação e Sistemas.

Orientador: Prof. Dr. Ricardo José Rabelo.

Florianópolis
2017

Ghoddosi, Nader
Um Método de Apoio à Decisão na Estimação de Software baseado em Serviços em uma Perspectiva SOA / Nader Ghoddosi ; orientador, Ricardo José Rabelo, 2017.
257 p.

Tese (doutorado) - Universidade Federal de Santa Catarina, 2017, Centro Tecnológico, programa de Pós Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2017.

Inclui referências.

1. Engenharia de Automação e Sistemas. 2.SOA. 3. Serviços de software. I. Rabelo, Ricardo José. II. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Engenharia de Automação e Sistemas. III. Título.

Nader Ghoddosi

Esta Tese foi julgada adequada para a obtenção do Título de Doutor em Engenharia de Automação e Sistemas e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas.

Florianópolis, 07 de dezembro de 2017.

_____

Prof. Daniel Ferreira Coutinho, Dr.
Coordenador do programa

_____

Prof. Ricardo José Rabelo, Dr.
Orientador

Banca Examinadora:

_____	_____
Prof. Leandro Buss Becker, Dr. DAS- UFSC	Prof. Leonardo Guerreiro Azevedo, Dr. UNIRIO - RJ
_____	_____
Prof. Jean Carlo Rossa Hauck, Dr. INE - UFSC	Prof. Rômulo Silva de Oliveira, Dr. DAS- UFSC

Dedico esta tese de doutorado aos meus pais, por me ensinarem em viver. À minha esposa Sheila, pelo amor, amizade dedicados a mim e pelo apoio durante a jornada universitária.
Dedico também a todos que escolherem o caminho espiritual e deixarem de lado seu ego, conquistando um nível elevado de desprendimento e amor ao próximo.

AGRADECIMENTOS

À Universidade Federal de Santa Catarina e a seus professores e servidores, em especial ao Programa de Pós-graduação em Engenharia de Automação e Sistemas, pelo apoio institucional à realização desta tese.

Aos coordenadores do Programa de Pós-graduação em Engenharia de Automação e Sistemas.

Ao meu orientador, professor Dr. Ricardo José Rabelo por todo apoio e atenção dispensados, sem o qual certamente não conseguiria levar esse trabalho ao patamar que chegou.

Aos amigos do grupo GSIGMA, em especial Maiara Cancian e João Ferreira de Santanna Filho, que me fizeram companhia nessa jornada.

À minha esposa Sheila Ghoddosi, que me apoiou e incentivou em todos os dias dessa jornada.

Finalmente, mas não menos importante, agradeço a Deus, sem Ele nada seria possível.

“Artes, ofícios e ciências elevam o mundo do ser e são conducentes à sua exaltação. O conhecimento é como asas para a vida do homem, é como uma escada pela qual ele possa ascender. Incumbe a cada um adquiri-lo. O conhecimento deve, porém se adquirido de tais ciências que possam prestar benefícios aos povos da terra, e não daquelas que começam por meras palavras e assim também terminam... Na realidade, o conhecimento é um verdadeiro tesouro para o homem, é para ele uma fonte de glória, de graça, de júbilo e exaltação, de alegria e contentamento. Feliz o homem que ele adere e infelizes os desatentos.”

Abdu´l-Bahá

RESUMO

A indústria de software tem se tornado cada vez mais um setor de grande relevância econômica e estratégica no desenvolvimento de um País. Cada vez mais pressionadas pela grande competitividade e pela constante necessidade de investimentos em inovação e recursos humanos, tem sido essencial para a sobrevivência das empresas a adoção de novos paradigmas tecnológicos que dê a elas melhores condições de implementação de mais rentáveis, modernos e sustentáveis modelos de negócio. A Arquitetura Orientada a Serviços (*Service Oriented Architecture - SOA*) tem emergido como um dos mais importantes novas paradigmas de software. Diferentemente do modelo tradicional de software, em SOA um sistema é constituído por uma composição de módulos de software distribuídos, desacoplados e heterogêneos - chamados *serviços* - cujas funcionalidades são invocadas dentro de variadas e flexíveis lógicas de negócio do sistema. Neste sentido, SOA introduz uma nova visão quanto ao projeto, desenvolvimento, reuso e integração de software, podendo este ser fornecido mais flexivelmente sob diversas arquiteturas e modelos de negócios com um maior grau de alinhamento à camada de negócio da empresa. Apesar de suas grandes potencialidades e dos grandes avanços verificados nos ambientes de desenvolvimento, em normas e em modelos de maturidade para ajudar no desenvolvimento de soluções SOA, a sua adoção num sistema e por parte de uma empresa tem grandes impactos, em diferentes dimensões. Como consequência, projetos SOA geralmente acabam por ser mais complexos de serem projetados, geridos, integrados e implantados e, portanto, com um razoável alto risco de insucesso. Além disso, dependendo do tipo de problema a ser resolvido, do que já existe de sistemas legados, dos requisitos do negócio em questão e de uma série de outros aspectos, nem sempre a adoção de SOA / orientação a serviços pode vir a ser a melhor decisão para o caso de negócios em questão. Dentro do ciclo de vida SOA, essas ponderações são feitas na etapa de análise e que via de regra são baseadas em *estimativa de software*, classicamente envolvendo o cálculo de custos, esforço e tempo de desenvolvimento e, em alguns casos, o do grau de alinhamento ao negócio. A literatura mostra uma lacuna sobre métodos mais completos ou que não sejam apenas baseados em *checklists* genéricos para apoiar gestores de TI nessa decisão sobre quando implementar demandas de software na forma de serviços e os seus custos gerais. Observações empíricas e relatos na Internet também comentam que muitos gestores tomam tal decisão

essencialmente com base nas suas experiências e níveis de conhecimento técnico, portanto com grande margem de subjetividade, baixa padronização, pouca mensuração quantitativa dos custos gerais daquela adoção e com limitada visão do grau de alinhamento ao negócio. Na direção de mitigar tais problemas, esta tese propõe um método que sistematiza o processo de análise sobre a adoção ou não de serviços numa dada solução de software. O método é constituído por 15 fatores de análise, inter-relacionados, organizados e sequencialmente percorridos pelo gestor de TI, que os alimenta com certas informações da empresa e da solução pretendida. Cada fator é calculado e gera um valor. Tomando cada funcionalidade candidata em análise, o método sintetiza os valores dos fatores a medida que são calculados, provendo ao final números concretos para custos, esforço, tempo de desenvolvimento e grau de alinhamento ao negócio para cada uma delas individualmente e em conjunto. O método foi desenvolvido com base nas melhores práticas de engenharia de software e de SOA, adaptando-as a um cenário de tomada de decisão multicritério e de comparação em relação a custos gerais de desenvolvimento de um software tradicional. Um conjunto de empresas de software foi utilizado para avaliar o método, onde se pôde confirmar a sua proposição de valor.

Palavras-chave: SOA, método de análise de software, serviço de software.

ABSTRACT

The software industry has increasingly become a very important and strategic sector for countries' development. More and more pressured by high competitiveness and by constant need of investments in innovation and human resources, the adoption of new technological paradigms by companies has been essential for their survival to leverage them to implement more profitable, modern and sustainable business models. Service Oriented Architecture (SOA) has emerged as one of the most important new paradigms of software. Unlike the traditional software model, in SOA a system is constituted by a composition of distributed, decoupled and heterogeneous software modules - called *services* - whose functionalities are invoked under several and flexible business' logics. Thus, SOA introduces a new vision in terms of systems' design, development, reuse and integration. New systems can be provided more flexibly under different architectures and business models and with a higher degree of alignment to the company's business layer. Despite the high potential of SOA and the large advances in supporting development environments, norms and maturity models to assist in the development of SOA solutions, SOA adoption by a company has big impacts, in different dimensions. As a consequence, SOA projects often end up being more complex to be designed, managed, integrated and deployed, and therefore having a reasonable high failure risk. In addition, depending on the type of problem to be solved, on what already exists in terms of legacy systems, on the current business requirements and on several other aspects, the adoption of SOA / service orientation may be not the best decision for the case in place. Regarding the SOA life cycle, this evaluation is carried out in the *analysis* phase mostly applying software estimation techniques that classically consider costs, development time and efforts and, in some cases, the degree of business alignment. Literature shows a *gap* of more comprehensive supporting methods or of ones that are not just generic checklists to support IT managers in that estimation. Empirical observations and reports in the Internet have been also pointed out that many managers use to take that decision essentially based on their experiences and technical background, therefore with a large level of subjectivity, low standardization, low quantitative measurement of the general costs of that adoption, and with a limited vision of the degree of business alignment. In order to mitigate such problems, this thesis proposes a method that systematizes the process of analysis about the adoption or not of services in given software solutions. The method is composed of

15 interrelated estimation factors, which are organized and sequentially executed by the IT manager, who feeds them with information related to the company and to the intended solution. Each factor has actually a formula and generates a value. Picking each candidate business functionality, the method synthesizes all factors' values and provides concrete numbers for costs, effort, development time and degree of alignment to the business at the end for each individual functionality and as a group. The method was developed based on the best practices of software engineering and SOA, adapting them to a multi-criteria decision-making scenario and comparing it to the general costs of developing software in the traditional way. A number of software companies was used to evaluate the method, where they could confirm its value proposition.

Keywords: SOA, software analysis method, software service.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

AMD	Auxílio Multicritério à Decisão
API	Application Programming Interface
BPMN	Business Process Management Notation
BPM	Business Process Management
CORDIS	Community of Research and Information Services
CMMI	Capability Maturity Model Integration
DFM	Diretriz Funcional do Modelo
EF	Environmental Factor
ESB	Enterprise Service Bus
FP	Function Points
GQM	Goal Question Metric
GSIGMA	Grupo de Sistemas Inteligentes de Manufatura ∑ Redes Colaborativas
IC	Índice de Consistência
ICT	Information and Communications Technology
KPI	Key Performance Indicator
LOC	Lines of Code
NPO	Número de Pontos de Objeto
P&D	Pesquisa e desenvolvimento
PCU	Pontos de Casos de Uso
PDS	Processo de Desenvolvimento de Software
PERT	Program evaluation and Review Technique
PMBOK	Project Management Body of Knowledge
PME	Pequenas e Média Empresa
PMI	Project Management Institute
PRINCE	Projects in a Controlled Environment
QoS	Quality of Service
RCs	Redes colaborativas
REST	Representation State Transfer
SaaS	Software as a Service
SLR	Systematic Literature Review
SOA	Service-oriented architecture
SOAP	Simple Object Access Protocol
SOC	Service-oriented Communications

SPI	Software process improvement
TFC	Technical Complexity Factor
TI	Tecnologia da Informação
TIC	Tecnologia da Informação e Comunicação
UAW	Unadjusted Actor Weights
UCP	Use Case Points
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
W3C	World Wide Web Consortium
WS	Web Service
WSDL	Web Service Description Language
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

A globalização tem pressionado as empresas a reagirem mais rapidamente às mudanças que vêm surgindo fruto dos novos cenários de negócios, inovação e sustentabilidade. Uma das ações nessa direção tem a ver com o crescente emprego das mais variadas TICs (Tecnologias de Informação e Comunicação) no suporte aos inúmeros processos de negócio das empresas (FUGITA; HIRAMA, 2012). Por outro lado, como uma das consequências, as suas arquiteturas computacionais têm se expandido e se constituído cada vez mais por sistemas distribuídos e heterogêneos, porém com ainda grandes dificuldades de suportar uma maior flexibilidade às mudanças, evoluções e escalabilidade dos processos e sistemas (ALVES, 2008).

SOA (*Service Oriented Architecture* - Arquitetura Orientada a Serviços) tem emergido como um poderoso paradigma e estilo arquitetural para fazer frente às dificuldades daquele cenário, potencializando, tanto um maior alinhamento das soluções de TIC aos negócios, como uma maior agregação de valor dos ativos de software às empresas (PAPAZOGLOU, 2012).

A adoção de SOA pode aumentar a competitividade das empresas na medida que se torna uma alternativa aos tradicionais e mais estanques modelos de aquisição, implantação e manutenção de software (FUGITA; HIRAMA, 2012). Isto se torna ainda mais relevante quando se considera a inter-relação de SOA com outros igualmente recentes e impactantes modelos de TIC, como os de Computação em Nuvem (*Cloud Computing*) e Internet das Coisas (*Internet of Things*), e os novos modelos de negócios baseados no acesso, uso e pagamento sob demanda, no escopo da Computação Orientada a Serviços (*Service Oriented Computing*) (PAPAZOGLOU, 2012).

Diferentemente da arquitetura tradicional de software, classicamente como um sistema "monolítico" e de alta granularidade, um sistema construído numa ótica SOA é constituído por uma composição de pequenos módulos de software, distribuídos, desacoplados, reutilizáveis e heterogêneos - chamados *serviços* - cujas funcionalidades são invocadas via rede dentro de variadas e flexíveis lógicas de processos de negócio. Neste sentido, SOA introduz uma nova visão de projeto, desenvolvimento, reuso, integração e manutenção de software, podendo um dado sistema ser fornecido e evoluir mais flexivelmente sob diversas arquiteturas e modelos de negócios, e com

um maior grau de alinhamento à camada de negócio da empresa (ERL, 2009) (BERNSTEIN; HAAS, 2008).

Dado que SOA é acima de tudo um estilo arquitetural em uma perspectiva de computação orientada a serviços, o desenvolvimento de uma dada solução de software para fazer frente aos requisitos funcionais e não funcionais de processos de negócio implica que todos os sistemas envolvidos numa arquitetura sejam vistos como "provedores de serviços". Porém, quando analisado sob uma ótica mais ampla, de SOSE (*Service Oriented Software Engineering*), há uma série de aspectos a considerar, que tornam projetos SOA usualmente bastante complexos, como (RABELO; NORAN; BERNUS, 2015):

- Cada serviço pode ser implementado com TICs diferentes, usando componentes proprietários, acarretando problemas de interoperabilidade, escalabilidade e de *lock-in* tecnológico e/ou comercial.
- Serviços podem ser de diversos tipos, englobando serviços de aplicação, de infraestrutura, de comunicação, de segurança, de persistência de dados, entre outros, sejam serviços nativamente desenvolvidos como tal, sejam serviços que encapsulam funcionalidades implementadas em sistemas legados ou embarcados, e cujas funcionalidades podem ser total ou parcialmente públicas para serem invocadas.
- Serviços tem interfaces, que expõem as suas funcionalidades e expressam a sua granularidade, impactando diretamente a capacidade de reuso.
- Serviços podem ter diferentes modelos de implementação, desde um "clássico" monolítico (mesmo que de pequena granularidade) a mais flexíveis, como um modelo 3 camadas, onde as camadas de apresentação, processo e dados são também desacopladas e distribuídas em um ou mais servidores, o que impacta por exemplo no reuso e desempenho.
- Serviços devem ser conhecidos pelos devidos atores de invocação, podem ter variados graus de acoplamento, e podem estar implantados e postos em execução em um ou diversos servidores em heterogêneas plataformas de software/hardware e domínios de segurança, suportando diferentes níveis de qualidade de serviço (QoS – *Quality of Service*), acordadas e expostas nos seus contratos (SLA – *Service Level Agreement*). Apesar desta flexibilidade, isso pode requerer mais complexas infraestruturas e *middlewares* de suporte para lidarem com

inúmeros problemas intrínsecos a sistemas distribuídos de forma a monitorar e garantir o *end-to-end QoS* requerido pelo processo de negócio.

- Serviços podem ser implementados com diferentes tipos de interfaces de apresentação (por exemplo, para diferentes *clients* e empresas usuárias) e podem ter diferentes modos de controle de acesso e de pagamento (*billing*), conforme os modelos de negócio adotados pelos seus provedores. Isso pode implicar no uso de mais robustos modelos de governança para lidar, por exemplo, com "conformidade", gestão de versionamentos e ciclo de vida de cada serviço.
- Serviços podem ser providos por diferentes provedores de software, com diferentes – e, portanto, nem sempre harmonizados / padronizados - graus de qualidade de (processos de) desenvolvimento. Isso pode gerar problemas de robustez ou instabilidade do sistema devido a uma não adequada qualidade do serviço e ao mesmo tempo exigir da empresa investimentos em infraestrutura de rede, servidores e *middlewares*, e em pessoal bastante qualificado para gerenciar o ambiente.
- Serviços podem ser implementados para terem algum grau de autonomia e de inteligência, podendo gerar comportamentos indesejáveis, não determinísticos e uma inviabilização de coreografias.
- O conceito de SOA pode ser aplicado tanto dentro de um único software como numa arquitetura que integre múltiplos softwares heterogêneos distribuídos, pertencentes ou não à uma mesma empresa.

Portanto, se por um lado SOA tem grandes potencialidades e tem havido grandes avanços nos ambientes de desenvolvimento, normas e modelos de maturidade para ajudar no desenvolvimento de soluções baseadas em SOA (PEREPLETCHIKOV; RYAN; TARI, 2013), por outro lado projetos SOA acabam por ser mais complexos de serem projetados, geridos, integrados e implantados e, assim, geralmente têm um risco maior de insucesso quando comparado com um projeto de software tradicional (PAPAZOGLOU, 2012). Além disso, dependendo do tipo de demanda de software a ser atendida, do que já existe de sistemas legados, dos requisitos do negócio em questão e de um grande número de outros aspectos - como os técnicos acima citados - nem

sempre a adoção de SOA / orientação a serviços pode vir a ser a melhor decisão para o caso (SCHEPERS; IACOB; VAN ECK, 2008).

A adoção de SOA por parte de uma empresa tem vários impactos, em diferentes dimensões, além da meramente tecnológica/TIC. Envolve finanças, modelos de negócios, cultura organizacional, processos, qualificação de recursos humanos, entre outras dimensões, além de uma cuidadosa análise se a empresa está suficientemente preparada para adotá-lo. Uma não adequada avaliação desses aspectos é tido como uma das principais causas de insucesso de projetos SOA nas empresas (MEULEN; PETTEY, 2009).

Neste sentido, torna-se relevante às empresas, não apenas conhecerem os impactos do uso de SOA como paradigma de base aos seus projetos e implementações, mas igualmente saberem se seu uso faz ou não sentido para os variados casos de negócio em que estão inseridas (SCHEPERS; IACOB; VAN ECK, 2008) (KAJKO; LEWIS; SMITH, 2007).

Esta relevância também se justifica ao se analisar o papel que o setor de TI desempenha nas economias nacional e mundial (KRAMER; JENKINS; KATZ, 2007) e, por conseguinte, da importância de haver melhores suportes metodológicos para análise de projetos SOA. Por exemplo, só na Europa há mais de 50 mil empresas de TI (OFFICE, 2017). De forma similar ao restante no mundo, o setor no Brasil é constituído majoritariamente (94%) por Micro, Pequenas e Médias Empresas (ABES, 2016) que, via de regra, têm grandes limitações para se manterem sustentáveis em um mercado cada vez mais competitivo e movido pela inovação (WESTPHAL; THOBEN; SEIFERT, 2010). Das cerca de 14 mil empresas de TI no Brasil, em torno de 9 mil são formalmente de desenvolvimento de software (ABES, 2016).

Marks e Bell (2006) ressaltam que a decisão sobre o uso de SOA por uma organização se inicia com uma análise criteriosa do negócio a fim de determinar como será a utilização de serviços em uma arquitetura global SOA na qual os vários sistemas da empresa possam usufruir de inúmeros serviços comuns uns aos outros. Azevedo *et. al.* (2009) observa que faltam metodologias sistemáticas para a implantação de projetos SOA. O´Brien (2009) destaca as limitações das abordagens atuais para analisar sistematicamente o alcance e a dimensão dos vários tipos de projetos de SOA, determinar o que está envolvido em tais projetos, e estimar o esforço necessário e seus custos.

1.1 PROBLEMA ESPECÍFICO

Vários autores, como em O´Brien (2009), Meulen e Pettey (2009), Papazoglou (2012), Fiammante (2009) e Marzullo (2009), ressaltam a complexidade de se considerar todas as dimensões de um projeto SOA para a sua adoção pelas empresas.

Esta tese foca no processo de análise e decisão sobre quando adotar uma abordagem baseada em desenvolver software como serviços para uma certa demanda sob uma perspectiva mais ampla, SOA, ao invés de o fazer da forma "tradicional", monolítica, não orientada a serviços.

Estimação de software tem sido a abordagem mais utilizada para embasar os gestores de TI nessa decisão (McCONNELL, 2006) (ERL, 2009) (PAPAZOGLOU, 2012). Estimação pode ser entendida em termos gerais como um *"processo de predizer o tempo e esforço necessários para desenvolver ou dar manutenção a um software"* (McCONNELL, 2006).

Porém, as características intrínsecas de SOA (anteriormente mencionadas) fazem com que o processo de estimação numa ótica SOA/de serviços não seja simples. A literatura mostra uma lacuna sobre métodos mais completos ou que não sejam apenas representados por *checklists* genéricos para apoiar gestores de TI nessa decisão. Alguns artigos, observações empíricas e relatos na Internet também comentam que isso é geralmente feito de forma não sistematizada, não padronizada, incompleta (dado que são inúmeros e adicionais aspectos a serem considerados numa decisão que também envolva serviços), nem sempre completamente correta (pois analisam serviços da mesma forma como que se fossem aplicações monolíticas) e com elevada margem de subjetividade, baseando-se sobremaneira na experiência do arquiteto / analista de TI (MAJLESI; MEHRPOUR; MOHSENZADEH, 2012).

Neste caso, a falta de uma metodologia decisória aumenta a probabilidade de se implementar serviços não alinhados à estratégia, levando o projeto SOA a resultados aquém dos esperados ou até mesmo ao seu fracasso (ERL, 2009). Uma análise não adequada implica numa estimação incorreta, acarretando maiores risco e custos gerais de desenvolvimento e posterior maior manutenção do software (MAJLESI; MEHRPOUR; MOHSENZADEH, 2012).

Essa dificuldade é exemplificada no trabalho de Meulen e Pettey (2009). Baseado em uma pesquisa com 20 empresas, 50% dos projetos de SOA fracassaram, enquanto outros 30% não conseguiram comprovar a sua validade para as empresas. Muitos dos que implantaram projetos

considerados bem sucedidos focaram apenas em um problema/processo específico, fazendo com que inúmeros serviços fossem construídos para uma única aplicação, sem uma visão de reuso e de integração global dentro de uma arquitetura mais ampla e escalável de orientação a serviços.

Após avaliação do estado da arte e da prática (apresentada nos Capítulos 3 e 4), verificou-se que existem várias importantes iniciativas sobre projetos SOA. Porém, elas atacam perspectivas isoladas destes (por exemplo, apenas custos ou tempo) e não olham tais perspectivas e fatores envolvidos de uma forma integrada e globalmente coerente.

1.2 PERGUNTA DE PESQUISA

Levando em consideração os pontos levantados até então e o problema específico, a pergunta de pesquisa desta tese de doutorado é:

> *Quais são os fatores a serem considerados num processo de estimação de um projeto SOA, e como tais fatores podem ser organizados de forma a sistematizar o processo de análise e decisão sobre implementar ou não um software na forma de serviços em uma perspectiva SOA ?*

1.3 OBJETIVO GERAL

Segundo Silva e Menezes (2005), os objetivos de uma pesquisa são explicitados para caracterizar o seu alcance e utilizados para delimitar o seu problema. Os objetivos estabelecidos nesta pesquisa são apresentados na forma de objetivo geral e objetivos específicos.

> O objetivo geral desta tese é *conceber um método de estimação de software que auxilie gestores de TI com um maior rigor técnico no processo de análise e decisão em projetos SOA.*

De acordo com a ABNT NBR ISO 9000:2000, *processo* é um *"conjunto de atividades inter-relacionadas ou interativas que transforma insumos (entradas) em produtos (saídas)"*. No contexto geral de sistemas de informação, *método* pode ser definido como um *"conjunto disciplinado e estruturado de passos para se resolver um problema, dentro de um contexto, dentro de premissas, através do uso de técnicas, ferramentas e princípios organizacionais"* (KIRIKOVA *et al.,* 2012). KIRIKOVA *et al.* (2012) também colocam que um método

pode ser organizado na forma de processos sistematizados mais especializados que organizam atividades visando se resolver um dado problema ou classe de problema.

O método neste tese é constituído por 15 fatores inter-relacionados, que são organizados e sequencialmente percorridos pelo gestor de TI. O método guia e auxilia o gestor de TI no provimento das informações necessárias sobre a empresa e a solução pretendida. Em cada etapa o método sintetiza rigorosamente os valores dos fatores para cada uma das funcionalidades-candidatas do negócio sendo analisado, tanto individualmente quanto em subconjuntos de composições, gerando ao final números concretos para os quatro principais elementos de decisão de implementação de projetos SOA: *custos, esforço, tempo de desenvolvimento* e *grau de alinhamento ao negócio* (segundo McCONNELL, 2006) (ERL, 2009) (PAPAZOGLOU, 2012). Apesar do método sugerir, baseado na estimação quantitativa, quais funcionalidades são mais recomendáveis e prioritárias de serem implementadas como serviços e quais como software tradicional, a decisão final não é automatizada, mas sim deixada a critério do gestor de TI da empresa.

O método foi desenvolvido com base nas melhores práticas de engenharia de software e de SOA, adaptando-as a um cenário de tomada de decisão multicritério e de comparação em relação a custos gerais de desenvolvimento de um software tradicional.

O método é aberto para considerar variadas tecnologias de implementação de serviços, como *web services*, *microserviços*, *REST* etc.

Em termos gerais, considera-se que os *gestores de TI* sejam os usuários por excelência do método, podendo representar tanto *software-houses* desenvolvedoras como organizações usuárias de soluções (que por sua vez podem ter também um departamento de desenvolvimento de sistemas e que também podem terceirizar certas atividades de desenvolvimento).

1.4 OBJETIVOS ESPECÍFICOS

Para atingir o objetivo geral desta tese, os seguintes objetivos específicos foram delineados:

- Elicitação dos fatores de análise em projetos SOA, tanto os operacionais como os de negócio;

- Definição dos elementos de análise a serem computados em cada fator;
- Definição da fórmula de cálculo de cada fator;
- Definição da sequência de cálculo dos diversos fatores e suas inter-relações visando sistematizar o processo decisório;
- Aplicação de método multicritério para um ranqueamento das funcionalidades prioritárias de serem implementadas tendo em vista o alinhamento ao negócio;
- Implementar um pequeno sistema que auxilie na computação dos fatores do método.

1.5 ASPECTOS DE INEDITISMO, CONTRIBUIÇÃO CIENTÍFICA E PROPOSIÇÃO DE VALOR

Como será explanado no Capítulo 4, não se encontrou na literatura e em nenhum relato do estado da prática um método de estimação voltado para projetos SOA que mensure e integre num mesmo "ambiente" (método) as quatro principais dimensões de análise (custo, tempo, esforço e alinhamento) para fins de decisão sobre quais funcionalidades de uma certa demanda de software são objetivamente justificáveis de receber investimentos para serem implementadas como serviços e assim se tornarem ativos de software de uma empresa.

Frente ao atual estado das pesquisas na área de estimação em SOA, o método desenvolvido visa propiciar as seguintes melhorias:

- Sistematização do processo: os fatores mais importantes para a estimação de um projeto SOA são explícita e formalmente considerados, rigorosamente calculados com base em referencial teórico e melhores práticas, e cuidadosamente correlacionados para análise e tomada de decisão;
- Esta sistematização dá base para processos decisórios com menor subjetividade, menos erros, melhoria contínua, potencializando uma tomada de decisão com maior confiança e assim que o método se torne uma prática de processo da empresa de análise em SOA;
- Esta sistematização potencializa uma análise e decisão mais rápidas na medida que um sistema computacional de suporte automatiza uma série de cálculos e comparativos;

- Uma análise multicritério permite aos gestores ponderar importâncias nos fatores de decisão com base nas características de cada demanda/negócio;
- O método permite comparar os custos gerais de uma demanda de software via uma abordagem SOA ou via uma abordagem tradicional, não SOA.

O método desenvolvido provê outras funcionalidades consideradas bastante importantes em projetos de software e SOA:

- Sugere uma priorização de ordem de desenvolvimento das funcionalidades, o que é importante para fins de gestão de projetos e de previsão de orçamentos da empresa;
- Analisa também globalmente as funcionalidades, sugerindo quando parece ser mais conveniente desenvolver conjuntamente duas ou mais funcionalidades como um (1) único serviço (ou como software tradicional, se for o caso) ao invés de um (1) serviço por funcionalidade; portanto pode ajudar na definição da granularidade e aspectos de reuso;
- Considera as vantagens e o custo do reuso;
- Permite avaliar o impacto da adição de um ou mais colaboradores na equipe de desenvolvimento do software no tempo e custo da solução, funcionando como uma espécie de "simulador" de apoio à gestão do projeto;
- Serve de referência para custos e prazos de desenvolvimento e de entrega quando da aquisição de software, de subcontratações ou terceirizações de desenvolvimentos; ou ainda de base de decisão sobre optar por acessar/pagar por uma dada funcionalidade ou serviço via um modelo SaaS (*Software-as-a-Service*) ao invés de desenvolvê-lo internamente.

Esses supostos ganhos advindos com o método desenvolvido consideram uma estimação SOA na sua perspectiva mais ampla, de ciclo de vida, de alinhamento ao negócio e de consideração das realidades cultural, organizacional, financeira e tecnológica de uma empresa. Portanto, uma maior acurácia - e mesmo uma maior utilidade do método - tendem a ser verificadas quando aplicado nessa perspectiva. Porém, apesar disto, o método não tem restrições em ser aplicado numa perspectiva simplista e talvez nem mesmo considerados como "projetos SOA" no seu sentido mais rigoroso. Por exemplo, em cenários

puramente tecnológicos e isolados, de análise de uma mera implementação de um certo serviço de software voltado para atender a um dado sistema tradicional (a ser por este depois invocado) e então se estimar o seu custo geral versus o de não desenvolvê-lo como um serviço.

1.6 DELIMITAÇÃO DO TRABALHO

A implementação de serviços baseados em SOA envolve inúmeros aspectos de uma empresa, cobrindo aspectos tão variados como os de recursos humanos, legais, de infraestrutura de TIC, financeiro, tecnológico, organizacional, cultural, governança corporativa e de TI, entre outros. Não está no escopo do método proposto cobrir modelos de melhorias sob quaisquer perspectivas que não a de TI, embora o método procure refletir nos seus fatores vários daqueles aspectos.

Um projeto SOA envolve várias etapas, conhecidas como *ciclo de vida*. A Figura 1 mostra tais etapas, segundo o modelo de Papazaglou (2006), um dos de referência na literatura. O método proposto nesta tese foca na etapa de *Análise* de projetos SOA.

De acordo com a NBR ISO/IEC 12207:1998:

- O ciclo de vida é a "estrutura contendo processos, atividades e tarefas envolvidas no desenvolvimento, operação e manutenção de um produto de software, abrangendo a vida do sistema, desde a definição de seus requisitos até o término de seu uso."
- Tarefa é uma ação com entradas e saídas. Pode ser um requisito, recomendação ou permissão.
- Atividade é um conjunto de tarefas.
- Etapa é o avanço de uma mesma coisa numa marcha, no tempo e/ou no espaço, que lhe altera a situação sem alterar-lhe a essência do conteúdo.

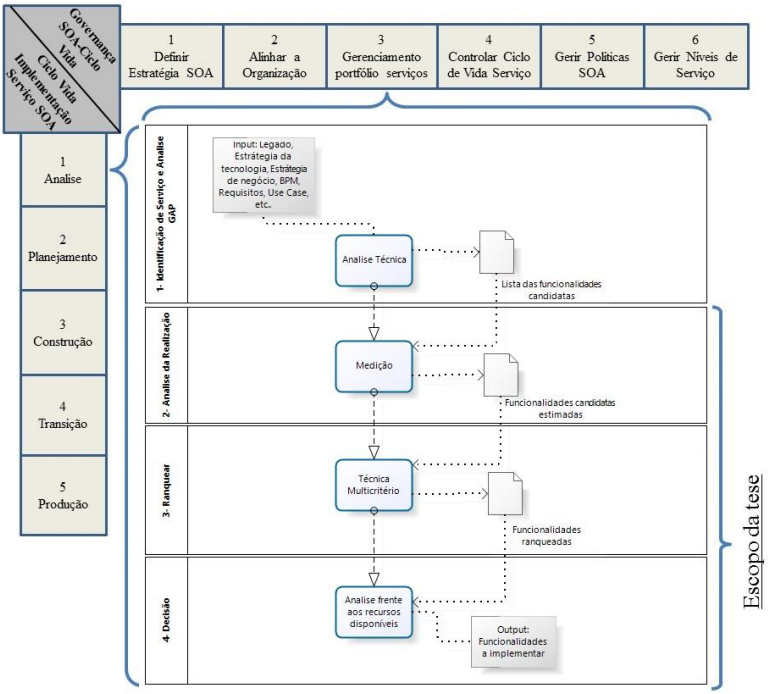


Figura 1: Delimitação da Tese
Fonte: Autor.

A etapa de Análise é composta por um conjunto de três atividades: *identificação de serviços candidatos*, *análise de gap* e *realização*.

De forma geral, a primeira consiste numa razoavelmente complexa atividade que, em suma, se desdobra numa lista de requisitos funcionais necessários para as demandas de negócios fechados, ou mesmo de demandas previstas ou planejadas para mais longo prazo em função das estratégias da empresa. Esta lista pode ser expressa de várias formas de acordo com as práticas da empresa; por exemplo, via *Casos de Uso UML*. A atividade de análise de *gap* visa basicamente confrontar o que é demandado versus os ativos de software já existes na empresa, identificando assim então o que não tem como ser reaproveitado. A atividade de *realização* essencialmente consiste na decisão final sobre o que será implementado (como serviços de software) após feita a *análise de gap*.

O método de estimação desenvolvido atua imediatamente após a atividade de análise de gap.

A contribuição desta tese se situa mais diretamente na atividade de *realização*. Porém, a estende, ponderando os pesos de importância dos fatores de análise consoante ao negócio em questão, ranqueando as funcionalidades preliminarmente decididas de acordo com o grau de alinhamento ao negócio, e finalmente comparando os seus custos gerais com os de desenvolvimento tradicional.

Um projeto SOA deve considerar também atividades de governança, como igualmente mostrado na figura 1 (BROWN; LAIRD; CLIVE, 2009). Porém, o método desenvolvido não visa propor melhorias em modelos de governança SOA, mas sim atuar na melhoria de processos de governança relativos à análise e seleção de serviços. Neste sentido, embora o contexto de governança seja amplo e deva estar presente em qualquer atividade, considera-se que o método proposto oferece algumas contribuições mais diretamente ligadas às etapas 2 e 3 (alinhamento com a organização e gerenciamento de portfólio).

Embora facilite a aplicação do método e seja uma prática usual nas empresas de software, o método proposto não exige ou depende de que a empresa utilize artefatos adotados nas várias metodologias de engenharia de software, por exemplo, diagramas de *Caso de Uso* UML.

Do ponto de vista de riscos, não se pretende fazer uma análise propriamente dita de riscos ou de sua gestão. O método dá suporte a uma rigorosa estimação de software e faz comparativos gerais, provendo valores concretos e mensuráveis para as várias dimensões *da etapa de análise* de um projeto SOA. Com isso, tem-se como premissa que o gestor de TI terá uma bem maior base de avaliação e assim um potencial menor risco da decisão na etapa de análise em projeto SOA.

1.7 ADEQUAÇÃO ÀS LINHAS DE PESQUISA DO PROGRAMA

O trabalho aqui apresentado se enquadra na linha de concentração de sistemas do Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina, dentro da linha de pesquisa de Informática, na área de Engenharia de Software.

Este trabalho está integrado com os trabalhos realizados no grupo de pesquisa GSIGMA (Grupo de Sistemas Inteligentes de Manufatura ∑ Redes Colaborativas).

1.8 NOTA SOBRE A TRADUÇÃO DOS TERMOS EM INGLÊS

Por se tratar de um tema onde grande parte da literatura está escrita em língua inglesa, optou-se por manter alguns termos na tese em inglês. Isto se deveu ou por falta de um termo substituto semanticamente mais adequado em língua portuguesa ou mesmo pelo desuso de termos traduzidos em literatura local. Dessa forma, optou-se por utilizar os seguintes critérios gerais para o uso de termos em inglês nesta tese:

- Utilizar os termos originais em inglês quando o termo original for considerado "aceitável" em português (e.g. *groupware*, *firewall*). Neste caso, usa-se o tipo de letra *itálico* para indicar mais claramente a procedência do termo;
- Utilizar o termo em inglês quando este for uma sigla internacionalmente difundida (e.g. *HTTP*, *UML*, *XML*). Neste caso, mantem-se a marcação em itálico para destacar a procedência;
- A sigla de Arquitetura Orientada a Serviços (SOA, do Inglês *Service Oriented Architecture*), um dos focos desta tese, não foi traduzida pois é o termo utilizado na literatura corrente mesmo em português; e

Todos os outros casos foram traduzidos para o português. Todavia, as traduções têm uma nota de rodapé ou a apresentação do termo/frase original entre parêntesis.

1.9 ESTRUTURA DO TRABALHO

O trabalho está estruturado em 8 capítulos. No Capítulo 1 foi apresentada a problemática, a pergunta de pesquisa, os objetivos da pesquisa e a delimitação do trabalho. O Capítulo 2 descreve a metodologia utilizada de condução da pesquisa. O Capítulo 3 apresenta a revisão de literatura sobre os conceitos científicos de base para o modelo desenvolvido. O Capítulo 4 apresenta a revisão do Estado da Arte na área da tese de forma a realçar os seus aspetos de ineditismo. O Capítulo 5 apresenta e detalha o método de estimação desenvolvido. O Capítulo 6 fornece um exemplo de aplicação do método. O Capítulo 7 detalha os aspectos de verificação, avaliação e validação do método. Finalmente, o Capítulo 8 apresenta as conclusões finais e sugestões de trabalhos futuros.

2 METODOLOGIA DA PESQUISA

O objetivo desse capítulo é apresentar de forma concisa como foi projetado e conduzido este projeto de pesquisa, que culminou com um método para apoio à decisão sobre a implementação de serviços de software em uma perspectiva SOA.

Segundo Gil (2010), a pesquisa é o procedimento racional e sistemático que tem como objetivo proporcionar respostas aos problemas que são propostos. A pesquisa se desenvolve com a utilização de planejamento, métodos, técnicas e outros procedimentos científicos ao longo de um processo que envolve inúmeras atividades até a apresentação dos resultados finais de acordo com os objetivos traçados.

Considerando os objetivos desta pesquisa, o método de pesquisa utilizado para desenvolver a proposta desta tese foi o *Design Science* (JÄRVINEN, 2004). Existem inúmeras definições para esse método, conforme o domínio de aplicação e áreas envolvidas (como as ciências sociais e as tecnológicas). Com base nas análises de Järvinen (2007), pode-se dizer em termos gerais que *Design Science* se refere ao desenvolvimento de artefatos (de conhecimento, de TI, ideias, teorias, etc.) para se resolver ou melhorar um dado problema já suficientemente compreendido. Isto é feito através de melhoramentos efetuados ciclicamente a partir de teorias e conhecimentos já existentes, e cujos resultados das melhorias possam ser concretamente avaliados, e, por fim, modificar o estado de conhecimento vigente. A Figura 2 ilustra o método *Design Science*.

Ressalta-se que no *Design Science* pode haver um avanço e retorno de etapas, em ciclos, não necessariamente simétricos, dependendo da dinâmica determinada dos pesquisadores, do número de artefatos sendo tratados, das suas inter-relações, e da situação pesquisada (JÄRVINEN, 2004).

Neste sentido, todas as atividades da tese se enquadraram nas etapas do processo desse método de pesquisa e que serão sumarizadas nas seções a seguir. O detalhamento de cada uma dessas etapas dar-se-á através dos capítulos subsequentes.

A adoção do *Design Science* significa essencialmente também a adoção de uma estratégia de pesquisa *incremental*, onde já consolidadas fundamentações teóricas existentes de engenharia de software serão aproveitadas, tomadas como base e adaptadas para o desenvolvimento do método proposto voltado para SOA.

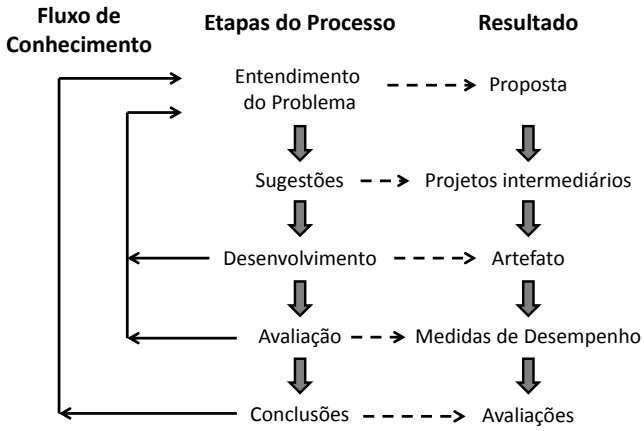


Figura 2: Visão geral da metodologia *Design Science*
Fonte: Adaptado de Järvinen (2007).

2.1 ENQUADRAMENTO METODOLÓGICO

O enquadramento metodológico define as condições de contorno existentes ou assumidas dentro das quais o experimento de pesquisa será realizado e que contribuem para o planejamento da execução pesquisa. São elas, com base principalmente em Berto e Nakano (1998) e Gil (2010):

- Premissa de pesquisa: *Convalida* e *Afirmativa*, de que a ausência de métodos sistematizados e mais amplos de análise e de concreta estimação de projetos SOA aumentam os custos e riscos gerais destes;
- Método de pesquisa: *Design Science*. Busca-se intervir diretamente para a concepção de um modelo voltado para a solução de um problema concreto e específico; no caso, de como melhorar o processo e melhor mensurar os fatores de decisão sobre desenvolvimentos de software baseados em serviços em uma perspectiva SOA, tendo como base inicial metodologias e técnicas de engenharia de software "clássicas".

- Natureza da pesquisa: *Aplicada*, pois se visa aplicar e adaptar uma série de teorias básicas já existentes para desenvolver o modelo decisório para SOA;
- Abordagem do problema: *Quali-quantitativa*, pois se pretende desenvolver um método (qualitativo-quanti) de decisão (o modelo em si do método, constituído por artefatos conceituais; e suas fórmulas, baseadas em modelos matemáticos) e cujos resultados serão avaliados também de forma quali-quantitativa (os resultados de estimações SOA e opinião de alguns usuários especialistas);
- Tipo de pesquisa: basicamente *Indutivo*, buscando uma generalização de fatores de decisão em estimação SOA com base em teorias já existentes e melhores práticas, e uma posterior sistematização desses fatores.
- Paradigma de pesquisa: *Positivista*. Assume-se que a realidade (problema de estimação em SOA) é tangível, palpável e passível de alguma predição e controle; há separação entre o pesquisador e o objeto de estudo; assume que é possível desenvolver um corpo de conhecimento genérico, independente de tempo e espaço;
- Objetivo de pesquisa: *Ação*, não visando explicar, descrever, predizer ou desvendar o fenômeno da estimação SOA, mas sim de agir para a construção de um melhor método de estimação;
- Tempo de pesquisa: *Estudo Longitudinal*, com os experimentos, contatos com empresas, refinamento do modelo, análise de dados, etc., sendo executados ao longo do desenvolvimento do método ao invés de em apenas uma única e pontual vez;
- Observação: variadas formas, de acordo com / ao longo das diferentes etapas de desenvolvimento do método de estimação dentro do *Design Science*. Foram previstas observações *factuais* (por exemplo, quando do preenchimento das planilhas de estimação pelos usuários das empresas) e os devidos registros e análises de forma individual pelo pesquisador; *sistemáticas*, feitas em condições controladas para observar / responder a propósitos previamente estabelecidos (por exemplo, para a realização de workshops e entrevistas com empresas na etapa de análise de requisitos do modelo); e *não participante* (por exemplo, para propiciar aos respondentes dos questionários

liberdade para expor suas opiniões sobre o método desenvolvido e suas melhorias);

- Procedimentos: igualmente variados, de acordo com as etapas do *Design Science*. Foram previstas *Pesquisas bibliográficas* para melhor entender o problema, cobrindo tanto os assuntos mais relacionados à tese (engenharia de software, SOA e técnicas multicritério) como de levantamento do estado da arte em bases de dados científicas visando identificar trabalhos correlatos; *Pesquisas documentais*, para levantar práticas de empresas de diferentes tamanhos e maturidades SOA sobre que tipo de processos, documentos, informações, etc., usam numa estimação SOA; *Levantamento*, para se poder obter informações dos membros das empresas sobre como fazem uma estimação SOA; *Participante*, para permitir o desenvolvimento do método a partir da interação entre o pesquisador e membros de empresas; *Experimental*, para realizar diversos testes e análises das várias evoluções do método de estimação desenvolvido assim como para sua verificação, avaliação e validação.

2.2 PROCEDIMENTOS PARA ELABORAÇÃO DO MODELO

As seções a seguir sumarizam os passos metodológicos realizados ao se seguir o *Design Science*.

2.2.1 PASSO 1 - ENTENDIMENTO DO PROBLEMA

Este passo iniciou-se pela identificação do problema em si. Isto deu-se a partir de leituras de artigos científicos, páginas e blogs técnicos na Internet, e de experiências empíricas relatadas por empresas de software que têm lidado com arquitetura SOA. Complementarmente, utilizou-se da experiência profissional do autor, que vivenciou várias iniciativas de implantação de SOA em clientes enquanto colaborador de uma grande empresa de software do norte de Santa Catarina.

Considerando o objetivo da tese, foi necessário compreender inicialmente as teorias de base existentes a ela relacionadas. Mais especificamente, sobre SOA, sobre o processo geral de estimação de software, e sobre alguns modelos de referência e técnicas de estimação.

Com base nas leituras e análises, verificou-se que um dos vários problemas na estimação tinha a ver com limitações variadas em termos

de “processo” de estimação assim como de mensurações mais concretas voltadas para projetos SOA.

Dado o grande escopo envolvido quando se considera o ciclo de vida de software & SOA e governança de TI, o passo de entendimento do problema foi de certa forma finalizado com a delimitação do método às atividades de análise e realização, e apoio à decisão de projetos SOA bem como com uma proposta mais concreta do tipo de método que seria desenvolvido e dos objetivos preliminares de pesquisa.

Os resultados estão explanados nos Capítulos 1, 3 e 4 desta tese.

2.2.2 PASSO 2 - SUGESTÕES

Neste passo foi feito um levantamento bibliográfico com o objetivo de consolidar os conceitos relacionados à proposta geral do método bem como iniciar um aprofundamento na identificação dos fatores envolvidos no processo de decisão SOA dentro daquela delimitação. Este levantamento foi realizado através da leitura de livros e artigos de congressos e periódicos, utilizando-se essencialmente a metodologia de Revisão Sistemática da Literatura sobre algumas bases de dados científicas, complementada com acessos a outras fontes. Paralelamente, procurou-se identificar com um grau maior de precisão os pontos de potencial ineditismo do trabalho pretendido.

Um primeiro projeto de método de estimação SOA foi idealizado e de forma cíclica foi sendo melhorado, inspirado no método espiral iterativo das metodologias ágeis de engenharia de software.

Tais melhorias vieram de três fontes. A primeira foi da contínua análise crítica em si do método por parte deste autor em função de permanentes leituras de artigos. A segunda foi através da publicação de um artigo em uma conferência internacional (GHODDOSI; RABELO, 2015) na área onde se pode aproveitar uma série de comentários dos revisores e da audiência. A terceira fonte foi a realização de dois workshops com duas empresas de software do norte de Santa Catarina com experiência em SOA, onde o método fora apresentado, exemplos foram discutidos sobre como a versão corrente do método atendia as necessidades de estimação, e finalmente opiniões foram coletadas e ponderadamente consideradas. As discussões foram moderadas / guiadas pelo autor tendo em vista o objetivo do experimento metodológico, mas sem, por exemplo, questionários estruturados.

Os resultados estão explanados no Capítulo 5 desta tese, tendo como base essencial os Capítulos 3 e 4.

2.2.3 PASSO 3 - DESENVOLVIMENTO

Este passo é representada pela versão final do método de estimação SOA, ou seja, do *artefato* desenvolvido e avaliável.

Da mesma forma que no passo anterior, um artigo foi publicado em uma conferência internacional na área, onde igualmente se pôde aproveitar uma série de comentários dos revisores e da audiência.

O método construído se constitui de uma generalização de uma série de técnicas de estimação, algumas plenamente empregues num cenário SOA, outras com necessárias adaptações / extensões para tal. Disto emergiram uma série de "fatores" de decisão, que foram inter-relacionados, formando uma "taxonomia" e um "processo". A "lógica" desse processo foi inspirada em vários modelos de engenharia de software e é explicada no Capítulo 5.

Cada fator corresponde a um artefato em si, composto por uma série de variáveis / elementos de análise, que são calculados matematicamente. As variáveis, seus valores e/ou sua consideração ou não em cada fórmula foram também inspiradas em variadas técnicas de estimação, e adaptadas para SOA.

A medida que a taxonomia é percorrida, valores calculados dos fatores anteriores são sintetizados e agregados novos aspectos de análise. Ao final, valores para custo financeiro, esforço, tempo de implementação e nível de alinhamento ao negócio são calculados, tanto com relação a um desenvolvimento como serviços como de forma 'tradicional'. Com isso, os gestores de TI podem analisar esses números, ponderar perante as demandas existentes, e tomar as decisões finais.

Com vistas a facilitar o trabalho dos analistas de TI para usarem o método de forma assistida e preencherem todas as informações, um pequeno protótipo de software foi desenvolvido.

O resultado está explanado nos Capítulos 5 e 6 da tese.

2.2.4 PASSO 4 - AVALIAÇÃO

Este passo teve por objetivo averiguar o artefato final desenvolvido (i.e. o método de estimação SOA) frente a um conjunto de *medidas de desempenho*. Tais medidas são diretamente relacionadas aos objetivos geral, específicos e proposição de valor do trabalho.

Este passo foi realizado em três etapas: verificação, avaliação e validação.

A etapa de *verificação* procurou testar o método em condições totalmente controladas pelo autor, onde se buscou aplicar o método em

inúmeros casos de uso (concebidos pelo próprio autor, desde simples a muito complexos) e verificar se ele de fato estaria calculando todos os fatores de forma correta. Neste etapa algumas correções foram efetuadas.

A *avaliação* foi uma segunda etapa. Considerando que o método estava preliminarmente correto, ele foi submetido para uso em condições perto do real. Isto ocorreu junto a uma grande empresa do Rio de Janeiro e que tem um também grande e muito sólido departamento de TI, além de muita experiência e maturidade em SOA. A empresa aplicou o método em alguns dos seus Casos de Uso reais. Houve várias interações entre um dos analistas de TI da empresa e este autor, e ao final alguns refinamentos no método foram efetuados.

Adicionalmente, o método foi aplicado a um grupo de alunos de pós-graduação em Automação da UFSC que fizeram a disciplina de SOA assim como avaliado por dois experientes pesquisadores do GSIGMA, grupo de pesquisas coordenado pelo orientador desta tese. Da mesma forma, ao final alguns refinamentos no método foram efetuados.

Todos esses ciclos de melhoramentos (*fluxos de conhecimento*) estão previstos na metodologia *Design Science*.

Essas duas primeiras etapas visaram principalmente averiguar a *corretude* em si do método, ou seja, observando as eventuais faltas, falhas e erros no método.

A terceira e última etapa, *validação*, visou averiguar o cumprimento dos objetivos da tese bem como sua proposição de valor, (descritos no Capítulo 1), de forma mais alargada, junto a um conjunto maior de empresas e de diferentes portes. Complementarmente, já se tinha recebido retornos muito positivos da comunidade científica sobre o método quando daquelas duas publicações (acima mencionadas).

Foram procuradas e selecionadas empresas de software e aplicada a técnica *Expert Panel* junto a um conjunto de profissionais delas. O uso desta técnica implicou na elaboração de um questionário estruturado que, em suma, visou se fazer refletir nas perguntas os vários aspectos desejados de serem avaliados no método (as *medidas de desempenho*) considerando os seus objetivos e proposição de valor.

Finalmente, analisaram-se os resultados obtidos com os questionários aplicados.

O resultado desta etapa está explanado no Capítulo 7 desta tese.

2.2.5 PASSO 5 - CONCLUSÕES

Esta última etapa do método *Design Science* visou sintetizar e fazer uma avaliação global dos resultados dos questionários juntamente com várias observações e reflexões deste autor sobre a pesquisa em si e sobre o método propriamente dito.

Por fim, são descritas algumas sugestões de trabalhos futuros, de continuação e melhorias no método.

O resultado desta etapa está explanado no Capítulo 8 desta tese.

3 REVISÃO DA LITERATURA

Este capítulo apresenta uma revisão bibliográfica das principais aspectos ligados às fundamentações teóricas que norteiam essa tese, a saber: SOA, Engenharia de Software e Técnicas Multicritérios. Cada um desses assuntos é coberto em uma seção.

3.1 ARQUITETURA ORIENTADA A SERVIÇOS (SOA)

São inúmeras as definições de SOA presentes na literatura. Algumas focam aspectos mais ligados à engenharia de software, outras a paradigmas de sistemas, e outras mais a questões de tecnologias.

Josuttis (2008), por exemplo, define SOA como uma abordagem para a realização e manutenção de processos corporativos existentes em grandes sistemas distribuídos, ajudando-os a se manterem flexíveis e escaláveis enquanto crescem e solucionam lacunas entre negócios e TI.

Dias, Oliveira e Meira (2013) resume SOA como uma forma de projetar uma arquitetura baseada na composição de serviços reutilizáveis e interoperáveis capazes de se comunicarem de forma transparente entre si, havendo semelhança ou não entre eles.

Crawford *et al*. (2005) e Kumar, Dakshinamoorthy e Krishnan (2007) afirmam que a orientação de serviços para a construção da arquitetura de sistemas de uma organização é um processo para alcançar maior agilidade no ambiente de negócios cada vez mais competitivo.

Papazoglou (2012) aborda SOA sob uma visão mais macro, a partir da Computação Orientada a Serviços, paradigma este de computação que considera como "serviços" os elementos fundamentais para desenvolver soluções e aplicações distribuídas de um lado, e provedores de serviços de outro lado. Em SOA, todo um sistema é composto por módulos de software independentes, auto contidos e distribuídos - chamados de *serviços de software* ou apenas *serviços* - que formam em conjunto, uma unidade lógica única para criar mais rapidamente diferenciados produtos e processos. Um dos objetivos-chave do SOA é facilitar o alinhamento da TI com o nível dos (processos de) negócios (PAPAZOGLOU, 2012).

Destacam-se ainda nas definições e propostas sobre SOA o foco na padronização de interfaces, reutilização de código e artefatos desenvolvidos (MEZO; CHAPARRO; HERAS, 2008) (STAL, 2002) (MARKS; BELL, 2006).

Um serviço de software pode ser definido como um geralmente pequeno módulo de software independente, desacoplado de qualquer outro sistema, que executa um certo conjunto de funcionalidades (normalmente poucas e bem específicas) e que é integrável a outros serviços via suas interfaces para formar uma dada aplicação SOA final. Há uma série de tecnologias para a implementação de sistemas baseados em serviços, sendo os *web services* uma das mais usadas (PAPAZOGLOU, 2012).

Um sistema SOA é um sistema distribuído composto por vários tipos de serviços, componentes e elementos de infraestrutura que no todo tornam o sistema efetivamente executável. Assim sendo, num projeto SOA a análise geral de viabilidade acaba por ter que considerar todo esse cenário, e não apenas o da eventual disponibilidade em si de um dado serviço ao nível de aplicação/negócio (O'BRIEN, 2009).

Como principais benefícios da arquitetura SOA podem-se apontar: a redução da dependência tecnológica e a simplificação do processo de desenvolvimento de software, o aumento do reuso nas soluções pelas corporações, e uma mais fácil integração de aplicações com outros sistemas (PAPAZOGLOU, 2012). Ainda, mencionam-se a flexibilidade e a agilidade na adaptação das aplicações às constantes mudanças tecnológicas e a novos processos de negócios (CRAWFORD *et al*., 2005) (KUMAR; DAKSHINAMOORTHY; KRISHNAN, 2007) (CIGANEK *et al.*, 2005); a mitigação de dificuldades tecnológicas e de integração oriundas da obsolescência dos sistemas legados (MEZO; CHAPARRO; HERAS, 2008) ou de problemas de utilização de ou adaptação a outras tecnologias de integração (KUMAR; DAKSHINAMOORTHY; KRISHNAN, 2007); a maior facilidade de substituição de componentes e pacotes de software devido a menor dependência dos serviços em relação aos sistemas que os suportam (PUSCHMANN; ALT, 2001) (SERMAN, 2007); o reuso (ERL 2009) (MARKS; BELL, 2006); uma maior flexibilidade e escalabilidade da arquitetura, na sua manutenção, e no gerenciamento de falhas (SORDI; MARINHO; NAGY, 2006).

SOA possui quatro elementos principais, que denotam o que se convenciona chamar de "arquitetura SOA" (Figura 3) (PAPAZOGLOU, 2012):

- Provedor de serviço: são empresas ou responsáveis gerais por implementar os serviços de software, fornecer suas descrições, publicá-los em um registro de serviços, prestar suporte técnico e de negócio, e que tem direitos variados sobre eles.

- Consumidor do serviço: são atores que requerem determinados serviços para execução de suas regras de negócio, podendo ser um serviço ou um outro tipo de software que requeira um serviço.
- Registro de serviço: consiste em um repositório no qual provedores de serviços registram os serviços para que estes possam ser localizados pelos consumidores. Abrigam informações sobre as funções oferecidas, os requisitos para a utilização, e orientações de como realizar a interação com o provedor de serviço.
- Contrato: é a especificação completa de um serviço. É ele que estabelece a relação consumidor: provedor. Os consumidores não têm acesso à implementação, mas somente aos detalhes do serviço exposto no contrato. Este contrato é representado pelo SLA (*Service Level Agreement*), que define níveis de qualidade a nível não-funcional que devem ser suportados pelo provedor, tais como níveis de desempenho e disponibilidade.

Provedor e consumidor são na verdade papéis lógicos, de forma que um mesmo ator pode exercer ambos, dependendo da lógica das regras de negócio.

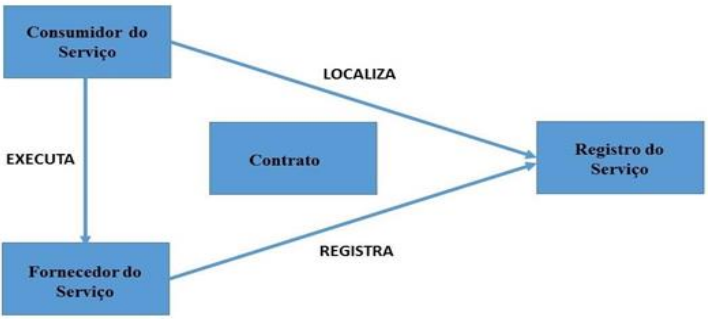


Figura 3: Funcionamento da SOA
Fonte: Adaptado de Marzullo (2009).

3.1.1 Elementos SOA

Conforme Krafzig, Banke e Slama (2004), SOA envolve diversos elementos inter-relacionados, podendo ser representados por uma hierarquia (Figura 4):

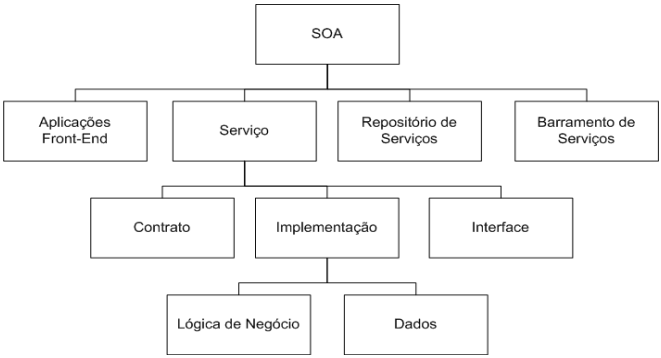


Figura 4: Hierarquia dos elementos da SOA
Fonte: Krafzig; Banke; Slama (2004).

- Aplicações *front-end*: são aplicações que possuem algum tipo de interface gráfica com o usuário, como páginas web e portais;
- Serviço: mecanismo para habilitar o acesso a uma ou mais funcionalidades fornecidas por um provedor de serviço;
- Repositório de serviços: área de dados que oferece informações sobre os serviços ofertados;
- Barramento de serviços: *middleware* que possibilita a comunicação entre vários componentes de arquitetura SOA;
- Contrato: estrutura que descreve o serviço de maneira independente de tecnologia, definindo funcionalidade, modo de utilização, restrições e nível de qualidade oferecido;
- Implementação: elemento que efetivamente executa a funcionalidade definida pelo serviço;
- Interface: parte do serviço exposta pelo provedor para os consumidores, abstraindo os detalhes da implementação;
- Lógica de negócio: regras de negócio envolvidas na funcionalidade dos serviços;
- Dados: são informações que podem ser fornecidas pelos consumidores ou serem provenientes de aplicações.

Em outra perspectiva e de forma complementar, Marks, Bell (2006) identificam os elementos que formam uma arquitetura de SOA (Figura 5):

Figura 5: Arquitetura da SOA
Fonte: Marks; Bell (2006).

- Estratégia de SOA: define o planejamento e as decisões relativas à arquitetura orientada a serviços na organização;
- Governança de SOA: é composta por processos, papéis, responsabilidades, padrões e políticas que definem a visão conceitual da arquitetura;
- Métricas: São definidas para verificar os resultados obtidos pela arquitetura SOA, tanto em termos de desempenho de processos de negócio quanto de retorno sobre investimento.
- Comportamento e Cultura: conjunto de hábitos da organização estabelecidos através de normas, valores e atitudes.
- Modelo de Arquitetura: visão conceitual da arquitetura necessária para implementação da arquitetura SOA.
- Tecnologia de Implementação: torna realidade a visão conceitual da arquitetura SOA, possibilitando a construção e operação dos serviços.
- Serviços: representa os artefatos de execução dos processos.

3.1.2 Serviços de Software

Segundo Papazoglou (2012), um serviço é um elemento computacional auto-descritivo e independente de plataforma que possibilita a composição de aplicações distribuídas de maneira mais rápida e com menor custo. Os serviços podem executar funções dos mais diversos níveis de granularidade, representando desde uma simples resposta a uma requisição até um processo de negócio inteiro.

Erl (2009) afirma que os serviços dão suporte ao alcance dos objetivos estratégicos associados à computação orientada a serviços. Para ele, um único serviço pode fornecer uma coleção de capacidades. Tais funcionalidades são agrupadas e se relacionam a um contexto funcional estabelecido pelo serviço.

Segundo Sampaio (2006), um serviço é um componente que atende a uma função de negócio específica para seus clientes, recebendo requisições e as respondendo ocultando todo o detalhamento do seu processamento.

Para Marks e Bell (2006) um serviço de software é visto como um componente de software formado por partes bem definidas e que tem por objetivo solucionar problemas das organizações que envolvam TI.

Um serviço deve executar unidades completas de trabalho, e não devem depender do estado de outros componentes. Desta forma, verifica-se a importância de serem *stateless*, ou seja sem armazenamento de estado de conversação (SAMPAIO, 2006). Isto aumenta significativamente sua reutilização. Para reter esta independência, os serviços encapsulam a lógica dentro de um contexto distinto. Este contexto pode ser específico a uma atividade de negócios, a uma entidade completa de negócios, ou algum outro agrupamento lógico.

Conforme Rosen *et al.* (2008), um serviço é constituído das seguintes partes (Figura 6):

- Políticas e Contrato: fornece informações específicas relacionadas às funcionalidades e como ele deve ser utilizado;
- Interface: a funcionalidade do serviço é exposta através da interface, permitindo que o cliente visualize as possíveis funções que podem ser desempenhadas por aquele serviço;
- Implementação: a implementação do serviço consiste de um ou mais artefatos, que pode ser representado por classes ou trechos de código, lógica de negócios.

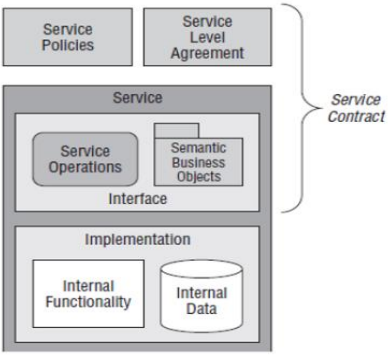


Figura 6: Arquitetura do serviço
Fonte: Rosen *et al.* (2008).

Um serviço pode fazer parte de outro serviço de maior abrangência ou executar uma tarefa distinta de forma individual, como ilustrado na Figura 7. Esta flexibilidade facilita o reuso dos serviços.

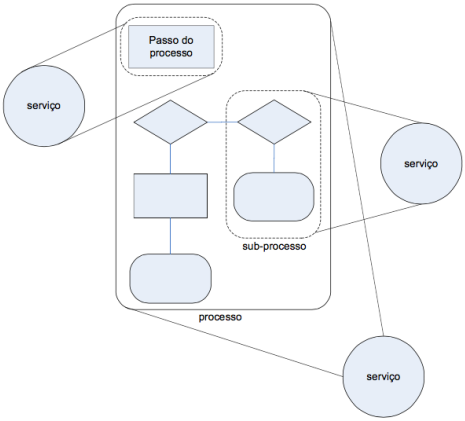


Figura 7: Lógica encapsulada nos Serviços
Fonte: Adaptado de Erl (2009).

*3.1.2.1 **Classificação dos Serviços***

Os serviços podem ser classificados dependendo do modelo de lógica, do potencial de reuso e de como ele se relaciona com os domínios existentes dentro do ambiente empresarial (ERL, 2009). Existem várias classificações na literatura, mas três categorias são comuns a elas (Figura 8):

- Serviços Utilitários: são conhecidos como serviços de aplicação, serviços de infraestrutura, ou serviços de tecnologia. São serviços que possuem lógica independente de negócio, ou seja, são responsáveis por realizar atividades que podem ser aplicadas a diversos processos de negócio e assim podem ser reutilizados com maior facilidade. Exemplos mais comuns são serviços de log, notificação (envio de e-mail, SMS), tratamento de erros, entre outros. Por sua natureza mais técnica, estas operações possuem alto potencial de reuso.
- Serviços de Entidade: serviços relacionados às entidades envolvidas no negócio, mas possuem características mais genéricas, não sendo específicos de um processo de negócio. Por exemplo, o serviço de Análise de Crédito não é específico ao Processo de Financiamento, pois pode ser usado no Cadastro de um Prospecto e na Revisão de Limite de Crédito.
- Serviços de Tarefa: responsáveis pelos processos específicos de cada negócio, reduzindo dessa forma seu potencial de reúso. Normalmente é implementado através de uma *composição* ou *orquestração* de serviços.

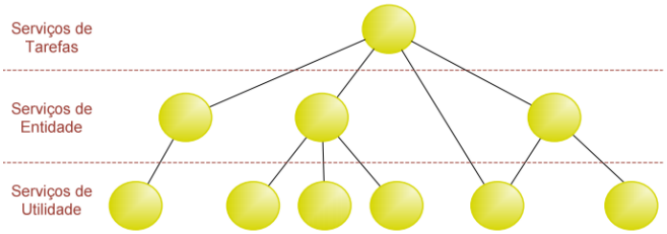


Figura 8: Camada de abstração dos Serviços
Fonte: adaptada de Erl (2009).

3.1.3 Governança SOA

Devido a existência de grande número de serviços disponíveis numa organização, é comum haver diversas equipes de projeto desenvolvendo-os e utilizando-os. Com isto, um serviço deixa de ser responsabilidade de uma única equipe de projeto e deixa também de pertencer a um único processo de negócio.

SOA demanda a existência de uma estrutura de *governança* na organização de modo a estabelecer papéis, responsabilidades e processos para organizar os serviços desenvolvidos (ERL, 2009). Governança SOA é definida como o conjunto de processos usados para prever e controlar todas ações ligadas ao ciclo de vida de um serviço ou aplicação baseada em serviço de acordo com as práticas, princípios e regulamentações vigentes (ERL, 2009). As ações administrativas envolvidas na governança SOA, como tomadas de decisão e o acompanhamento dos projetos, são consideradas essenciais para se ter sucesso num projeto SOA (WEBMETHODS, 2006).

Josuttis (2008) aponta os seguintes aspectos na governança SOA:

- Arquitetura de referência: É necessária uma referência que demonstre decisões arquiteturais, incluindo a tecnologia preferida, protocolo de troca de mensagens, etc.
- Papéis e responsabilidades: Precisam ser definidos para quem está envolvido com as questões arquiteturais de análises técnicas e de tomadas de decisão.
- Políticas, padrões e formatos: Decisões arquiteturais, artefatos e tecnologias levam a políticas e requisitos, os quais podem se desdobrar no uso de padrões ou formatos proprietários.
- Processos e ciclos de vida: Precisa definir processos e ciclos de vida para soluções, serviços e outros.
- Documentação: Ajuda a promover, manter e consolidar questões não técnicas, como processos, responsabilidades e políticas.
- Gerenciamento de serviços: Tem o objetivo de gerenciar os serviços e seus contratos.
- Monitoramento: Verifica regras, políticas, contratos/SLAs.
- Gerenciamento de mudança e configuração: As ferramentas de gerenciamento de configuração ajudam a gerenciar software, artefatos e documentação SOA.

Schepers, Iacob e Van Eck (2008) consideram que, em geral, um modelo de ciclo de vida de governança SOA contém as seguintes etapas:

- Definição da estratégia SOA: análise da organização a fim de determinar as contribuições da arquitetura SOA. Além disto, esta etapa avalia as necessidades da arquitetura SOA, definindo o escopo do modelo de governança.
- Alinhamento organizacional: tradução da estratégia para o contexto organizacional, onde a atenção é dada para a propriedade e custos. Nesta etapa são tratadas a mudança de cultura através da criação de um centro de excelência ou escritório dedicado para promover SOA na organização.
- Gestão de portfólio de serviços: garantir qualidade de serviço, controlando a qualidade do projeto no ciclo de vida do serviço, onde os princípios de projeto são criados e usados.
- Gestão do ciclo de vida de serviços: define a catalogação adequada de serviços a fim de explicitar os relacionamentos de serviços com seus clientes, permitindo a análise dos impactos das mudanças dos serviços. Além disto, trata da importância de procedimentos corretos para gerência de mudanças dos serviços.
- Gestão de políticas SOA: Verifica a conformidade dos serviços com os normas internas e a legislação.
- Gestão de níveis de serviço: cuidar da qualidade operacional dos serviços.

Todavia, SOA não é adequado para quaisquer casos. Por exemplo, ele não é recomendado em certos casos ou pouco agrega em relação aos objetivos para os quais SOA foi idealizado (KRAFZIG; BANKE; SLAMA, 2004):

- aplicações autossuficientes (*stand-alone*) que para seu funcionamento não necessitam de softwares auxiliares ou fazer uso de outros serviços;
- Para aplicações que requeiram interfaces gráficas complexas, onde o volume e tráfego de dados seja muito grande;
- Para aplicações de tempo real, em função das limitações do protocolo *http*;
- Para aplicações cujo fluxo de execução é intrinsecamente não estruturado e dinâmico, por exemplo, um processo de criação de produtos.

Outros aspectos, apesar de já usados em outras abordagens, passam a ser mais críticos em SOA. Por exemplo, o de padrões abertos para protocolos de comunicação e documentação (CURBERA *et al*., 2004) (DIETRICH; KIRN; SUGUMARAN, 2007), e o de uso de um meio de comunicação universal, como a Internet, para dar suporte à tecnologias abertas, como no caso dos *web services* (CURBERA *et al*., 2004) (STAL, 2002).

Há diversas semelhanças ao se comparar algumas das propriedades de SOA com o desenvolvido em outras abordagens. Por exemplo, a reutilização de código ou de componentes já é sugerida como boa prática há muitos anos (PRESSMAN, 2016).

3.1.4 Desenvolvimento de Software baseado em SOA

Qualquer software, seja um "tradicional", seja um orientado a serviços, tem um ciclo de vida. No entanto, existem algumas diferenças quando se trata de SOA (BANO; IKRAM, 2010). Neste contexto surge a Engenharia de Sistemas Orientada a Serviços (SOSE), que "adapta" a engenharia de software (de sistemas tradicionais) com a visão SOA (GU; LAGO, 2009). Conforme Tsai et al. (2007), SOSE visa propor abordagens sistemáticas para a concepção, desenvolvimento e manutenção de sistemas orientados a serviços. Além dos requisitos usuais, como custo e qualidade de software, SOSE também considera outros requisitos, como os focados em composição e integração do sistema (ZHOU; NIEMELA, 2005).

A Figura 9 procura ilustrar vários dos modelos de ciclo de vida de orientação à serviços propostos por diversos autores na literatura em relação às etapas clássicas de engenharia de software.

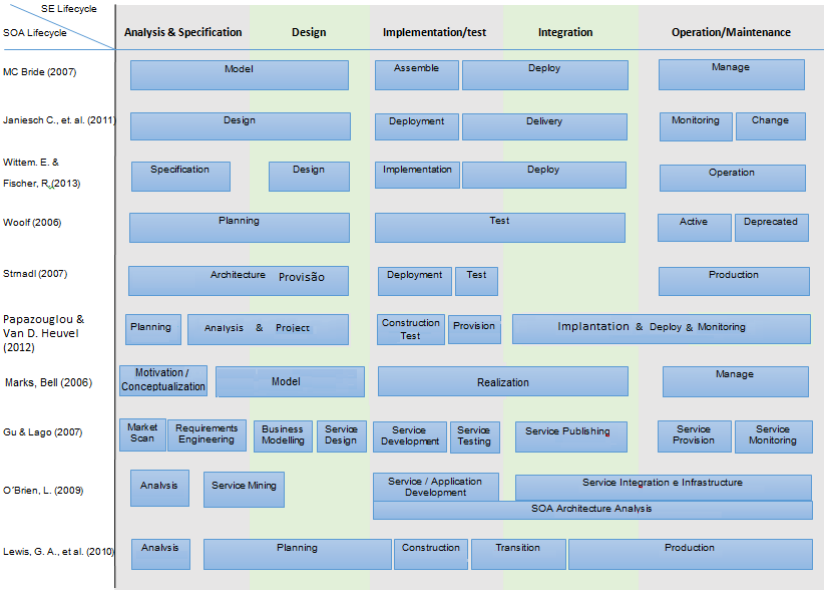


Figura 9: Desenvolvimento SOA x Desenvolvimento tradicional
Fonte: Autor (2015).

Na parte superior da figura 9 estão relacionadas as etapas do ciclo da vida da Engenharia de Software. No lado esquerdo da figura encontra-se as obras dos autores pesquisados. Como se pode observar, cada autor tem sua visão particular de ciclo de vida SOA, ora mais abrangente, ora mais detalhada em termos de atividaes, seus escopos e até mesmo semântica dos termos.

Como o método proposto nesta tese de doutorado foca na etapa de *Análise* no ciclo de vida, serão analisadas as suas principais abordagens.

*3.1.4.1 **Etapa de Análise por Papazoglou e Van Den Heuvel***

Papazoglou e Van Den Heuvel (2006) descrevem as etapas do ciclo de vida dos serviços consistindo de um processo cíclico e iterativo, que envolve diferentes atividades e que trabalha com modelagem de processos em análise, projeto, implementação, execução e monitoramento (Figura 9). Eles enfatizam que a modelagem de serviços deve tratar, no mínimo, das seguintes partes fundamentais de serviços:

- Estrutura: definição do tipo do serviço, mensagem, interfaces e operadores;
- Comportamento: documentação dos efeitos do serviço e suas operações, e da semântica das mensagens de entrada e saída;
- Política: políticas e restrições do serviço e as considerações sobre os serviços em relação às partes que interagem com ele.

Na etapa de Análise são definidos os requisitos da aplicação, que direcionam o desenvolvimento dos processos de negócio. A etapa de análise é subdivida nas seguintes atividades (Figura 10):

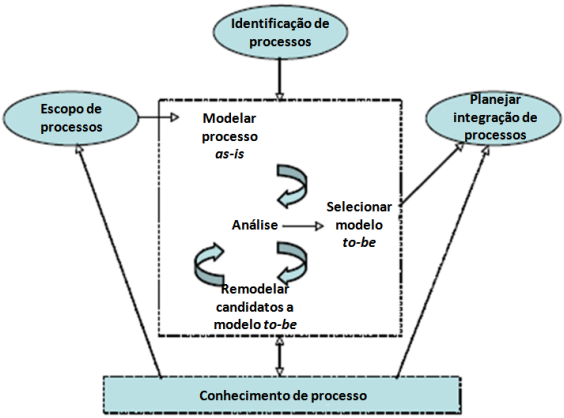


Figura 10: Identificação de processo e escopo de processo
Fonte: Papazoglou (2006).

- Identificação de processos: nesta atividade é analisado o funcionamento interno de organização em termos de processos de negócio e identificados quais são as atividades deste processo. Após a análise, é elaborado o modelo *as-is* dos processos de negócio. Em seguida, este modelo é analisado com vistas a possíveis melhorias.
- Escopo de processos: O escopo de um processo de negócio inclui a definição de início e término do processo.
- Planejar integração de processos: nesta atividade é feita planejamento da integração de processos identificados.

- Conhecimento de processo: as informações sobre os processos que são utilizados para apoiar na análise e planejamento da integração de processos.

*3.1.4.2 **Etapa de Análise proposta por Marks e Bell***

O ciclo de vida proposto por Marks e Bell (2006) define as atividades, que vai desde a identificação, análise e projeto de serviços até sua implementação e gerenciamento (Figura 9). As fases do ciclo de vida dividem-se em dois grandes domínios: o domínio do problema e o domínio da solução. O domínio do problema é formado pelas atividades do ciclo de vida que resultam na identificação e análise dos serviços da organização. Nestas atividades, o resultado será um conjunto de serviços candidatos. Já o domínio da solução envolve o projeto dos serviços candidatos, a implementação e a integração, resultando nos serviços de solução construídos (Figura 11).

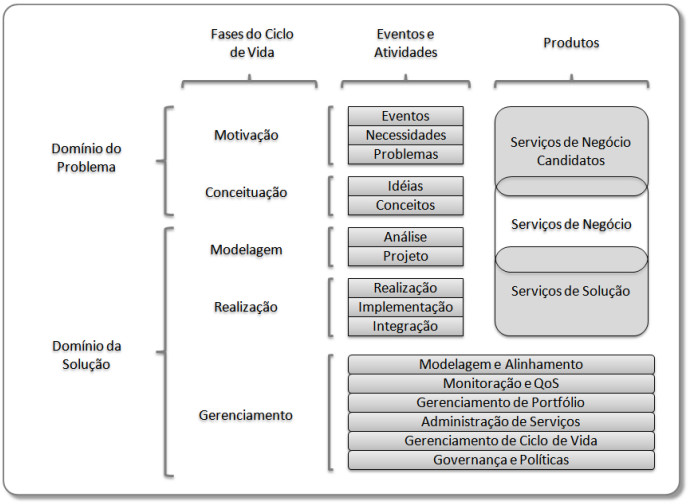


Figura 11: Ciclo de vida de serviços de Marks e Bell
Fonte: Marks e Bell (2006).

- Motivação: Consiste nas forças motivadoras que terão o objetivo de tratar eventos e resolver necessidades da organização. São divididas em 3 tipos:
  - Os *eventos* podem ser tendências de mercado, ameaças competitivas, regulamentações ou ocorrências tecnológicas

que criam oportunidades e demandam algum tipo de resposta por parte da organização.
  - As *necessidades de negócio* são questões relativas ao modelo de negócio da organização, por exemplo “baixa satisfação dos clientes”, “aumento na competitividade”. Já as necessidades de TIC são focadas em tecnologia, como custos excessivos de integração, dificuldade de suportar mudanças no negócio.
  - Os *problemas* devem ser identificados e estudados de modo a determinar sua severidade, consequências e possíveis causas.
- Conceituação**:** são levantadas ideias que consistem em alternativas de solução para as necessidades e problemas identificados na etapa anterior, e são definidos conceitos que serão posteriormente utilizadas na modelagem dos serviços.
- Modelagem**:** visa identificar, projetar e implementar os serviços identificados na fase anterior. Na atividade de *análise* são realizadas operações lógicas de modelagem sobre os serviços candidatos. A atividade de *projeto* define o escopo dos serviços que serão criados.

Na atividade de análise utilizam-se modos *top-down* e *bottom-up*, em ciclos iterativos. Isto porque, de um lado o modo *bottom-up* pode deixar os serviços acoplados com seu ambiente tecnológico, dificultando o seu reuso. Por outro lado, o modo *top-down* pode facilitar o reuso, mas os serviços podem ser difíceis de implementar. Assim, o modo *top-down* é usado para identificar os serviços necessários, descobri-los e analisá-los, enquanto que o projeto em si é realizado pelo modo *bottom-up* (Figura 12).

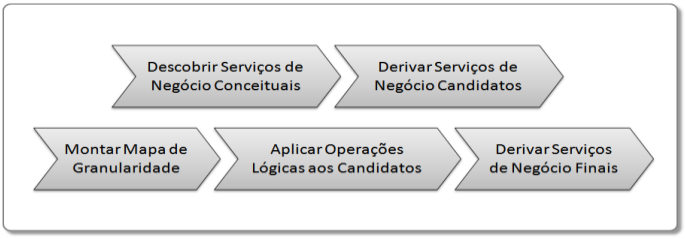


Figura 12: Processo de identificação e análise de serviços
Fonte: Marks e Bell (2006).

- Realização: é composta por atividades de *realização*, *implementação* e *integração*. Na atividade *realização* as especificações dos serviços são traduzidas em especificações técnicas. Na atividade de *implementação* ocorre o desenvolvimento e os testes dos componentes, e na atividade *integração* são realizadas as interações com aplicações e serviços já existentes na arquitetura.
- Gerenciamento: são realizadas atividades para garantir que os serviços operem adequadamente.

Conforme Marks e Bell (2006), a melhor abordagem para a identificação de serviços é iniciar com análise dos requisitos de negócio, mas estes requisitos não são a única fonte para tal, devendo ainda incluir como fontes a documentação do domínio do problema, a documentação de modelagem de processos de negócios, a documentação de estratégia de negócios e tecnologia, e os requisitos técnicos e especificações.

3.1.4.3 ***Etapa de Análise proposta por Sommerville***

Sommerville (2011) define três estágios lógicos (Figura 13):

- Identificação de serviço candidato;
- Projeto de serviço: Projetar a lógica e as interfaces;
- Implementação e implantação de serviço.

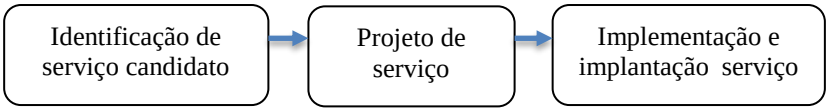


Figura 13: Processo de identificação e análise de serviços
Fonte: Sommerville (2011).

A atividade identificação de serviços candidatos é a primeira atividade da etapa de análise no ciclo da vida de SOA. Ela recebe como entrada artefatos, como o modelo de processo de negócio e a especificação de requisitos. Quando o artefato da entrada for um modelo de processos de negócio, a identificação do serviço é baseada em atividades que podem ser executadas por serviços. Já quando o artefato de entrada for um Caso de Uso, regra de negócio ou requisitos não funcionais, a identificação é baseada nas especificações dos artefatos. A saída desta atividade é uma lista de operações ou atividades candidatas que podem se tornar ativos de software.

3.1.4.4 ***Etapa de Análise proposta por Bieberstein***

O *Rational Unified Process (RUP)* é um processo de engenharia de software desenvolvido pela IBM. Trata-se de um método de gerenciamento de atividades e papéis em uma organização de desenvolvimento de software. Para a utilização do RUP em projetos baseado SOA, a IBM disponibiliza um "*plugin*". O *RUP plugin for Service-Oriented Modeling and Architecture* (SOMA) (ZIMMERMANN, 2005) contém e define as atividades, papéis e artefatos relacionados ao desenvolvimento orientado a serviços e que devem ser incorporados ao processo da organização.

O modelo do RUP pode ser descrito em duas dimensões: o conteúdo do método e o processo. O conteúdo do método é constituído por atividades, papéis e artefatos. O processo representa o fluxo de trabalho ao longo do tempo, com suas etapas e iterações (SHUJA; KREBS, 2007) (Figura 14).

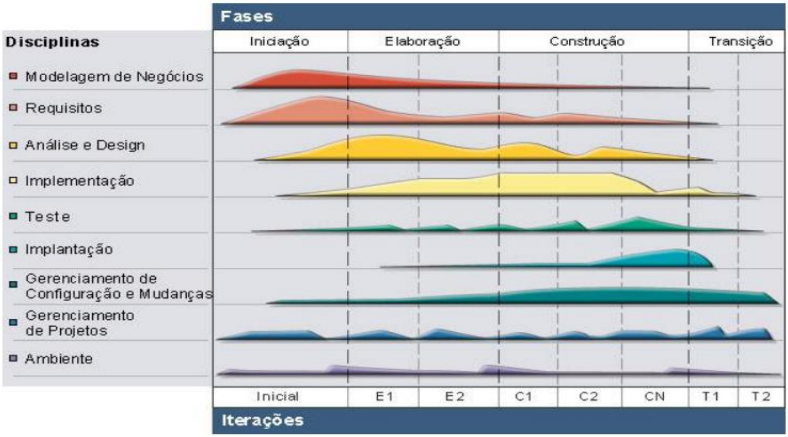


Figura 14: Arquitetura do Rational Unified Process
Fonte: Shuja e Krebs (2007).

De acordo com o RUP, o ciclo de vida de desenvolvimento de software é dividido em quatro etapas, sendo que ao final de cada uma delas é desenvolvida uma nova geração do produto:

Na *Iniciação* é feita a elicitação e a análise dos requisitos, junto com a definição de custo e tempo do desenvolvimento. A *Elaboração* parte dos requisitos do domínio do problema para propor uma arquitetura para a solução levando em consideração o escopo. Na

Construção são produzidos os componentes de software junto com a documentação. A *Transição* lida com a implantação da solução no ambiente e inclui treinamento para os usuários finais.

Os insumos de entrada para o SOMA são os modelos de processos de negócios *to-be* a serem posteriormente tomados como base para os serviços a serem construídos e sua orquestração.

As técnicas usadas para identificação de serviços no processo descrito por Bieberstein *et al.* (2005) são (Figura 15):

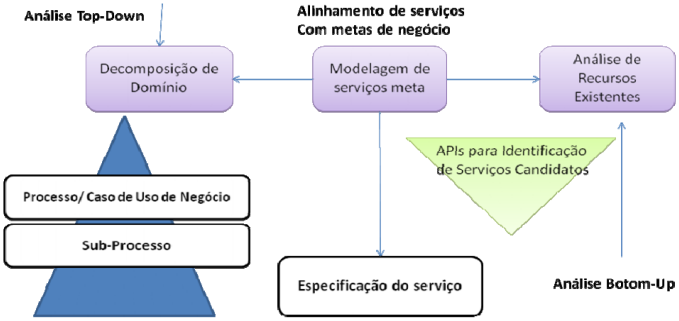


Figura 15: Três técnicas diferentes para identificação de serviços
Fonte: Bieberstein *et al.* (2005).

- Decomposição de domínio (*Top-down*): Esta técnica analisa as áreas funcionais e os subsistemas que fazem parte de um domínio e depois analisa também processos de negócios e casos de uso de negócios. É semelhante à técnica de Papazouglou e Van Den Heuvel (2006).
- Análise dos ativos já existentes (*Bottom-up*): A partir dos recursos existentes na organização podem ser identificados componentes que têm potencial de reuso e estes serão considerados como candidatos a se tornar serviços / ativos.
- Modelagem dos serviços-meta: Caso as lógicas dos serviços identificados pelas técnicas *top-down* e *bottom-up* não incluam todas as necessidades de negócio, as lacunas de necessidades constituem uma oportunidade para a definição de serviços-meta. O termo *serviço-meta* refere-se a um serviço voltado para atender a uma meta de negócio.

As abordagens para identificação de serviços do SOMA são similares ao utilizado no processo de Papazoglou e Van Den Heuvel

(2006) e ambos trabalham com foco em processos de negócios. Os passos para identificação do serviço são detalhados, mas não há enfoque para a garantia de princípios como o de reutilização.

3.1.5 Discussão sobre Insumos de Entrada e Atividades de Análise

O resultado da atividade de análise, fruto de diferentes visões/ciclos de vida, como as acima descritas, é uma lista de serviços candidatos que serão avaliados para se extrair os serviços mais relevantes de serem implementados.

A Figura 16 procura ilustrar uma visão "unificada" das atividades do processo identificação de serviço candidatos, baseado nos trabalhos citados acima. Neste sentido, a atividade de identificação de serviço pode ter insumos como BPM, estratégias, requisitos e domínios. A saída da atividade é uma lista de serviços candidatos que serão avaliados na atividade "Análise *GAP*" para verificar a possibilidade de reaproveitar ativos e legados. Após esta avaliação, a saída será a lista dos serviços a serem analisados pela atividade "Análise de realização", para verificar a viabilidade da suas implementações.

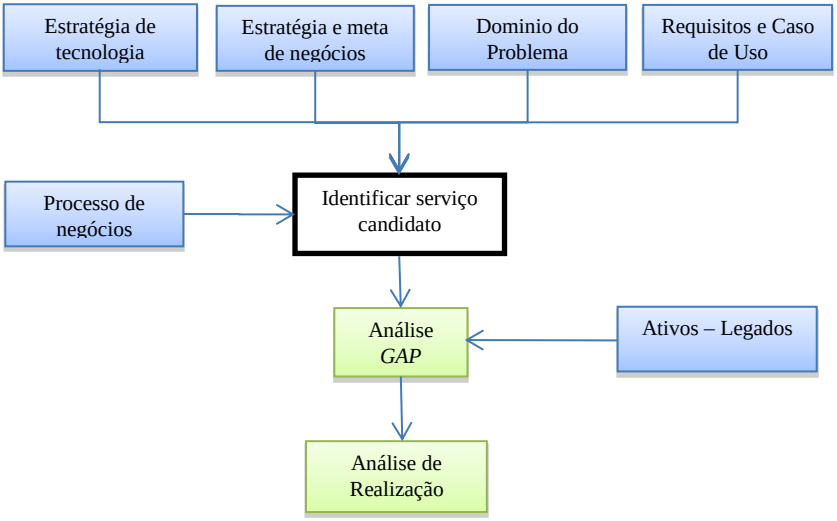


Figura 16: Visão consolidada do processo de identificação de serviços
Fonte: Autor.

A Tabela 1 resume os estudos realizados focado em atividade análise e apresenta uma consolidação dos insumos para a atividade de identificação de serviços e também os fatores que cada autor identificou como necessários para a análise de serviços candidatos.

Tabela 1: Atividades da identificação dos serviços candidatos

Autores	**Input para Identificação de Serviços**					**Fatores**											
	BPM	Requisitos Funcionais	Requisitos Não Funcionais	Aspectos Negócio	Legados	Experiência Profissional	Conformidade com os objetivo de negócio	Condição Ambiental	Custo	Esforço	Tempo de desenvolvimento	Produtividade	Tamanho	Reuso	Complexidade ambiental	Complexidade integração	Complexidade serviço	Tamanho de equipe
Papazouglou e Van Den Heuvel	x			x	x				x				x	x	x	x	x	
Marks & Bell	x	x	x	x	x					x			x	x	x		x	
Bieberstein	x	x	x	x					x		x			x				
Sommerville	x	x	x															
Método proposto	***x***	***x***	***x***	***x***	***x***	***x***	***x***	***x***	***x***	***x***	***x***	***x***	***x***	***x***	***x***	***x***	***x***	***x***

Fonte: Autor.

Como não existe uma padronização nos diversas modelos abordados por esses autores, ainda existe uma discussão sobre os prós e contras de cada um e assim de desenvolvimento de novas técnicas (RAMOLLARI; DRANIDIS; SIMONS, 2007).

Os modelos estudados possuem algumas similaridades com relação aos recursos utilizados na identificação de serviços. Pela Tabela 1, com a comparação dos principais métodos de identificação de serviços, notou-se que o método de Marks & Bell parece ser o mais completo, pois utiliza maior quantidade de insumos de entrada. Os outros modelos são baseados principalmente na modelagem de

processos de negócio e trazem um enfoque basicamente na questão de reutilização dos recursos existentes da infraestrutura de TI.

Conclui-se que os métodos possuem certa similaridade nas abordagens e que todos têm a preocupação (embora por vezes muito genérica) em alinhar os serviços com o negócio. Numa visão geral, as limitações deles se concentram na falta de detalhamento dos passos para se extrair os serviços, como em Biebertein (2008) e Papazoglou e Van Den Heuvel (2006). Contudo, o modelo abordado pelo Marks e Bell (2006) é bem extenso e possui muitas atividades para realizar a definição completa do serviço.

3.1.6 Estratégia SOA e Necessidades do Negócio

Conforme levantamento da literatura, a decisão de desenvolvimento como serviço numa estratégia SOA e que vise a geração de benefícios para a organização passa por avaliar aspectos do negócio. As seções a seguir sumarizam algumas das abordagens encontradas na literatura consideradas como as mais completas para se considerar o aspecto de negócio.

*3.1.6.1 **Lewis***

Lewis, Smith e Kontogiannis (2010) relatam que existe uma forte relação entre a estratégia do negócio e a estratégia SOA. Neste trabalho, a identificação dessa relação inclui:

- Identificação do problema SOA e sua solução;
- Elaborar ciclo de vida para suportar estratégia SOA;
- Uma taxonomia única apresentando dimensões alinhada com aspectos de Negócio, Engenharia, Operacional e outras dimensões transversais (ex. governação, segurança, gestão de riscos, aspectos sociais e legais, e treinamentos.) envolvidas no ciclo da vida da arquitetura SOA.

Os autores apontam que no desenvolvimento SOA existe um conjunto distintos de preocupações para diferentes tipos de atores:

- Engenheiros de Software: as preocupações são referentes aos requisitos funcionais, componentes, técnicas da integração, mensagens, ferramentas e ambiente de desenvolvimento;

- Analistas de Negócios: as preocupações são referentes à implementação de estratégias de negócios, conduzindo a novos e mais ágeis processos em prestação de serviços;
- Usuários: as preocupações são referentes à consideração dos níveis de qualidade geral esperadas para a aplicação ou serviço a ser desenvolvido e os respectivos SLAs.

Figura 17 ilustra as etapas do modelo proposto referente ao problema SOA e solução.

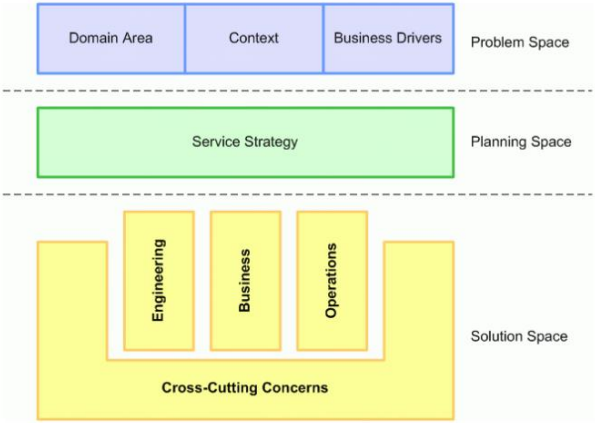


Figura 17: Visão geral do espaço do problema SOA e espaço solução
Fonte: Lewis, Smith e Kontogiannis (2010).

Problem Space: Identifica as características internas e externas da organização e a definição das necessidades do negócio que a adoção de SOA deve contribuir para atender:

- *Domain Area*: define normas regulatórias e padrões de mercado;
- *Context*: é o ambiente em que a organização opera e em que os serviços serão implantados;
- *Business Drives*: aponta necessidades do negócio que orientam a adoção de SOA como solução e seus objetivos-fim.

Planning Space: Define o planejamento para desenvolver soluções que atendam às necessidades do negócio. Recebe como entrada dados do domínio do problema e sua saída é um conjunto de planos de execução

SOA (Figura 18). Na figura os quadrados arredondados representam atividades e os retângulos representam artefatos.

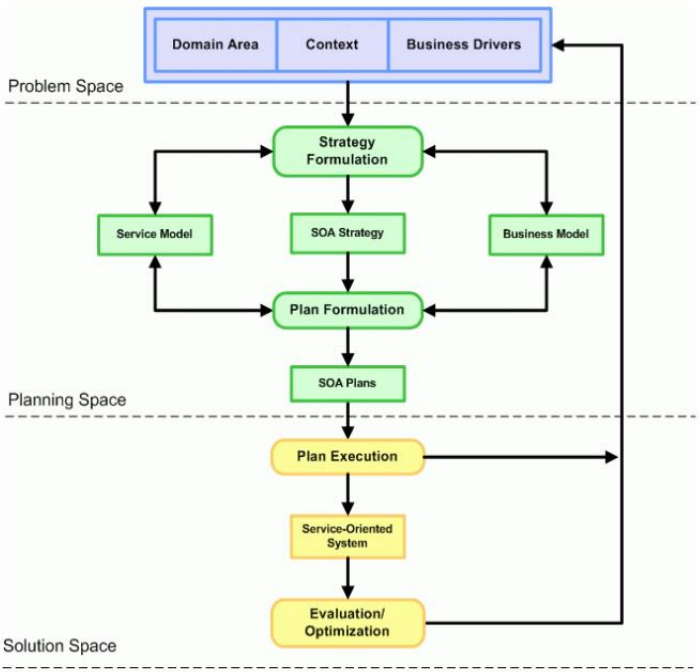


Figura 18: Diagrama do modelo sugerido
Fonte: Lewis, Smith e Kontogiannis (2010).

Solution Space: representa a execução dos planos para produzir um sistema orientado a serviços, incluindo um conjunto de serviços que ofereçam funcionalidades reutilizáveis de negócios.

3.1.6.2 ***Börner & Goeken***

Börner & Goeken (2009) propuseram uma lista de vários critérios e perguntas sobre como considerar aspectos de negócio num projeto SOA (Tabela 2).

Tabela 2: Aspectos de negócio

Aspectos de Negócio	Descrição
Aspectos Estratégicos	Organização tem aspectos estratégico bem definido?
Conformidade com aspectos legais	Existe regras alinhados com aspectos legais implementado na organização? (por exemplo, se dados do cliente devem ser mantidos na estrutura interna; regras para manipulação dos dados clientes)
Políticas internas e governança TI	Existe normas para implementações das funcionalidades? Organização tem governança de TI?
Acordos de Nível de Serviço	Existe SLA por nível de serviço?
Meta	Organização tem alguma meta para o desenvolvimento das funcionalidades?
BPM	A organização utiliza algum modelo de processos de negócio?
Ciclo da vida SOA / Governança	Organização utiliza ciclo da vida e governança SOA?
Similaridade funcional	A organização utiliza padrões para definições das funcionalidades?

Fonte: Börner & Goeken (2009).

3.1.6.3 ***Luthria***

Luthria e Rabhi (2009) propõem um lista de aspectos genéricos de fatores que influenciam a adoção de SOA pelas empresas na ótica dos negócios:

- Os avanços técnicos de SOA;
- Os potenciais benefícios de SOA para a empresa;
- Nível de alinhamento das soluções técnicas baseadas em SOA com as necessidades do negócio;
- Complexidade de implantação de SOA na empresa;

Vantagens competitivas para a empresa com a adoção de SOA.

3.1.6.4 ***Bianco***

Em uma perspectiva mais orientada à TI, Bianco, Kotermanski e Merson (2007) destacam a importância de avaliar a arquitetura de software baseada em SOA durante as etapas iniciais para mitigar riscos de não atendimento às necessidades do negócio. Numa visão mais *bottom-up*, a TI para eles é tida como o alicerce para a efetivação da estratégia de negócios, considerando ao mesmo tempo os necessários e preparados recursos humanos de suporte a todo ciclo de vida do software. Nessa base mais voltado à TI deve-se focar nos mais relevantes aspectos, a saber:

- Características das mensagens;
- Protocolos e tecnologias utilizados (*REST*, *WSDL*, *SOAP*, protocolos proprietários, etc);
- Sincronismo e Assincronismo das mensagens para atender os requisitos funcionais e não funcionais do processo de negócio;
- Mecanismos e abordagens de integração e interoperação;
- Uso de *middlewares*, como ESB (*Enterprise Service Bus*);
- Uso de orquestração de processos coordenado pelo nível de negócios (e.g. via *BPM*);
- Uso de registro de Serviços (e.g. via *UDDI*).

3.2 ENGENHARIA DE SOFTWARE

Engenharia de Software é a área da computação voltada à especificação, desenvolvimento, manutenção e criação de sistemas de software, com aplicação de tecnologias e práticas de gerência de projetos e outras disciplinas, visando a organização, produtividade e qualidade (SOMMERVILLE 2011).

Considerando o foco desta Tese, os temas descritos nessa seção irão nortear apenas os aspectos mais relacionados com o modelo de estimação proposto, a saber: artefatos, requisitos, QoS/SLA e técnicas de estimação de software. Eles são cobertos nas subseções a seguir. No entanto, dado que estimação é um aspecto essencial nesta tese, ela será coberta numa seção à parte, após essas subseções.

3.2.1 Artefatos de Software

As etapas do processo de desenvolvimento de software estão focadas na criação, manipulação e gerenciamento de artefatos de software. Segundo (Kroll & Kruchten, 2003), um artefato é um pedaço de informação que é produzida, modificada ou utilizada por um processo, compreendendo assim desde a especificação do software até a sua implementação, passando por atividades como planejamento, testes e controle de qualidade (PRESSMAN, 2016).

A necessidade de uma nova implementação de software (seja na forma "tradicional" seja na forma de orientação a serviços) pode ter diversos origens, gerando diferentes artefatos. Conforme

(PRIKLADNICKI; WILLI; MILANI, 2014), a origem de uma demanda de novas implementações basicamente advém de:

- Mudanças em legislação;
- Mudanças na estratégia de negócio;
- Introdução de melhorias gerais ou de inovações (de processo, produto, tecnologia ou modelo de negócio);
- Necessidades dos clientes.

Portanto, há sempre um conjunto de *requisitos* a serem cumpridos. Isto é importante na medida que esta tese foca na etapa de Análise, que, em termos gerais toma como base tais requisitos para ao final se poder identificar como os softwares terão que ser desenvolvidos para atendê-los. Para compreender os requisitos é importante abordar os conceitos fundamentais da engenharia de requisitos.

A partir da engenharia de requisitos no desenvolvimento tradicional de software desenvolveu-se a Engenharia de Requisitos Orientada a Serviços (SORE – *Service-oriented Requirements Engineering*). Esta tem por objetivo atender às necessidades de processos para captar os requisitos de serviços, tanto do ponto de vista de consumidores de serviços, quanto de provedores de serviços (LAMSWEERDE, 2000). Conforme Sommerville (2011), a engenharia de requisitos tradicional tem seis etapas distintas: *elicitação*, *análise*, *especificação*, *gerenciamento*, *negociação* e *verificação de requisitos*.

No entanto, as abordagens de engenharia de requisitos tradicional não contemplam algumas das necessidades e particularidades de desenvolvimento baseado em SOA ou devem ser vistas sob óticas mais amplas (TSAI *et al*., 2007) (JOSUTTIS, 2008).

Souza e Fantinato (2014) afirmam que ainda não há trabalhos de referência com abordagens em SORE que abranjam todas as etapas da engenharia de requisitos tradicionais, senão apenas as três primeiras.

3.2.2 Requisitos de Software

Todo projeto de software tem um conjunto de requisitos. Requisito é qualquer condição ou capacidade que deve ser implementada por determinado software ou componente deste para alcançar determinado fim (ENGHOLM, 2010) (SOMMERVILLE, 2011). Pode ser:

- *Requisitos Funcionais (RF)*: são declarações de funcionalidades que devem ser oferecidas pelo sistema e de como devem se comportar em determinadas situações. Requisitos Funcionais podem ser modelados de diversas formas, como por exemplo via diagramas de caso de uso.
- *Requisitos Não Funcionais (RNF)*: são características que não estão diretamente ligadas a serviços específicos que o sistema ofereça a seus usuários, mas sim relacionadas às propriedades do sistema, como desempenho, usabilidade e segurança. Definem restrições do sistema e têm influência na complexidade deste (e dos seus serviços).

Há inúmeros RNF. Mamani (1999), por exemplo, propõe a seguinte classificação (Figura 19):

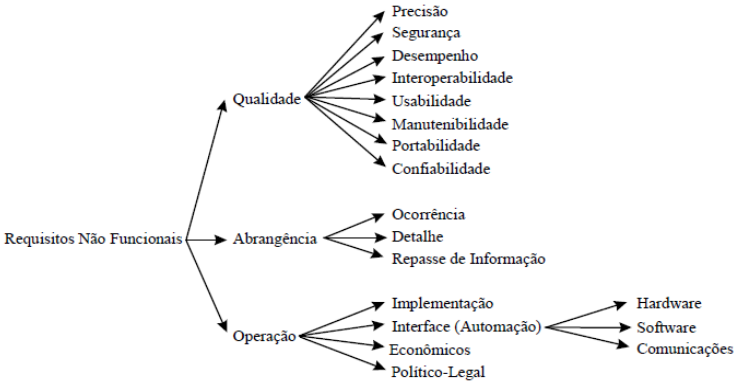


Figura 19: Árvore de RNFs
Fonte: Mamani (1999).

Já Sommerville (2011) apresenta outra forma de classificação de RNFs (Figura 20):

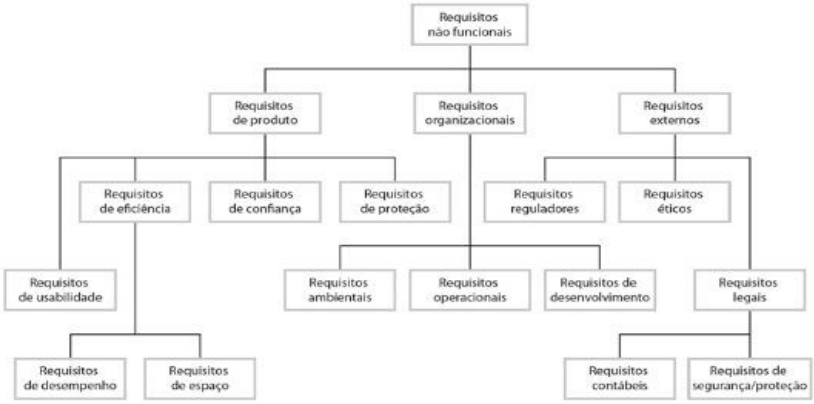


Figura 20: Classificação de RNFs
Fonte: Sommerville (2011).

3.2.3 QoS e SLA

Os requisitos não funcionais costumam ser expressos por atributos e níveis de "qualidade de serviço" (QoS). Garcia e Toledo (2006) listam os seguintes atributos de QoS para "qualificar" os RNFs de um dado sistema (Tabela 3):

Tabela 3: Atributos de QoS

Atributo	Definição
Tempo de resposta	O tempo que o serviço demora para retornar a resposta
Latência	O tempo necessário para iniciar o atendimento de uma requisição de serviço
Taxa de transferência	A taxa de processamento de requisição que um serviço suporta
Escalabilidade	A capacidade de receber cargas de trabalho variadas sem deixar de atender a todos os seus requisitos.
Capacidade	O número de requisições concorrentes que um serviço permite
Disponibilidade	O percentual de tempo em que o serviço está operando
Confiabilidade	Se o sistema executará suas funções como esperado
Acurácia	Taxa de erro do serviço em um intervalo de tempo
Robustez	O nível de "flexibilidade" do serviço para entradas incorretas e invocação de sequências variadas
Estabilidade	Capacidade de evitar efeitos inesperados decorrentes de modificação no software.
Custo	Custo envolvido na utilização do serviço

Segurança	Nível de segurança do serviço em termos de confidencialidade, integridade e autenticação
Mensagem confiável	Se o serviço oferece mecanismos confiáveis de garantia de entrega de mensagens
Integridade	Capacidade de manter a integridade transacional em cenários de alta concorrência.
Interoperabilidade	Capacidade do interagir com outros sistemas sem perdas ou distorções de informação.

Fonte: Garcia e Toledo (2006).

De acordo com Khaluf, Gerth e Engels (2011), o nível geral de QoS oferecido é um dos fatores mais importantes para a o sucesso do negócio. Nesse contexto, são estabelecidos Acordos em Nível de Serviço (SLA) para garantir e formalizar que tais restrições de operação e qualidade devam ser atendidas. No caso de soluções baseadas em serviços de software / SOA, um SLA é o meio mais adequado para especificar e medir os atributos de QoS (FERGUSON; HUSTON, 1998).

3.3 ESTIMAÇÃO DE SOFTWARE

Estimar é prover uma visão do projeto clara o suficiente para que a gerência possa tomar boas decisões de como gerenciar o projeto para que o mesmo atinja seus objetivos (MCCONNELL, 2006). Um dos propósitos da estimação é determinar se esses objetivos são tangíveis o suficiente para que o mesmo possa ser gerenciado de forma a atingir esses objetivos (PRESSMAN, 2016).

McConnell (2006) define boas estimativas do ponto de vista estatístico como aquelas que variam em até 25% durante pelo menos 75% das vezes. Este critério de certa forma empírico do autor reforça o conceito de que boas estimativas são as que proporcionam boas aproximações suficientes para que a maioria das decisões sobre projetos estejam corretas. Com isto, a estimação deve considerar diversas influências sobre o desenvolvimento de software, como os fatores relacionados ao problema, ao cliente, ao ambiente de desenvolvimento e às pessoas envolvidas.

Com o objetivo de estimar com mais acurácia os projetos de software, diversas técnicas de medição têm sido propostas para, em suma, minimizar os fracassos dos projetos obtidos. As seções a seguir sumarizam as técnicas de estimativa de projetos de software tidas como referências na literatura.

3.3.1 Estimativa por analogia

O objetivo desta técnica é a criação de estimativas para um novo projeto por meio de comparação deste com projetos anteriores, identificando diferenças e similaridades (MCCONNELL, 2006). Neste caso, o gestor pode realizar ajustes nos dados obtidos em casos anteriores e aplicar no que está sendo o objeto atual de trabalho.

Encontrar semelhanças e diferenças entre softwares é a principal dificuldade desta técnica. Como alternativa tem-se procurado automatizar o processo, utilizando-se para isto a distância euclidiana entre os casos, que pode ser obtida pela fórmula abaixo (HUGHES; COTTERELL, 1999) (Equação 1):

$$\text{Distância} = \sqrt{\sum_{i=1}^{n} (parâmetro\_atual_i - parâmetro\_origem_i)^2}$$

Equação 1: Distância Euclidiana
Fonte: Hughes; Cotterell (2006).

3.3.2 Estimativa de três pontos ou PERT (Program evaluation and Review Technique)

Esta técnica baseia-se em três tipos de estimativas para apresentar um resultado mais próximo possível do desejado. As estimativas são classificadas como a mais provável (M), a otimista (O) e a pessimista (P) (PMI, 2013) (HUGHES; COTTERELL, 2006):

- A mais provável (M) prevê expectativas realistas para a estimativa, ou seja, espera-se que apresente os resultados em circunstâncias normais;
- O melhor caso possível é focado na estimativa otimista (O), quando se assume que nenhum problema grave irá ocorrer, por vezes até mesmo superando o desempenho em casos equivalentes anteriores;
- A estimativa pessimista (P) considera a ocorrência de todas as circunstâncias e eventualidades que possam prejudicar o projeto ou o acontecimento de algumas delas num nível profundo de impacto no projeto.

O PERT combina as três estimativas previstas utilizando uma fórmula (Equação 2), em que T é a média para a estimativa do parâmetro desejado.

$$T = (O + 4M + P)/6$$

Equação 2: Fórmula PERT
Fonte: Hughes; Cotterell (2006).

É possível para a técnica PERT calcular o desvio padrão que proporciona um nível de confiança para a estimativa encontrada (Equação 3).

$$\text{Desvio Padrão} = (P - O)/6$$

Equação 3: Fórmula Desvio Padrão
Fonte: Hughes; Cotterell (2006).

Quanto maior for o desvio padrão obtido maior é a distância entre o previsto entre as estimativas Otimista e Pessimista. Portanto, se deseja que o resultado do desvio padrão seja pequeno.

3.3.3 Opinião especializada

Esta técnica é a mais comum de ser utilizada. As pesquisas apontam que 72% dos projetos utilizam opinião especializada para estimar um projeto de software (KITCHENHAM; PFLEEGER, 2002 apud MCCONNELL, 2006).

Os membros da equipe com conhecimento especializado e com conhecimento de dados históricos podem fornecer estimativas na área desejada e específica de seu conhecimento (PMI, 2013). Este tipo de estimativa exige que exista um especialista disponível (MCCONNELL, 2006) (LAIRD; BRENNAN, 2006).

3.3.4 Linhas de código

A técnica da estimação por linhas de código (LOC – *Lines of Code*) é uma das medidas mais antigas para estimar o tamanho, esforço e, consequentemente, a produtividade no desenvolvimento de software. Consiste na contagem da quantidade do número de linhas de código de um programa de software, sendo de fácil automação e requerendo pouco

esforço manual. Os dados obtidos por esta técnica podem ser utilizados para dimensionar cada elemento do software, podendo ser advindos de dados históricos de projetos (PRESSMAN, 2016).

Nesta técnica, após a delimitação do escopo do projeto, este é decomposto em funções, que serão estimadas individualmente, prevendo o número de linhas de código-fonte por pessoa/mês.

Assim como as demais, esta técnica também apresenta algumas limitações (SOMMERVILLE, 2011):

- Mede a produtividade somente após a etapa de codificação;
- Depende da experiência do desenvolvedor, pois o número de linhas de código pode variar de pessoa a pessoa;
- Depende da linguagem de programação;
- Ausência de padrões de contagem, já que não há uma definição de certas características, como a contagem de comentários, declarações de dados.

Um modelo para a predição do esforço, considerando as LOC, será através da seguinte equação:

$$E = S * fpl$$

Equação 4: Esforço baseado LOC
Fonte: Sommerville (2011).

Onde:
E = Esforço aplicado (em pessoas/mês).
S = Tamanho do sistema em KLOC (milhares de linhas de código).
Fpl = Fator de Produtividade Linear (1/(KLOC/pessoas-mês)).

Analisando a equação, pode-se assumir que a relação entre esforço, tamanho (quantidade de linhas de código) e produtividade (baseado experiência dos profissionais) é linear e ilustrado na Figura 21.

Ainda, considerando que baseado nesta técnica, a quantidade de linhas de código representa o tamanho do serviço, e produtividade depende da experiência dos profissionais. A Figura 21 ilustra os fatores e suas inter-relações.

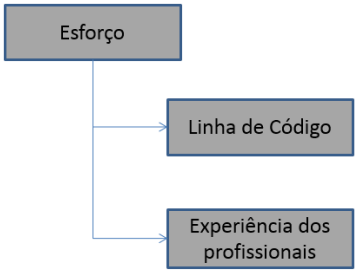


Figura 21: Relações entre fatores
Fonte: Autor.

3.3.5 COCOMO II (Constructive Cost Model)

O COCOMO II é uma técnica que tem o objetivo de estimar o processo de desenvolvimento de software, fornecendo ferramentas de suporte e melhoria contínua e provendo um *framework* analítico. Este modelo permite estimar *esforço*, *tempo* e *tamanho* de desenvolvimento de software (BOEHM et al., 1997).

Ele foi projetado para trabalhar com três modelos em diferentes estágios (PRESSMAN, 2016) (SOMMERVILLE, 2011):

- *Application composition* - prototipação: Neste estágio é necessário estimar o esforço por "pontos de objeto", que são um tipo de contagem funcional de telas e relatórios onde cada um dos elementos contados recebe uma classificação em níveis de complexidade (*simples*, *média* e *alta*).
- *Early design* – arquitetura: As estruturas fundamentais do software são projetadas no nível inicial do projeto utilizando pontos por função.
- *Post architecture* – desenvolvimento: Quando um projeto se apresenta pronto para ser desenvolvido já deve possuir uma arquitetura dentro de um ciclo de vida que possa fornecer informações mais precisas para as entradas dos direcionadores de custo, proporcionando estimativas mais precisas. Esse modelo prevê a utilização de linhas de código e/ou pontos de função.

Neste sentido, para cada estágio o COCOMO II sugere o cálculo diferenciado de estimativas atentando para as informações existentes em cada um desses níveis no intento de alcançar a melhor precisão possível.

Em relação ao seu modelo predecessor, COCOMO "I" (BOEHM, 1984), no COCOMO II foi inserido o nível de prototipação, buscando auxiliar na realização da estimativa de esforço em projetos desta natureza e nos quais o desenvolvimento ocorre pela composição de componentes já existentes (HUGHES; COTTERELL, 1999) (SOMMERVILLE, 2011).

Neste caso, o esforço (E) é medido em pessoa/mês (Equação 5):

$$E = (NPO * (1-\%reuso/100))/PROD$$

Equação 5: Cálculo do esforço (Pessoa/mês)
Fonte: Sommerville (2011).

Onde:

NPO (número de pontos de objeto) é usado para estimar o tamanho de aplicações. Esta estimativa se baseia no número de telas exibidas, no número de relatórios produzidos e no número de módulos na linguagem de programação. As telas exibidas classificam-se em: *simples* (1 ponto), *moderadamente complexas* (2 pontos) e *muito complexas* (3 pontos). No caso dos relatórios, são classificados como *simples* (2 pontos), *moderadamente complexos* (5 pontos) e *muito complexos* (8 pontos) (SOMMERVILLE, 2011).

O percentual de *reuso* refere-se ao percentual que se espera para aquele projeto. No COCOMO II há uma calibração de 2,94. Em casos em que a empresa tenha outros sistemas semelhantes, o percentual de reuso mede o quanto o projeto será copiado ou modificado a partir de produtos já existentes (SOMMERVILLE, 2011).

PROD é a produtividade, obtida dentro de cinco níveis (Tabela 4), conforme os "padrões" de cada empresa.

Tabela 4: Obtenção de produtividade

Experiência e capacitação do Desenvolvedor	Muito baixa	Baixa	Nominal	Alta	Muito Alta
Maturidade e capacitação de CASE	Muito baixa	Baixa	Nominal	Alta	Muito Alta
PROD	4	7	13	25	50

Fonte: Sommerville (2011).

Referente ao cálculo de esforço, o nível inicial do projeto há uma equação (Equação 6) considerada padrão que é adotada, em que *E* é o esforço. Boehm propôs que o coeficiente *A* deve ser 2,5 para as estimativas feitas nesse nível. O *T* (tamanho) é obtido por KLOC, ou seja, o número de milhares de linhas de código-fonte. (SOMMERVILLE, 2011). O expoente *B* refere-se ao aumento do esforço à medida que o projeto também aumenta. Esse expoente não é fixo, como na primeira versão do COCOMO, e pode variar de 1,1 a 1,24 (SOMMERVILLE, 2011). *M* refere-se a um conjunto de direcionadores de processos (Tabela 5) :

$$E = A * T^{B} * M$$

Equação 6: Fórmula-padrão
Fonte: Sommerville (2011).

Esses direcionadores podem ser classificados em uma escala de 1 a 6, sendo 1 considerado muito baixo e 6 muito alto (Tabela 5). Nesta tabela, cada linha representa a faixa de valor mínimo e máximo a ser utilizado nos cálculos. Por exemplo, no caso *RCPX*, a faixa seria entre 0,75 a 1,66 e no caso *RUSE* seria entre 0,91 a 1,49.

Tabela 5: Direcionadores de cálculo

Sigla	Descrição	Muito Baixo	Baixo	Normal	Alto	Muito Alto	Extra Alto
RCPX	Confiabilidade e complexidade	0,75	0,88	1	1,09	1,13	1,66
RUSE	do produto	-	0,91	1	1,14	1,29	1,49
PDIF	Reuso requerido	-	0,87	1	1,06	1,21	1,57
PERS	Dificuldade de plataforma	1,24	1,1	1	0,83	0,67	-
PREX	Capacitação pessoal	1,22	1,10	1	0,88	0,81	-
SCED	Experiência pessoal	1,29	1,1	1	1	1	1
FCIL	Tempo de desenvolvimento	1,24	1,10	1	0,86	0,72	0,78

Fonte: Hughes; Cotterell (1999).

Neste caso, o cálculo de M é dado por:

$$M = RCPX \times RUSE \times PDIF \times PERS \times PREX \times SCED \times FCIL$$

Equação 7: Cálculo do conjunto simplificado de direcionadores de processo
Fonte: Sommerville (2011).

3.3.6 Análise por pontos de função

De acordo com Albrecht e Gaffney (1983), esta técnica surgiu da necessidade de identificar o tamanho funcional de um programa, independente da linguagem de programação com a qual seria desenvolvido. Para tal, há que se considerar cinco dos chamados componentes da técnica (HUGHES; COTTERELL, 1999):

- Entradas externas: referem-se a entradas de transações que alimentarão arquivos internos do sistema.
- Saídas externas: são transações em que os dados são externalizados ao usuário, normalmente na forma de relatórios impressos ou na tela.
- Arquivos internos lógicos: são arquivos utilizados internamente pelo sistema.
- Arquivos externos de interface: são os que permitem a entrada e saída de dados em uma aplicação computacional.
- Consultas externas: são informações solicitadas pelo usuário, mas que não permitem a atualização de arquivos internos do sistema.

Para esta técnica adotam-se graus de dificuldade, definidos como *baixo, médio* e *alto*, relacionados com cada componente (Tabela 6). Os valores utilizados nesta tabela são os definidos por Albrecht e Gaffney (1983).

Tabela 6: Pesos dos componentes de acordo com as complexidades

Componente	Baixa	Média	Alta
Entradas externas (*External Inputs* - EI)	3	4	6
Saídas externas (*External Outputs - EO*)	4	4	7
Arquivos internos lógicos (*Internal Logical Files* - ILF)	7	10	15
Arquivos de interface externos (*External Interface* Files-EIF)	5	7	10
Consultas externas (*External Inquiry* - EQ)	3	4	6

Fonte: IFPUG (2010).

Baseado nisso podem-se calcular os pontos por função (*UFP*), consistindo na fórmula da Equação 8:

$$UFP = \Sigma(EI * PESO)+\Sigma(EO* PESO)+\Sigma(ILF * PESO)+\Sigma(EIF * PESO)+ \Sigma(EQ* PESO)$$

Equação 8: Cálculo do Ponto de Função não ajustado
Fonte: IFPUG (2010).

Na Equação 8, para composição da fórmula, deve-se separar cada componente por complexidade baseado nos pesos identificados na Tabela 6. Em seguida, a quantidade de cada componente de uma complexidade deve ser multiplicada por seu peso e por fim o somatório.

Com base no UFP é realizado um ajuste baseado na complexidade ou dificuldade de implementação do sistema. Para isso torna-se necessário calcular o valor do fator de ajuste (*value adjustment factor* - VAF). Para tal tem-se um conjunto de 14 características de sistema, com perguntas relacionadas que ajudam a determinar qual o grau de influência de cada característica para um projeto em questão:

1- Requer cópias de segurança confiáveis?
2- O sistema requer comunicação de dados?
3- Existe processamento distribuído?
4- O desempenho é um aspecto crítico?
5- O sistema rodará em um ambiente operacional existente e intensamente utilizado?
6- O sistema requer entrada on-line?
7- As transações de entrada on-line são compostas em múltiplas telas ou operações?
8- Os arquivos são utilizados on-line?
9- As entradas, saídas, arquivos ou consultas são complexas?
10- O processamento interno é complexo?
11- O código é projetado/construído para ser reutilizado?
12- O projeto inclui conversão ou instalações?
13- O sistema é projetado para o uso em múltiplas instalações em diferentes organizações?
14- O sistema é projetado para ser de fácil utilização e para facilitar mudanças?

Para tanto, utilizam-se de pesos, classificados numa escala de 0 a 5 (Tabela 7).

Tabela 7: Graus de influência das características de sistema de UFP

Peso	**Grau de influência**
0	Insuficiente
1	Mínima
2	Moderada
3	Média
4	Alta
5	Essencial

Fonte**:** IFPUG (2010).

O Grau Total de Influência (*TDI – Total Degree of Influence*) é obtido quando se faz o somatório de todos os pesos relacionados àquelas 14 características do sistema. Desta forma, é possível obter o Ponto de Função ajustado, conforme Equação 9. No caso, 0.01 é o Coeficiente por Grau de Influência e 0.65 é uma constante empírica que pode alterar a contagem de pontos de função numa amplitude de –35% até +35%. O valor final do fator de ajuste pode variar de 0,65 até 1,35. O fator de ajuste é responsável pela correção das distorções ocorridas na etapa de contagem das funções de dados e funções transacionais.

$$VAF = (TDI * 0.01) + 0.65$$

Equação 9: Valor do fator de ajuste
Fonte: IFPUG (2010).

O valor do fator de ajuste pode ser sempre calculado. Entretanto, em diversas situações ele não é aplicado para ajustar a contagem por diversas razões, entre elas a aderência estrita aos padrões definidos pela ISO/IEC para mensuração de software (a norma ISO/IEC 10926 só reconhece a APF como uma medida de software válida se não for aplicado o fator de ajuste), razões de mercado e também para que a contagem esteja em conformidade com as definições incluídas em editais ou contratos (SUMMERVILLE, 2011).

Para finalizar este processo, obtém-se o Ponto de Função ajustado (AFP) a partir da Equação 10.

$$AFP = UFP * VAF$$

Equação 10: Ponto de Função ajustado
Fonte: IFPUG (2010).

Esta técnica possibilita a estimativa de tamanho de um sistema a partir do número de características/funcionalidades visíveis e previstas no projeto. Para obter o tamanho do projeto faz-se a conversão de pontos de função para LOC (Seção 3.3.4) utilizando uma tabela com a média de LOC por Ponto de Função considerando as diferentes linguagens de programação. Neste caso, é preciso saber a linguagem de programação que será utilizada no desenvolvimento do projeto que está sendo estimado (BFPUG, 2008).

A recomendação do BFPUG (2008) é que sejam medidos alguns projetos para determinar o valor médio da razão LOC por PF em cada caso específico. No entanto, existem valores baseados em vários

projetos que indicam uma relação referida para cada situação, apresentada na
Tabela 8 (BFPUG, 2008):

Tabela 8: LOC por pontos de função(PF)

Linguagem	Baixo	Mediano	Alto
Access	15	38	47
Ada	104	-	205
ASP	32	62	127
Assembler	86	157	320
C	9	104	704
C++	29	53	178
C#	51	59	66
Clipper	27	39	70
COBOL	8	77	400
DBase III	-	-	-
DBase IV	-	-	-
Excel	31	46	63
FORTRAN	-	-	-
FoxPro	25	35	35
HTML	35	42	53
Informix	24	31	57
J2EE	50	50	100
Java	14	59	97
JavaScript	44	54	65
JCL	21	48	115
JSP	-	-	-
Lotus Notes	15	22	25
Mapper	16	81	245
Oracle	4	29	122
Perl	-	-	-
PL/1	22	58	92
PL/SQL	14	31	110
SAS	33	41	49
Smalltalk	17	32	55
SQL	15	35	143
VBScript	27	34	50
Visual Basic	14	42	276
VPF	92	95	101
Web Scripts	9	15	114

Fonte: BFPUG (2008).

Nesse sentido, para se obter o tamanho do software utilizando a conversão para LOC, multiplica-se o número encontrado na

Tabela 8 pelos pontos de função encontrados (AFP). De posse de dados históricos das empresas é possível estimar esforço, considerando a produtividade da equipe; por exemplo, quantas pessoas por mês a equipe leva para produzir 1 Ponto de Função. Com esta informação é possível aplicar a equação 11:

Esforço = produtividade * AFP

Equação 11: Cálculo do esforço
Fonte: IFPUG (2010).

A estimativa de tempo de desenvolvimento também é possível desde que se tenha o esforço (E), o tamanho da equipe de trabalho (TE) e a quantidade de horas diárias trabalhadas (HT) por seus integrantes, conforme equação 12.

Tempo de desenvolvimento (em dias) = E (horas) / (TE *HT)

Equação 12: Cálculo do tempo de desenvolvimento
Fonte: IFPUG (2010).

Com a aplicação das equações 11 e 12 finaliza-se o processo de obtenção dos atributos esforço e tempo de desenvolvimento com base no tamanho do software obtidos em pontos por função por meio da técnica de análise por pontos de função (IFPUG, 2010).

Neste trabalho de tese optou-se pelo uso da análise por pontos de função criada por Albrecht (1981) e atualmente o desenvolvimento continua com IFPUG (2010). No entanto, torna-se relevante salientar que existem outras versões desta técnica desenvolvidas e adaptadas por outras entidades, como MARK II, NESMA, COSMIC-FFP.

O MARK II é um método de análise quantitativa e medição de tamanho funcional, considerando o conjunto de requisitos funcionais exigidos pelos usuários para aplicação de produto de software. Este método destina-se a se alinhar à norma ISO14143/1 e à norma internacional para Medidas de Tamanho Funcional (UKSMA, 1998).

A diferença entre o AFP e o MARK II encontra-se no fato de que, no primeiro, a mensuração da contagem dos arquivos lógicos ocorre apenas uma vez para cada parte do software, diferentemente do MARK II, em que toda vez que "tipos de entidades" são referenciados em cada transação lógica estas são então mensuradas.

Apesar da sua importância e grande uso por empresas, a técnica AFP tradicional não atende a todos os requisitos da SOA (Mahmood et al., 2012). Um desenvolvimento baseado SOA requer a aplicação

explícita de métodos e princípios para obtenção de baixo acoplamento, capacidade de composição, padronização e mediação de transações, entre outros, que não estão presentes no desenvolvimento de software convencional (ERL, 2009).

Muitos ajustes para a abordagem tradicional de Análise Ponto Função têm sido propostas a fim de estimar o esforço de projetos SOA. Um deles é *Cosmic*.

3.3.7 Cosmic Function Point for SOA

A técnica nomeada COSMIC-FFP (*Common Software Measurement International Consortium – Full Function Points*) é um método padrão de medição de tamanho funcional do software a partir domínios funcionais para sistemas de informações gerenciais e software de tempo real. Esta técnica é descrita na norma ISO/IEC 19761: 2011, onde especifica o conjunto de definições, convenções e atividades do método que é aplicável aos softwares de aplicação, software de tempo real ou ainda sistemas híbridos (ISO/IEC, 2011).

O COSMIC preocupa-se apenas com os requisitos funcionais do usuário (COSMIC, 2007) e foi proposta para superar as limitações do Ponto de Função tradicional quando aplicado a SOA (SETH; AGRAWAL; SINGLA, 2014).

O Processo de medição COSMIC consiste de três etapas principais: (i) definir a estratégia de medição; (ii) mapear os Requisitos Funcionais do Usuário; e (iii) executar a mediação do modelo COSMIC. A Figura 22 ilustra o cenário tradicional do COSMIC.

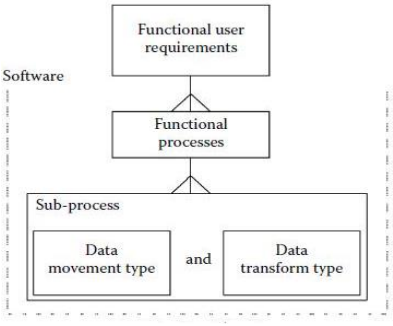


Figura 22: Fatores baseado COSMIC
Fonte: Zayaraz; Thambidurai (2006).

Conforme mostra a figura, os requisitos funcionais de usuário são alocados para uma ou mais partes do software e são decompostos e representados por processos funcionais. Cada processo funcional, por sua vez, é representado por subprocessos. Um subprocesso pode ser um tipo de movimento de dados ou um tipo de transformação de dados.

Um movimento de dado, move um único grupo de dados de atributos descrevendo um único "objeto de interesse", onde o último são dados de interesse para um usuário funcional;

Um processo funcional COSMIC é definido como um conjunto de movimentos de dados únicos e ordenados que são desencadeados por um evento fora do software que está sendo medido e que, quando completo, deixa o software em um estado coerente em relação ao evento externo.

3.3.8 Use Case Point - UCP

Os Use Case Point (UCP), ou Pontos de Casos de Uso (PCU), foram propostos em 1993 por Gustav Karner com base nos *Function Points* (FP), *Mark II* (KARNER, 1993) e no *Modelo de Casos de Uso* para determinar estimativas de tamanho de softwares orientado a objetos. Os PCU estabelecem uma medida para tamanho do software baseado na complexidade das funcionalidades do software.

Neste modelo, Karner (1993) separa o fator da complexidade por categorias e o subdivide em complexidade técnica e ambiental.

O fator de complexidade técnica varia numa faixa de 1 a 5, de acordo com o grau de dificuldade do sistema a ser construído. O número 1 representa pouco complexo e 5 muito complexo. Exemplos de dificuldades incluem o desempenho da aplicação, aspectos de portabilidade e facilidade de manutenção.

Já o fator de complexidade ambiental indica a eficiência do projeto, numa escala de 1 a 5, e está relacionado ao nível de experiência dos profissionais e às condições ambientais e de trabalho.

Os fatores de complexidade técnica e ambiental são importantes nas estimativas de riscos e custos do projeto (KARNER, 1993). Dessa forma, é possível calcular mais rapidamente as mudanças nas estimativas do sistema a cada pequena alteração de requisitos, refazendo-se apenas alguns cálculos. O *Function Point*, ao contrário, exige que novos documentos para o cálculo das estimativas sejam adicionados ao sistema a cada pequena mudança no orçamento, prazo ou requisitos, sendo, dessa forma, menos flexível às mudanças. Além disso,

os Pontos de Caso de Uso contribuem para a diminuição de algumas dificuldades impostas pelo mercado em relação à resistência de adoção de métricas de estimativa, pois é um método simples, fácil de usar e rápido de se aplicar (SOMMERVILLE, 2011).

Em projetos de software em que se utiliza análise orientada a objetos, os requisitos funcionais são representados por meio de Casos de Uso (*Use Cases*). Estes podem ser definidos como sendo a forma com a qual o usuário utiliza o sistema, sendo compostos por um conjunto de interações sequenciais referentes às relações entre o sistema que está sendo desenvolvido e seu exterior (SOMMERVILLE, 2011).

O método necessita que seja possível que para cada caso de uso seja contabilizado o número de transações, sendo estas consideradas qualquer evento que ocorra entre o ator e o sistema, podendo ser realizado por completo ou não (BENTE e*t al.*, 2001).

A análise por Pontos de Caso de Uso tem basicamente quatro passos a seguir:

I) Primeiramente, há uma categorização dos atores dos casos de uso em Simples (S), Médio (M) ou Complexo (C), e o cálculo de um valor não ajustado dos atores. Os pesos para cada categoria são 1, 2 e 3, respectivamente (SOMMERVILLE, 2011) (PRESSMAN, 2016).

Atores podem ser classificados como:

- Simples: refere-se, por exemplo, a uma API (*Application Programming Interface*) com outro sistema; possui uma interface bem definida; as saídas do sistema ou entrada recebidas por ele são bastante previsíveis.
- Médio: quando a comunicação é realizada com um sistema externo por meio de um protocolo como o TCP/IP ou ainda representa um sistema de hardware. Estes atores têm mais propensões a erro e são mais difíceis de serem controlados.
- Complexo: representa pessoas que irão interagir com o sistema, normalmente via interface gráfica ou uma página web. Por isso há maior complexidade neste tipo de ator, já que este pode executar qualquer atividade sem previsibilidade.

Finalizando este passo calcula-se o total não ajustado do ator (*unadjusted actor weights* - UAW), somando a quantidade classificada para cada categorização e multiplicando estes totais por seus respectivos pesos (Equação 13).

$$UAW = \Sigma\ Atores*S + \Sigma\ Atores*M + \Sigma\ Atores*C$$

Equação 13: Total não ajustado de atores - UAW
Fonte: Sommerville (2011); Pressman (2016).

II) A segunda etapa constitui-se em classificar os casos de uso também em Simples (S), Médio (M) ou Complexo (C) considerando-se o número de transações existentes em cada caso de uso. Os pesos para cada categoria e número de transações são definidos na Tabela 9 (SOMMERVILLE, 2011 ; PRESSMAN, 2016).

Tabela 9: Relação número de transações e pesos de caso de uso

Categorias	Número de transições	Pesos
Simples (S)	1..3	5
Médio (M)	4..7	10
Complexo (C)	8..*	15

Fonte: Sommerville (2011); Pressman (2016).

Pode-se também realizar a classificação medindo a complexidade dos casos de uso baseado na contagem das classes de análise que identificam como um caso de uso é implementado (SOMMERVILLE, 2011) (PRESSMAN, 2016) (Tabela 10).

Tabela 10: Relação número de transações e pesos de caso de uso considerando as classes

Categorias	Número de Classes	Pesos
Simples (S)	1..4	5
Médio (M)	5..10	10
Complexo (C)	11..*	15

Fonte: Sommerville (2011) ; Pressman (2016).

Para completar deve-se calcular o total não ajustado de casos de uso (*unadjusted use case weights* - UUCW) somando a quantidade de casos de uso em cada categorização e multiplicando por seu peso respectivo (Equação 14):

$$UUCW = \Sigma\ USC*S + \Sigma\ USC*M + \Sigma\ USC*C$$

Equação 14: Cálculo do total não ajustado de caso de uso
Fonte: Sommerville (2011) ; Pressman (2016).

O Ponto de Caso de Uso não ajustado (UUCP) é calculado somando-se total não ajustado do ator (UAW) obtido pela Equação 13 e o total não ajustado de caso de uso (UUCW) obtido pela Equação 14. Esta operação pode ser observada na Equação 15.

$$UUCP = UAW + UUCW$$

Equação 15: Cálculo do total não ajustado de caso de uso
Fonte: Sommerville (2011) ; Pressman (2016).

III) A próxima etapa é a obtenção do valor ajustado de Ponto de Caso de Uso considerando fatores técnicos, como complexidade e fatores ambientais, de maneira a quantificar requisitos não funcionais (por exemplo, a usabilidade e motivação do programador) (RIBU, 2001).

Cada fator técnico é associado a um peso que varia de 0 a 5, dependendo da influência que exerce sobre o projeto como um todo (KARNER, 1993). Neste caso, o valor 0 significa que o fator é irrelevante no contexto do projeto; o valor 3 significa que o fator é relevante com um grau de influência média; o valor 5 significa que o fator é essencial para o sucesso do projeto.

A Tabela 11 apresenta os fatores técnicos a serem considerados para encontrar o valor ajustado de Ponto de Caso de Uso.

Tabela 11: Fatores técnicos de complexidade

Fator	Descrição	Peso
T1	Sistema distribuído	2
T2	Desempenho	2
T3	Eficiência do usuário	1
T4	Complexidade de processamento	1
T5	Reuso	1
T6	Facilidade para instalação	0.5
T7	Usabilidade	0.5
T8	Portabilidade	2
T9	Manutenibilidade	1
T10	Concorrência	1
T11	Segurança	1
T12	Conexão com outros sistemas	1
T13	Necessidade para treinamento de usuário	1

Fonte: Sommerville (2011); Pressman (2016).

Baseado na Tabela 11 inicia-se o cálculo do Fator de Complexidade Técnica (*Technical Complexity Factor - TCF*), multiplicando-se cada fator por seu peso e em seguida realizando a somatória da multiplicação citada obtendo-se assim o TFator. Na Equação 16, FatorTn identifica cada fator técnico descrito na Tabela 11 e PesoTn reflete o peso do determinado fator técnico.

$$TFator = \Sigma\ FatorTn * PesoTn$$

Equação 16: Cálculo do TFator
Fonte: Sommerville (2011); Pressman (2016).

Segue abaixo detalhamento da Tabela 11:

- Sistema Distribuído: descreve o nível em que a aplicação transfere dados entre seus componentes. Um sistema de desktop isolado (aplicação e banco de dados local) pontuará com 0. Um sistema de n camadas pontuará com 4. Para pontuar com 5, o sistema dever ter componentes executando em múltiplos servidores ou processadores.
- Desempenho: descreve o nível em que considerações sobre o tempo de resposta e taxa de transações influenciam no desenvolvimento da aplicação. A questão que deve ser avaliada é "quão rápida deve ser a aplicação e o quanto isto influencia o projeto?". Por exemplo, um sistema bancário onde uma regra de negócio tenha que o tempo para a autenticação de um documento pelo caixa não possa ser superior ao atual sistema teria pontuação 5.
- Usabilidade: descreve em que nível considerações sobre fatores humanos e facilidade de uso pelo usuário final influenciam no desenvolvimento da aplicação. Aplicações servidoras que não possuam interação com usuário final pontuarão 0. Outras pontuarão de 3 a 5.
- Complexidade de Processamento: descreve em que nível o processamento influencia o desenvolvimento da aplicação. A complexidade de processamento influencia no dimensionamento do sistema, e, portanto, deve ser quantificado o seu grau de influência. O fato de haver uma funcionalidade com grande processamento matemático deve ser considerado no contexto de todo o sistema.
- Reuso: descreve em que nível a aplicação e o seu código foram especificamente projetados, desenvolvidos e suportados para serem utilizados em outras aplicações.
- Facilidade de Instalação: determina o grau de facilidade no processo de instalação do sistema. Normalmente o desenvolvedor define como será implementado o instalador, mas se o cliente está procurando por uma instalação sob medida pode depender de um módulo mais complexo.

- Facilidade de operação da aplicação: descreve em que nível a aplicação atende a aspectos operacionais, por exemplo, se procedimentos de inicialização, segurança e *backup* foram desenvolvidos e testados.
- Portabilidade: descreve em que nível a aplicação foi especificamente projetada, desenvolvida e suportada para ser instalada em múltiplas plataformas de hardware e software.
- Manutenibilidade: descreve em que nível a aplicação foi desenvolvida para facilitar a mudança de sua lógica ou estrutura de dados.
- Concorrência: descreve em que nível a aplicação foi especificamente desenvolvida para proporcionar acessos simultâneos.
- Exigências de Segurança: descreve em que nível a aplicação foi especificamente desenvolvida para proporcionar variados e devidos níveis de segurança.
- Desacoplamento: indica o grau de interdependência do projeto em relação a uso de controles externos (depender de estímulos ou estimular, componentes / sistemas externos), sendo assim necessário um determinado esforço no sentido de compreender esses controles e avaliar seus prós e contras em relação a sua utilização. Valor do grau de dependência:
  0 = Não há interdependências com componentes externos;
  1 = Há pouco acesso a/de sistemas externos;
  2 = Esporadicamente ocorrem acessos a/de componentes externos;
  3 = Um médio número de acessos a/de sistemas externos;
  4 = Frequentemente ocorrem acessos a/de componentes externos;
  5 = Forte dependência com sistemas externos.
- Exigência de Treinamento: descreve a complexidade de utilização da aplicação sob a perspectiva do usuário e assim do nível de treinamento necessário específico.

Para encontrar o valor final do TCF deve-se multiplicar o TFactor por 0.1 e somar a 0.6 (Equação 17). A constante 0.6 deriva da técnica de FPA proposta por Albrecht (1981) em que se tinha 0,65.

$$TCF = 0.6 + (0.01 * TFator)$$

Equação 17: Equação 14 - Cálculo do TCF
Fonte: Sommerville (2011); Pressman (2016).

Na sequência deve-se calcular o Fator Ambiental tendo como base a Tabela 12, que elenca os fatores considerados, descrições e pesos destes.

Tabela 12: Fatores ambientais

Fator	Descrição	Peso
F1	Equipe tem conhecimento no processo de desenvolvimento	1.5
F2	Experiência da equipe com o tipo de aplicação	0.5
F3	Experiência da equipe com orientação a objetos	1
F4	Capacidade de análise chefe	0.5
F5	Motivação da equipe	1
F6	Estabilidade dos requisitos	2
F7	Trabalhadores em tempo parcial	1
F8	Dificuldade com a linguagem de programação escolhida	2

Fonte: Sommerville (2011); Pressman (2016).

Desta forma, cada um destes fatores deve ser classificado de acordo com seu grau de influência no projeto, que pode variar entre 0, 3 ou 5, de acordo com os valores da Tabela 13.

Tabela 13: Grau de influência dos fatores

Fatores	0	3	5
F1 a F4	Nenhuma experiência	Experiência média ou	Alta experiência
F5	Nenhuma	Média ou normal	Alta
F6	Mudam constantemente	Mudam na média	Praticamente não
F7	Não haverá	Poucos – Média	Pessoal Técnico
F8	Fácil	Média	Muito difícil

Fonte: Sommerville (2011); Pressman (2016).

O cálculo do Fator de Ambiente (Environmental Factor - EF) é feito multiplicando-se cada fator por seu peso e realizando a somatória de todos os produtos. Observando-se a Equação 18, FatorFn se refere a cada um dos fator da Tabela 12, e PesoFn ao peso atribuído a este conforme Tabela 13.

$$\text{EFator} = \Sigma\ FatorFn * PesoFn$$

Equação 18: Cálculo do EFactor
Fonte: Sommerville (2011); Pressman (2016).

Para encontrar o valor final do EF deve-se realizar o cálculo da Equação 19.

$$EF = 1.4 + (-0.03 * \text{EFator})$$

Equação 19: Cálculo do EF
Fonte: Sommerville (2011); Pressman (2016).

Com base nos cálculos do TCF e EF é possível obter o valor total ajustado de Ponto por caso de uso (*adjusted use case points* - UPC) por meio da Equação 20.

$$UPC = UUCP*TCF*EF$$

Equação 20: Cálculo do UPC
Fonte: Sommerville (2011); Pressman (2016).

O UPC é obtido por meio da multiplicação entre o total não ajustado de ponto por caso de uso (UUCP) pelo Fator de complexidade técnica (TCF) e Fator ambiental (EF).

IV)

Por fim, pode-se obter o esforço necessário a determinado projeto de software, ou seja, o número total de pessoa/horas necessárias para realizar todo projeto (KARNER, 1993); (SOMMERVILLE, 2011); (PRESSMAN, 2016).

O valor "padrão" de representação de produtividade é um fator de 20 pessoas/hora por Ponto de Caso de Uso (BENTE *et al.*, 2001). Porém, se o número de fatores ambientais entre F1 e F6 (Tabela 13) for entre 3 e 4, o esforço é de 28 pessoas/horas; e se passar de 4 o esforço é de 36 pessoas/horas (RIBU, 2001) (BENTE *et al.*, 2001).

3.3.9 PUTNAM

O modelo de Putnam (1978) relaciona o número de linhas de código ao tempo e esforço de desenvolvimento. É um modelo dinâmico de múltiplas variáveis que pressupõe uma distribuição do esforço específico ao longo da existência de um projeto de desenvolvimento de software. O esforço é calculado como mostrado na Equação 21.

$$K = L^3/(C_k^3.td^4)$$

Equação 21: Cálculo de Putnam
Fonte: Putnam (1978).

K= esforço (pessoa/ano)
L= linha de código
C_k= constante de estado de tecnologia
td= tempo de desenvolvimento (anos)

Ck é constante de estado de tecnologia, e conforme o modelo num ambiente de desenvolvimento ‘pobre’ Ck = 2.000. Para um ambiente de desenvolvimento de software ‘bom’ (com aplicação de metodologia, documentações, revisões adequadas e desenvolvimento interativo) Ck = 8.000. Num ambiente de desenvolvimento de software ‘excelente’ (uso de ferramentas e técnicas automatizadas) Ck = 11.000.

3.3.10 Técnicas de estimação voltadas para projetos SOA

Como poderá ser observado no Capítulo 4, de revisão do estado da arte, vários autores têm proposto adaptações de técnicas clássicas (listadas neste capítulo) para estimação de projetos SOA, embora com foco em aspectos isolados destes.

Em uma perspectiva mais ampla, de procurar abarcar vários aspectos de projetos SOA, alguns trabalhos têm proposto algumas técnicas.

Linthicum (2006) elaborou uma fórmula geral para medir o custo de um projeto SOA, expressa na Equação 22:

Cost of SOA = Cost of Data Complexity + Cost of Service Complexity + Cost of Process Complexity + Enabling Technology Solution

Equação 22: Cálculo de Linthicum
Fonte: Linthicum (2006).

Todavia, como se pode observar, ela é bastante abstrata e atua mais como uma lista de aspectos a considerar e não como algo prontamente calculável (ERL, 2009) (SETH; AGRAWAL; SINGLA, 2014).

Já Li e Keung (2010) propuseram um cálculo com o objetivo de estimar o tamanho, custo e esforço para projetos SOA. Neste framework o projeto de SOA é decomposto em serviços menores. Cada serviço é classificado de acordo com seu tipo, por exemplo, de desenvolvimento de serviços, de integração de serviços ou de desenvolvimento de aplicações. Cada tipo de serviço tem suas próprias atividades específicas, modelos, fatores de custo e funções de custo. O custo total do projeto será um somatório do custo dos seus serviços constituintes.

Este framework ainda está sendo desenvolvido e também funciona mais numa perspectiva de diretrizes gerais e não de fórmulas de cálculo propriamente ditas (SETH; AGRAWAL; SINGLA, 2014).

Baseado estudos abordados nos tópicos anteriores foi elaborado uma digrama único que apresenta todos os fatores e suas inter-relações que precisam ser estimados durante a implementação do serviço SOA. Este diagrama representa o método desenvolvido. Os detalhes de como foram abordados e considerados são explicados no Capítulo 5.

3.4 TÉCNICAS MULTICRITÉRIO

No método proposto nesta tese é realizado o ranqueamento das funcionalidades. Ele é baseado nos critérios definidos pelos gestores, onde são dados graus de importância relativos entre os quatro fatores essenciais de decisão SOA no método proposto: custo, tempo de desenvolvimento, esforço e grau de alinhamento ao negócio. Portanto, há mais do que um critério (i.e. multicritério) que, conforme a demanda, diferentes prioridades, restrições ou convenções da empresa, etc., podem ter pesos de importância diferentes entre si.

Uma análise multicritério corretamente realizada permite aos gestores ponderar importâncias nos fatores de decisão com base nas características de cada demanda, otimizando assim a aplicação do método. Dessa forma, este subcapítulo apresenta uma breve revisão da literatura em torno das técnicas multicritérios.

O *Auxílio Multicritério à Decisão* (AMD) é um ramo da Pesquisa Operacional que tem objetivo de fornecer ao gestor/decisor algumas ferramentas para o auxiliar no tratamento de um problema decisório em que vários, e frequentemente contraditórios, critérios e pontos de vista devem ser considerados (FREITAS, 2007).

Nesse contexto, a AMD parte do princípio de que não existe uma alternativa que seja a melhor em todos os critérios. Segundo Roy (1996) e Vincke (1992), as abordagens de AMD podem ser classificadas em:

- *Abordagens de subordinação*: um conjunto de alternativas (A) são valoradas sobre um conjunto de critérios, construindo-se relações de subordinação não compensatórias entre elas.
- *Abordagens interativas*: alternam atividades de cálculo com atividades de decisão, nas quais o analista de decisão interage com o modelo, construindo a decisão mais adequada. Este tipo de abordagem é importante em situações em que se esteja buscando uma única solução que seja ótima ou que esteja próxima do ponto ótimo.

- *Abordagens do critério único de síntese:* buscam uma função que agregue diferentes funções de utilidade em uma função única. Os métodos que se baseiam nesta abordagem são chamados de métodos multicritério. Dentre os métodos que se baseiam nesta abordagem destacam-se: a Teoria da Escolha Social (ARROW, 1963), a Teoria da Utilidade Multiatributo (Multi-Attribute Utility Theory - MAUT) (KEENEY; RAIFFA, (1976), e o método AHP (Analytic Hierarchy Process) (SAATY, 1980).

Dentre os vários métodos de análise multicritério, o método AHP destaca-se por possibilitar a obtenção da importância relativa dos critérios envolvidos em problemas decisórios e a ordenação de tais critérios segundo esta importância. Esse raciocínio constitui a essência do problema de elicitação de requisitos no desenvolvimento de um software. Por isto, nesta tese será utilizado o método AHP.

3.4.1 O Método AHP

Proposto por Saaty no início dos anos 70, o método AHP (*Analytic Hierarchy Process*) tem como objetivo a seleção/escolha de alternativas em um processo decisório que considere múltiplos critérios. Vários autores realçam sua utilidade, como a seguir relatado.

Allen et al. (2008), citam o AHP ao listarem os métodos de priorização de requisitos. Segundo eles, o AHP é um método de apoio à decisão utilizado em situações nas quais múltiplos objetivos estão presentes. O método utiliza a comparação paritária para calcular o valor relativo de cada item (requisito). Os autores destacam a possibilidade de confirmação da consistência dos resultados como um aspecto positivo do método.

Para Karlsson, Wohlin e Regnell (1998), o AHP é bastante confiável visto que a redundância que existe nas comparações par a par reduz a ocorrência de erros de julgamento. Além disso, esta comparação força os usuários a analisar os requisitos de uma perspectiva diferente, possibilitando até mesmo a descoberta de erros na especificação da lista de requisitos.

Karatzas, Dioudi e Moussiopoulos (2003), utilizam o AHP para priorização dos requisitos de um sistema de gerenciamento de qualidade de ar urbano. De acordo com esses autores, estudos anteriores indicaram que fazer julgamentos relativos tende a ser mais rápido e mais confiável do que em termos absolutos. Para eles, o AHP é ideal para resolver

problemas de natureza qualitativa, como é o caso dos requisitos dos usuários.

Karlsson (1996) realizaram um estudo de caso para priorização dos requisitos por meio do AHP. Foram selecionados 14 requisitos para o estudo e 5 pessoas participaram na avaliação. Para os autores, devido ao fato do AHP basear-se em uma escala de razão, a diferença de prioridade entre os requisitos torna-se mais clara. Por exemplo, considerem-se dois requisitos: A, que equivale a 30% da prioridade total, e B, que equivale a 10%. Com isso é possível afirmar que A é 3 vezes mais importante que B. Lee, Joshi e Bae (2008) utilizarão o AHP para definir a prioridade relativa dos requisitos necessários em sistemas web. Para eles, embora o AHP possa ser utilizado em outros contextos, o método se mostra satisfatório para a obtenção das prioridades relativas de requisitos.

Em seu estudo comparativo sobre AHP em tomada de decisão multicritério, Neyra (2008) apresenta como vantagens do AHP sobre outros métodos de análise de decisão multicritério: i) não são métodos muito complexos; ii) fatores qualitativos e quantitativos podem ser utilizados; iii) podem ser utilizados com outros métodos; iv) vêm sendo muito utilizado com base em estudos feitos; e v) auxiliam as múltiplas partes a alcançarem uma solução de consenso. O método AHP está estruturado em três princípios (SAATY, 2001):

- Construção de hierarquias: sistemas complexos podem ser melhor compreendidos através do particionamento deste em elementos menores, estruturando tais elementos hierarquicamente e então sintetizando os julgamentos da importância relativa destes em cada nível da hierarquia em um conjunto de prioridades. A Figura 23 ilustra a estrutura de hierarquias do AHP.

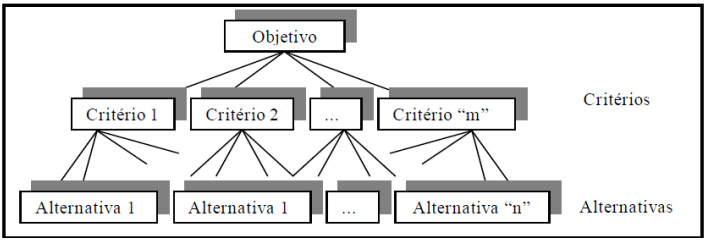


Figura 23: Estrutura Hierárquica Básica
Fonte: Saaty (2001).

- Definição de prioridades: é necessário cumprir basicamente as seguintes atividaes:
  - Julgamentos paritários: comparar par a par os elementos de um nível da hierarquia à luz de cada elemento em conexão em um nível superior, compondo a matriz "A" (Figura 24) (baseado das escalas na Tabela 14). A quantidade de comparações necessárias para a elaboração de uma matriz de julgamentos "A" é n(n-1)/2, onde n é o número de elementos pertencentes a esta matriz. Os elementos da matriz "A" são definidos pelas condições apresentadas na Figura 24
  - Normalização das matrizes de julgamento: obtenção de quadros normalizados através da soma dos elementos de cada coluna das matrizes e posterior divisão de cada elemento pelo somatório dos valores da respectiva coluna.
  - Cálculo das prioridades médias locais (PML): são as médias das linhas dos quadros normalizados;
  - Cálculo das prioridades globais: identifica um vetor de prioridades global (PG) que armazene a prioridade associada a cada alternativa em relação ao foco principal.
- Consistência lógica: Conforme Saaty (2001), o método AHP se propõe a calcular a Razão de Consistência das comparações, denotada por RC = IC/IR, onde IR é o Índice de Consistência Randômico obtido para uma matriz recíproca de ordem n, com elementos não-negativos e gerada randomicamente. O Índice de Consistência (IC) é dado por IC = (λmáx –n)/(n-1), onde λmáx é o maior autovalor da matriz de comparação. Segundo Saaty (2000) a condição de consistência dos julgamentos é RC $\leq$ 0,10.

Tabela 14: Escalas de valor para julgamentos paritários

Escala Verbal	Escala Numérica
Igual preferência (importância)	1
Preferência (importância) fraca	3
Preferência (importância) moderada	5
Preferência (importância) forte	7
Preferência (importância) absoluta	9
2, 4, 6, 8 são associadas a julgamentos intermediários	

Fonte: Saaty (2001).

$$A = \begin{bmatrix} 1 & a_{12} & \cdots & a_{1n} \\ 1/a_{21} & 1 & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ 1/a_{n1} & 1/a_{n2} & \cdots & 1 \end{bmatrix}, \text{ onde:}$$

$$a_{ij} > 0 \Rightarrow positiva$$
$$a_{ij} = 1 \therefore a_{ji} = 1$$
$$a_{ij} = 1/a_{ji} \Rightarrow reciproca$$
$$a_{ik} = a_{ij} \cdot a_{jk} \Rightarrow consistência$$

Figura 24: Condições para o elemento A
Fonte: Saaty (2001).

Segundo Van Den Honert (2001), frequentemente existe grande diferença em termos de formação profissional, competência e experiência no âmbito de um problema num grupo de decisores. Além disto, nem todos têm o mesmo objetivo na análise do problema, os critérios podem ser variados, tornando evidente também que se os decisores não estiverem equitativamente qualificados para contribuir igualmente no processo decisório os resultados podem ser controversos ou mesmo equivocados ao final.

Dyer e Forman (1999) relatam que o AHP pode ser utilizado em 4 contextos:

- Consenso: é aconselhável que estes se reúnam e se esforcem para obter o consenso na estruturação do problema;
- Votação: se o consenso não puder ser obtido em determinada situação, então o grupo de avaliadores pode realizar uma votação ou outro mecanismo acordado;
- Uso de média geométrica: se o consenso não puder ser obtido e os membros não desejarem realizar uma votação, a média geométrica dos julgamentos dos membros pode ser calculada.
- Modelos ou Avaliadores Distintos: se os avaliadores tiverem objetivos muito divergentes, cada avaliador pode emitir seus julgamentos separadamente. Os julgamentos de cada avaliador podem ser obtidos por: Modelos distintos: cada avaliador atribui seus julgamentos em um modelo distinto e as prioridades resultantes podem ser obtidas pelo cálculo da média;

Avaliadores: na estrutura hierárquica é construído um "nível de avaliadores" abaixo do nível do objetivo principal. Os critérios e subcritérios são alocados em um nível abaixo do respectivo avaliador e não necessariamente devem ser os mesmos para todos os avaliadores.

4 REVISÃO DO ESTADO DA ARTE

Este capítulo tem como objetivo apresentar uma revisão do estado da arte em estimação de projetos SOA, área geral abarcada pela proposta desta tese.

Esta revisão faz parte da primeira etapa da metodologia de base deste trabalho *Design Science*, *entendimento do problema*, e teve quatro objetivos:

i. Analisar os mais importantes trabalhos correlatos à tese e identificar o que já existe, as questões com propostas já bem sólidas, os problemas não muito bem cobertos, e as várias abordagens em si de solução ao problema de estimação SOA;
ii. De posse dessa análise, identificar mais claramente as contribuições científicas e o ineditismo da proposta de tese;
iii. Ainda de posse daquela análise, e como parte da estratégia de pesquisa, identificar os artefatos conceituais propostos na literatura que serão inteira ou parcialmente aproveitados no método proposto, de forma a tanto não refazer trabalhos já existentes como focar nos aspectos da contribuição científica, ineditismo da tese e proposição de valor;
iv. Num processo indutivo, e complementar aos estudos descritos no Capítulo 3, atuar como identificador-base para uma generalização dos fatores (e suas terminologias e inter-relações) a considerar no método proposto de estimação de projetos SOA com base nas "instâncias" (trabalhos de vários autores) elicitadas.

Considerando que o principal interesse foi o de obter uma visão alargada dos trabalhos já que se visava uma generalização, o método adotado para tal foi o da revisão sistemática da literatura (SLR - *Systematic Literature Review*) (KITCHENHAM, 2007).

O SLR corresponde a um meio para identificar, avaliar e interpretar as pesquisas disponíveis relevantes dentro de um certo período de tempo para uma questão de pesquisa específica, ou área de tópico, ou fenômeno de interesse.

O SLR envolve três fases principais (Figura 25):

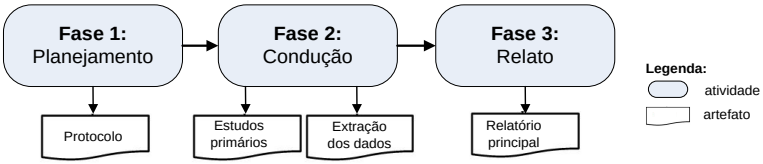


Figura 25: Fases da revisão sistemática de Literatura
Fonte: Kitchenham, 2007.

- *Fase 1 - Planejamento*: nesta fase os objetivos da pesquisa e o protocolo de revisão são definidos. O protocolo constitui um plano pré-determinado que descreve as questões de pesquisa e como a revisão sistemática será realizada.
- *Fase 2 - Condução*: durante esta fase os estudos primários são identificados, selecionados e avaliados de acordo com critérios de exclusão estabelecidos. Para cada estudo selecionado os dados são extraídos e sintetizados.
- *Fase 3 - Relato*: nesta fase um relatório formal é apresentado.

Nas seções seguintes, é descrito como essas fases foram realizadas nesta revisão sistemática.

4.1 PLANEJAMENTO

Inicialmente, nesta etapa é solicitada a identificação da necessidade de uma revisão da literatura. Nesta pesquisa, essa necessidade foi descrita na introdução, no Capítulo 1, que mostrou o cenário atual e a problemática que gerou esta proposta de tese e que norteiam a pergunta da tese:

> *Quais são os fatores a serem considerados num processo de estimação de um projeto SOA, e como tais fatores podem ser organizados de forma a sistematizar o processo de análise e decisão sobre implementar ou não um software na forma de serviços em uma perspectiva SOA ?*

Os parâmetros do protocolo de revisão em que esse SLR foi conduzido e as informações deste protocolo foram especificados pelo autor desta proposta de tese com base na sua experiência e do grupo de pesquisas no qual está inserido:

- tipo de material pesquisado: artigos em revistas, anais de congressos, e teses e dissertações nas áreas da tese*;*
- idioma pesquisado: inglês e português;
- período: 2000 a 2016;
- fontes de pesquisa: repositórios científicos.
  - *IEEExplore (http://ieeexplore.ieee.org/search),*
  - *ACM Digital Library (http://portal.acm.org),*
  - *ScienceDirect - http://fase.sciencedirect.com,*
  - *UFRGS – Tese de doutorado http://www.lume.ufrgs.br/handle/10183/1,*
  - *UFPE – Tese de doutorado http://www.bdtd.ufpe.br/handle/123456789/50,*
  - *USP – Tese de doutorado http://www.teses.usp.br/,*
  - *UFSC / Ciências da computação - http://ppgcc.posgrad.ufsc.br/ – Teses e Dissertações,*
  - *UFSC / Engenharia de Automação e Sistemas - http://pgeas.ufsc.br/ – Teses e Dissertações;*
  - *UFRJ – Tese / Centro de Ciência Matemática e da Natureza-CCMN https://minerva.ufrj.br/F?RN=23564698*
  - *PUC-RJ – teses/dissertações- http://www.ccpa.puc-rio.br/sintetico/relat_externo/gera_relatorio.lua?tp=relatorio_aluno_dtda*
  - *UNIRIO - http://www.seer.unirio.br/index.php*

4.2 CONDUÇÃO DA REVISÃO

Uma vez selecionadas as bases onde as pesquisas seriam executadas, foi necessário adequar a *string* de busca para cada sistema. As *strings* e os critérios de busca atuam nos *metadados* dos repositórios, focando no título, palavras-chave e resumo. As *strings* e os critérios de exclusão estão detalhados no **Apêndice A**.

A busca nos repositórios retornou os resultados (Tabela 15):

Tabela 15: Trabalhos retornados

Fonte	**Trabalhos retornados**
IEEExplore	1998
ACM Digital Library	21
ScienceDirect	44
UFRGS – Tese de doutorado	250

UFPE – Tese de doutorado,	1880
USP – Tese de doutorado	1898
UFSC/ Ciências da Computação– Tese doutorado	2
UFSC/ Engenharia de Automação e sistemas – Tese doutorado	2
UFRJ – Tese / Centro de Ciência Matemática e da Natureza-CCMN	6
PUC-RJ – teses/dissertações	5
UNIRIO	8

Fonte: Autor.

Após ter acesso às fontes foi realizada a seleção dos estudos primários. Para isso, todos os 6114 trabalhos foram acessados e verificados se seriam excluídos desse SLR (de acordo com os critérios de exclusão), ou se fariam parte dos trabalhos relacionados a esta proposta de tese de doutorado.

As bases de pesquisa dos trabalhos ordenam os resultados de acordo com sua relevância (baseados na *string* de busca). Por esse motivo, os 1950 primeiros trabalhos tiveram o *abstract* lido por este autor para verificar a sua relevância nesta proposta de tese. A partir disso, os títulos já deixavam claro se o trabalho se enquadrava no contexto desta tese. Desta forma, 4145 trabalhos foram excluídos apenas a partir da leitura dos seus títulos.

Por fim, 85 artigos foram completamente lidos. Após análise e refinamento, gerou-se uma lista de 15 trabalhos considerados relevantes (de acordo com a proposta deste doutorado), apresentados na

Tabela 16 e detalhados a seguir.

Importante enfatizar que esses trabalhos *complementam* o referencial teórico (descrito no Capítulo 3) usado nesta tese para a concepção do método de estimação proposto.

Tabela 16: Trabalhos relevantes

Titulo	**Referência**
A Framework for Scope, Cost and Effort Estimation for Service Oriented Architecture (SOA) Projects	(O'Brien, 2009)
Service Oriented Computing in Practice – An Agenda for Research into the Factors Influencing the Organizational Adoption of Service Oriented Architectures	(Luthria e Rabhi, 2009)

Adopting SOA – Experiences from nine Finnish organizations	(Kokko, Antikainen e Systa, 2009)
Analyzing the Reuse Potential of Migrating Legacy Components to a Service-Oriented Architecture	(Lewis, Morris e Smith, 2006)
Managing the QoS of E-Government: Metrics for Large Scale SOA	(Monsalve, 2007)
The Role of Service Granularity in A Successful SOA Realization – A Case Study	(Kulkarni e Dwivedi, 2008)
Estudo Empírico sobre Adoção de SOA: Um Mapeamento Sistemático da Literatura	(Dias, Oliveira e Meira, 2013)
A Method for Bridging the Gap between Business Process Models and Services	(Azevedo et al., 2013)
An Approach for a more Agile BPM&SOA Integration supported by Dynamic Services Discovery	(Perin e Rabelo, 2010)
Service Identification Approach to SOA Development	(Fareghzadeh, 2008)
To Establish Enterprise Service Model from Enterprise Business Model	(Jamshidi, Sharifi e Mansour, 2008)
Identification and Analysis of Business and Software Services - A Consolidated Approach	(Kohlborn et al., 2009)
Guideline for Sizing Service-Oriented Architecture Software	(Fagg et al., 2010)
Functional Size, Effort and Cost of the SOA Projects with Function Points	(Gomes e Pereira, 2012)
Service point estimation model for SOA based projects	(Gupta, 2013)

Fonte: Autor.

No trabalho de **O´Brien (2009)** foi proposto um modelo chamado SMAT-AUS, representando uma abordagem inicial para o desenvolvimento de uma estrutura para capturar o que é necessário em escopo, dimensionamento, custo e estimativa de esforço para diferentes tipos de projetos SOA, mostrado na Figura 26.

A figura ilustra em forma cubo o modelo, onde as dimensões apontam todos os fatores que teriam impacto no custo e esforço, incluindo os aspectos técnicos e sociais / culturais / organizacionais bem como a maturidade da organização em termos de sua experiência na realização de Projetos SOA.

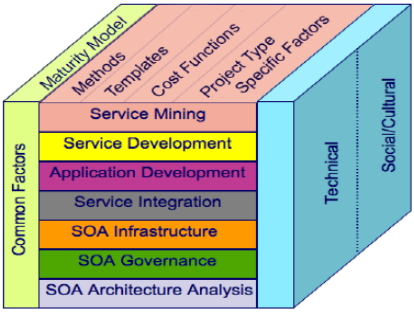


Figura 26: Modelo SMAT-AUS
Fonte: O´Brien (2009).

Inicialmente, o autor define tipos de projetos de SOA. Define também os serviços existentes, os sistemas legados, o estado de SOA desejado. Daí surge uma relação de necessidades que se torna base para se desenvolver a estratégia da implementação SOA. O SMAT-AUS é alimentado com informações visando ajudar os usuários a documentar certas informações sobre seus serviços. Além disto, algumas outras considerações para projetos de SOA são discutidas, incluindo custos de desenvolvimento do serviço, custo de terceirização do fornecimento de serviços e custo das operações.

Apesar da grande contribuição do SMAT-AUS, a forma da mensuração descrita é bastante superficial / abstrata, não contribuindo diretamente para a concepção das fórmulas de cada fator do método proposto nesta tese.

Por outro lado, dos sete fatores identificados, seis deles inspiraram os fatores do modelo de estimação proposto nesta tese: desenvolvimento dos serviços, desenvolvimento das aplicações, integração de serviços, infraestrutura SOA, governança SOA, e análise da arquitetura SOA. A não consideração do fator Identificação de serviço tem a ver com o fato de que atua em ação num projeto SOA fora do escopo do objetivo desta tese.

No trabalho de **Luthria e Rabhi (2009)** os autores identificaram inicialmente sete aspectos gerais relevantes de considerar em projetos SOA, a saber: características do SOA; arquitetura de funcionamento; infraestrutura necessária; visão empresarial do SOA; TICs usadas em SOA; melhores práticas em SOA; modelos de negócio do SOA.

Apesar do levantamento desses pontos, os autores se aprofundaram apenas nos mais ligados às estratégias e gestão de negócios, identificando dentro destes também sete aspectos gerais relevantes de considerar: valor do SOA para a organização; estratégia organizacional; cultura organizacional; processos organizacionais; riscos; governança; e gerenciamento de SOA.

Assim como no modelo SMAT-AUS, a forma da mensuração descrita é bastante superficial / abstrata, não contribuindo diretamente para a concepção das fórmulas de cada fator do método proposto nesta tese. Por outro lado, o trabalho trouxe importantes contribuições na definição dos fatores do método proposto bem como alguns elementos a considerar dentro de alguns desses fatores, a saber:

- Inclusão do uso de abordagens baseadas em ESB como elemento a ser avaliado no fator "Condição Ambiental";
- A possibilidade do serviço ter integração com consumidor externo, contribuindo com fator "Complexidade de Integração"; e a inclusão do item "integridade" no fator "Requisitos Não Funcionais";
- A consideração dos elementos BPM, indicadores de desempenho e metas no fator "Aspectos de Negócio";
- A consideração de se estimar os serviços não apenas isoladamente, mas também de forma composta;
- A consideração da experiência dos profissionais envolvidos num projeto SOA no fator "Experiência Profissional";
- A consideração de alguns elementos de qualidade de serviços, traduzidos na forma / expressos no fator "Requisitos Não Funcionais" assim como a inclusão do elemento "conformidade" no fator "Aspectos Legais";
- A constatação de que aspectos de governança interferem num projeto SOA, sendo considerado então no fator "Condição Ambiental" do método desenvolvido nesta tese.

O trabalho de **Kokko, Antikainen e Systa (2009**) traz uma pesquisa em nove organizações finlandesas, concentrando-se no processo da adoção de SOA por elas, analisando o começo e a evolução deste processo. O estudo aponta os desafios enfrentados e apresenta melhores práticas na adoção da SOA nestas organizações. No trabalho, os autores apontaram os métodos a serem utilizados na identificação de serviços ao migrar para SOA e também as ferramentas para especificação dos serviços. Os autores procuraram responder perguntas em relação aos benefícios e aos desafios enfrentados durante a adoção

da SOA, melhores métodos de especificação de serviços SOA, o papel da modelagem de processos de negócios e da governança SOA.

Assim como os dois trabalhos anteriores, o trabalho do Kokko, Antikainen e Systa (2009) não visara identificar concretamente os fatores de decisão numa estimação em SOA. Por outro lado, o trabalho contribuiu na consolidação dos fatores do método em termos de:

- A constatação da importância do fator "tamanho" e sua influência no reuso;
- A constatação da influência do "conhecimento dos profissionais" na implantação da SOA, considerada no fator "Experiência de Equipe" no método proposto;
- A constatação da influência dos aspectos Processo de Negócio, Governança de SOA e ferramentas de desenvolvimento para o fator "Aspecto Negócio", e a inclusão dos aspectos governança e ferramentas no fator "Condição Ambiental".

Lewis, Morris e Smith (2006) focaram na importância da consideração dos sistemas legados em projetos SOA, apresentando um estudo de caso referente à migração de componentes de software ligados aos serviços e à reutilização destes serviços. O artigo descreve várias questões a serem abordadas nesse contexto SOA, sob três perspectivas: do consumidor do serviço, da arquitetura e do provedor do serviço.

Do ponto de vista de estimação, a grande contribuição deste trabalho foi a de melhor elucidar os aspectos a considerar quando da (usual) existência de sistemas legados na arquitetura SOA de uma empresa. Porém, a estimação final em si é bastante subjetiva e um tanto superficial. Já para as atividades de *análise de gap* e de *especificação das necessidades* (Figura 1), os autores utilizaram o método *SMART*, que reúne informações sobre legados e os envolvidos serviços SOA para produzir uma estratégia de desenvolvimento de produto de software.

Por outro lado, o trabalho trouxe importantes contribuições para o método proposto nesta tese:

- Identificação de vários aspectos técnicos relacionados à questão da invocação de serviços. Isto fomentou a inclusão dos itens 1, 2 e 3 no fator "Requisitos Funcionais";
- Identificação de vários características técnicas, como linguagens de descrição de serviços e mecanismos de descoberta de serviços. Com isso foi constatado que esses aspectos têm influência na implementação de serviços. Desta forma, foram incluídos como elementos no fator "Condição Ambiental".

- Identificaram características gerais de serviços que acabaram por apoiar na definição dos fatores “Tamanho” e “Complexidade”; na inclusão dos elementos “fraco acoplamento” e ‘interoperabilidade” no fator “Requisitos Não Funcionais”; na inclusão do elemento “estabilidade dos requisitos” no fator “Condição Ambiental”; e na inclusão do elemento “objetivo da adoção da SOA em relação aos objetivos de negócio” no fator “Aspecto Negócio”.

Monsalve (2007) propôs um conjunto de métricas para avaliação dos aspectos operacionais dos sistemas SOA com base em critérios técnicos e econômicos tendo como premissa de que a gestão de projeto e sua eficácia depende de indicadores de desempenho.

No contexto dos objetivos desta tese, a principal limitação deste trabalho é que os autores definiram métricas para serviços já desenvolvidos e em execução, e não sobre como deveriam ser consideras e calculadas quando ainda do projeto dos serviços e, assim, como poderiam impactar no custo e tempo geral do projeto.

Por outro lado, o trabalho contribuiu para o método proposto nesta tese em termos de:

- Ao analisar que serviços, quando postos em execução, podem ter variadas demandas de uso/acesso de serviços e taxas de comunicação de dados, fez com que se passasse a considerar concorrência, latência de rede e taxa da transferência de dados no fator “Requisitos Não Funcionais”.
- De forma análoga, aqueles aspectos chamaram a atenção para a influência da infraestrutura no funcionamento de serviços, fortalecendo a necessidade da inclusão dos elementos “ferramentas e plataformas de desenvolvimento” e “ESB” no fator “Condição Ambiental”.

Kulkarni e Dwivedi (2008) pesquisaram sobre regras que facilitam a definição da granularidade ou tamanho dos serviços identificados, uma questão de extrema importância em SOA. O trabalho foi baseado em um estudo de caso de instituição financeira, cujo acervo de ativos de software era constituído por milhares de *web services*. Uma série de questões foram apontadas sobre como e ao mesmo tempo dificuldades de gerenciá-los, mantê-los e integrá-los, considerando que um serviço tem que oferecer benefícios para a organização.

Tais benefícios são mensurados na forma de: retorno de investimento, reusabilidade, complexidade técnica para migrar ou

integrar sistema legado com serviço, e complexidade técnica para implementar o serviço.

O trabalho não propõe nenhuma técnica ou fórmula de cálculo desses aspectos mencionados, mas serviu para reflexões sobre o que considerar e calcular no fator "Reuso" do método proposto nesta tese.

Dias, Oliveira e Meira (2013) analisaram trabalhos sobre a adoção de SOA na indústria, identificando aspectos gerais, como atividades realizadas, vantagens, lições aprendidas, desafios e fatores que influenciam a sua adoção, independentemente de terem tido ou não sucesso. Um conjunto de 27 trabalhos, sendo 21 estudos de caso, foram analisados. Os autores consideraram que SOA é uma estratégia que precisa de um estrutura organizacional, pessoas, processos e tecnologias de fluxo de trabalho. Como resultado da pesquisa, foram identificados vários fatores a serem considerados num processo decisório de adoção de SOA, quais sejam: BPM, complexidade tecnológica, tamanho dos projetos e seus custos, governança SOA, estrutura organizacional, experiência da equipe, treinamento e motivação, reuso, esforço e segurança.

Este trabalho também não propôs nenhuma técnica ou fórmula de cálculo desses aspectos mencionados, mas serviu para reflexões sobre o que considerar nos diversos fatores do método proposto / processo decisório em si de se usar ou não serviços/SOA numa dada solução.

O trabalho do **Azevedo *et al*. (2013)** apresenta um estudo de caso com aplicação de métodos para identificação e análise de serviços em um ciclo de vida SOA a partir de modelos de processos de negócio modelados no nível de BPM. Eles demonstram que os modelos de processos de negócio podem auxiliar na implantação de uma abordagem SOA, pois contêm uma série de informações sobre as atividades realizadas pela organização. Essas informações podem ser utilizadas na identificação de serviços a serem implementados através de heurísticas de identificação e análise.

O trabalho ofereceu contribuições no processo da identificação dos serviços candidatos, ilustrando as atividades necessárias e também os fatores que interferem na priorização na implementação de cada serviço. Desta forma, contribuíram nas reflexões sobre o que considerar e como calcular os fatores "Reuso" e "Tamanho" do método. Porém, o trabalho não considera os requisitos funcionais e não funcionais dos sistemas a serem gerados assim como não considera a complexidade ambiental para implementação de cada uma das funcionalidades.

Perin e Rabelo (2010) propuseram um modelo de identificação automática dos serviços através da integração dos níveis SOA e BPM. Foi desenvolvido um mecanismo de descoberta dinâmica de serviços, abrangendo ecossistemas de provedores de serviços que usariam um catálogo eletrônico de processos de negócios (modelado no padrão UBL) (*Universal Business Language* - Oasis, 2006) e acessados como SaaS. A descoberta considera os requisitos funcionais e não funcionais, QoS e contextos dos processos para a identificação dos serviços.

Na pesquisa os autores apontaram quatro dimensões que apoiam na descoberta de serviços alinhados ao negócio: recuperação de informações (semântica para proporcionar maior precisão na seleção de serviços), arquitetura SOA, QoS (métricas, framework, estratégias) e normas (governança e interoperabilidade).

Este trabalho não propõem nenhuma técnica ou fórmula de cálculo desses aspectos mencionados e nem como um gestor deve decidir ou não por implementar algo como serviços. Por outro lado, serviu para reflexões sobre como considerar os elementos "interoperabilidade", "segurança", "disponibilidade" e "escalabilidade" no fator "Requisitos Não Funcionais" além de reforçar a importância de se considerar questões de BPM no método proposto.

Fareghzadeh (2008) propôs uma abordagem de identificação seletiva de serviços com base em análise de processos de negócios nos casos de uso, na análise de ativos existentes, e na lista dos serviços candidatos. Baseado em vários modelos da literatura, este trabalho enfatiza as principais atividades para a análise e identificação de serviços em projetos SOA, mencionando algumas práticas concretas recomendadas e atividades para uma correta identificação do serviço.

Apesar de importante pela "completude" das etapas, o trabalho basicamente identifica quais modelos podem ser usados em algumas das atividades, deixando suas escolhas completamente a cargo dos gestores e assim com grande subjetividade e dificuldade de padronizar processos e normalizar resultados finais da estimação. Ainda, não apresenta os princípios a serem utilizados na análise do serviço candidato para fins de decisão se este deve ou não ser implementado como serviço.

Por outro lado, contribuiu nas reflexões sobre o que considerar no fator "Complexidade Integração" e atores envolvidos bem como demonstrou a possibilidade de aquelas etapas serem transformadas em um processo (método) de tomada de decisão.

Jamshidi, Sharifi e Mansour (2008) propuseram um processo para identificação e especificação de serviços a partir de modelos de negócio. Os conceitos, princípios, elementos de modelo e diretrizes para a construção de soluções orientadas a serviços a partir da modelagem de processos de negócio contribuem para a definição dos fatores envolvidos no processo de implantação SOA.

A proposta é, entretanto, apresentada de forma genérica e sem um exemplo concreto, o que dificulta o seu entendimento de como usá-lo na prática. Por outro lado, o trabalho trouxe contribuições que ajudaram na identificação do que considerar nos fatores "Reuso", "Tamanho", "Aspectos Negócio" assim como nos elementos *fraco acoplamento* e *interoperabilidade* no fator "Requisitos Não Funcionais" do método.

Kohlborn *et al.* (2009) realizaram um estudo sobre diversos métodos de identificação de serviços, apontando seus pontos fortes e fracos. O trabalho foi realizado baseado em critérios gerais definidos, como visão SOA, estratégia de implementação de SOA, cobertura do ciclo de vida, grau de prescrição, acessibilidade e validade. Baseado no resultado obtido, os autores propuseram um método geral que contempla todos os requisitos associados àqueles critérios.

De forma resumida, tais requisitos convergem para os aspectos de custo, requisitos não funcionais, contexto de negócio e objetivo do negócio, porém sem oferecer suporte de como são calculados. Como tais aspectos são contemplados nos fatores do método desenvolvido nesta tese, tais requisitos serviram para refinar os elementos a serem considerados nas fórmulas de cada fator.

Em relação a estimação de Serviço SOA, o guideline de estimativa COSMIC oferece algumas orientações concretas neste sentido **(FAGG et al. 2010).** Os autores criaram diretrizes para estimar tamanho, custo e esforço de implementação serviço SOA. O ponto importante do trabalho é a importância da estimação na definição, se é mais vantajosa comprar, alugar ou desenvolver internamente uma aplicação. O guideline aborda 4 fatores importantes neste processo de estimação: objetivo e escopo da estimação, a identificação dos usuários e o nível de granularidade do serviço.

As principais contribuições deste guideline no método proposto nesta tese foram o reforço e aspectos do que considerar na identificação dos fatores Custo, Esforço e Tamanho. Além disto, o trabalho dos autores trouxe contribuições na identificação dos itens 1 a 9 do fator Requisitos Funcionais.

A principal limitação do trabalho dos autores se refere ao de estimar um serviço unicamente baseado em requisitos funcionais.

Gomes e Pereira (2012) apresentam uma proposta detalhada de como mapear os elementos para posterior estimação de projetos de SOA baseado em Pontos de Função.

Este trabalho contribuiu para se identificar o fator "Complexidade de Integração" como necessário no método proposto e deu algumas bases de como mensurar a integração dos serviços com diferentes tipos de atores. No entanto, não considera outras questões importantes, como requisitos de QoS esperados e os aspectos ambientais do projeto.

Gupta (2013) elaborou um amplo estudo sobre o processo de estimação de serviços em SOA levando em consideração os aspectos funcionais e ambientais das técnicas de Caso de Uso e Ponto de Função.

Este trabalho contribuiu para se identificar os fatores (e alguns dos seus elementos) "Requisitos Funcionais", "Requisitos Não Funcionais" e "Complexidade Ambiental". Porém, pode-se dizer que as principais limitações do trabalho de Gupta (2013) frente aos objetivos desta tese são de que ele não oferece nenhum "processo" de estimação e que considera apenas a perspectiva tecnológica de projetos SOA, e não também, por exemplo, a de negócios.

4.3 RELATÓRIO DA REVISÃO

Após se efetuar o SLR foram encontrados poucos trabalhos que se possam dizer realmente equivalentes à proposta desta tese. Na verdade, não foi encontrado nenhum trabalho que especificamente tratasse de um método decisório sistematizado e que envolvesse "todas" as principais dimensões de análise num único "modelo" aplicado no processo de estimação de SOA.

Os vários trabalhos analisados tratam das experiências dos gestores e projetistas de software com observações empíricas e análises teóricas gerais ou de estudos de caso de implantações SOA. Apenas dois trabalhos foram encontrados que de alguma forma tentaram dar um viés mais metodológico à atividade de estimação, o de O'Brien (2009) e Kokko, Antikainen e Systa (2009). Porém, são descritos de forma genérica e os processos não são detalhados, não abordando nem os processos e nem os critérios de decisão em si. Alguns trabalhos se

mostraram enquadrados em algumas das etapas do método proposto desta tese, porém funcionam apenas como guias gerais.

Após este passo metodológico de revisão de estado da arte, dos workshops realizados nas empresas de TI, dos retornos de revisores de artigos, e da análise dos artefatos teóricos de estimação existentes na literatura (descritos no Capítulo 3), identificaram-se os quinze fatores de decisão como os mais adequados de serem considerados num processo decisório sobre SOA. Esses fatores e o "mapa mental" da fundamentação de cada um desses fatores são descritos no Capítulo 5.

Além destes fatores decisórios, essa análise bibliográfica também serviu para comparar os benefícios trazidos por cada um desses trabalhos do estado da arte frente à proposição de valor que o método desta tese pretende trazer, listadas na Seção 1.5.

Neste sentido, a

Tabela 17 busca sintetizar esse comparativo, embora, como já frisado, nenhum deles objetivou dar um enfoque metodológico de decisão SOA, foco desta tese. Ressalta-se que a proposta de tese não se encontra, portanto, "comparada" nessa tabela, que visa apenas explicitar as questões de proposição de valor.

Tabela 17: Tabela comparativa das características dos trabalhos

BENEFICIO / **TRABALHOS**	Reuse	Comparar custo SOA *x* Tradicional	Avaliar desenvolvimento composto	Priorização no processo de desenvolvimento	Ponderar importâncias nos fatores	Qualidade das Decisões	Rapidez nas Decisões	Sistematização do processo	Apoio na decisão na aquisição, subcontratações ou terceirizações de software.
A Framework for Scope, Cost and Effort Estimation for Service Oriented Architecture (SOA) Projects	☆☆					☆		☆☆	☆

Service Oriented Computing in Practice – An Agenda for Research into the Factors Influencing the Organizational Adoption of SOA					★★	★★	★★	★★	
Adopting SOA – Experiences from nine Finnish organizations					★	★	★	★	
Analyzing the Reuse Potential of Migrating Legacy Components to SOA projects	★★		★	★				★	
Managing the QoS of E-Government: Metrics for Large Scale SOA		★	★	★	★				★★
The Role of Service Granularity in A Successful SOA Realization – A Case Study	★		★		★		★	★	
Estudo Empírico sobre Adoção de SOA: Um Mapeamento Sistemático da Literatura	★	★		★	★	★	★	★	
A Method for Bridging the Gap between Business Process Models and Services	★★		★★	★★	★★			★★	
An Approach for a more Agile BPM&SOA Integration supported by Dynamic Services Discovery				★			★		
Service Identification Approach to SOA Development								★★	
To Establish Enterprise Service Model from Enterprise Business Model	★★			★				★	
Identification and Analysis of Business and Software Services - A Consolidated Approach			★	★★	★	★★	★	★★	
Guideline for Sizing Service-Oriented Architecture Software			★				★	★★	
Functional Size, Effort and Cost of the SOA Projects	★			★	★			★★	

with Function Points									
Service point estimation model for SOA	★	★		★	★★	★	★	★★	

Fonte: Autor.

Legenda:

Assunto não abordado	=	Campo vazio
Assunto pouco detalhado	=	★
Assunto abordados detalhadamente	=	★★

5 MÉTODO PARA ESTIMAÇÃO SOA

Este capítulo visa apresentar o método de apoio à decisão na estimação de software baseado em serviços, em uma perspectiva SOA, que representa o objetivo geral desta tese. A sua elaboração corresponde à etapa 3, de *Desenvolvimento*, do método de pesquisa *Design Science* adotado neste trabalho (ver Capítulo 2).

Para essa apresentação, este capítulo está organizado em quatro seções. A Seção 5.1 resume a "lógica" do método, sua delimitação e condicionantes. A Seção 5.2 explicita as fundamentações teóricas que foram utilizadas no método, identificando também as fontes de conhecimento que contribuíram para se chegar a seu modelo. Isto está alinhado ao processo indutivo de pesquisa adotado nesta tese, que buscou criar um método essencialmente fazendo uma generalização de diversos estudos e depois os adaptando ao cenário desejado. A Seção 5.3 apresenta o método propriamente dito, enquadrando-o dentro das atividades de *Análise* do ciclo de vida SOA, que é o escopo do método. A Seção 5.4 ilustra mais precisamente onde e de que forma os vários aspectos de SOA foram considerados em cada fator de estimação. A Seção 5.5 apresenta o resumo das contribuições do método.

Esta apresentação é totalmente conceitual. A sua exemplificação é mostrada no Capítulo 6.

5.1 LÓGICA GERAL DO MÉTODO

Como explicado no Capítulo 2, a estratégia de pesquisa foi *incremental*, de forma que o método desenvolvido fora concebido tendo como base fundamentações teóricas de engenharia de software já consolidadas e contribuições de outros trabalhos (compiladas nos Capítulos 3 e 4), adaptadas para um cenário de estimação SOA desejado.

O método é constituído por 15 fatores, que foram elicitados, inter-relacionados e organizados numa sequência de passos – na forma de um processo sistematizado dentro de uma taxonomia – a serem percorridos por um gestor de TI quando necessita fazer uma estimação de software. Cada fator tem uma fórmula, composta por uma série de variáveis relacionadas ao contexto do fator. O gestor de TI alimenta os fatores com certas informações da empresa e da solução pretendida a medida que percorre o método. Cada fator calcula um valor associado. O método sintetiza os valores dos fatores e "converge" no sentido de

prover ao final números concretos para custos, esforço, tempo de desenvolvimento e grau de alinhamento ao negócio para cada funcionalidade candidata oriunda da atividade de análise de gap.

Suscintamente, o método recebe como entrada uma lista de funcionalidades a serem implementadas (realizadas nas atividades de *identificação de serviços* e *análise de gap* dentro da etapa de *Análise* do ciclo SOA), podendo ser expressa como *Casos de Uso* ou qualquer outro tipo de artefato. Com base nisso, o método faz uma análise sobre as funcionalidades considerando os 15 fatores e ao final gera valores comparativos entre custos gerais se elas forem ser desenvolvidas como serviços de software e como software tradicional (*análise da realização*).

Adicionalmente, as prioridades de negócio da empresa são consideradas no cálculo (*ranqueamento*), o que é feito através do uso do método multicritério *AHP*. Com base nesses cálculos e comparativos o gestor de TI toma a decisão final sobre quais funcionalidades se recomenda desenvolver como serviços e quais como software tradicional (*decisão*). Uma vez implementadas, estas tornam-se *ativos de software* da empresa, i.e. artefatos prontos para serem usados nos processos de negócio.

Portanto, este é o objetivo primordial do método desenvolvido: prover informações rigorosas e quantitativas de estimação ao gestor para ajudá-lo naquela decisão.

Usuários do método de estimação incluem, tanto *softwarehouses* que (pelo menos também) adotam SOA, como empresas em geral que tenham equipes de desenvolvimento (próprias ou terceirizadas) e que também sejam usuárias de soluções baseadas em serviços para a execução dos seus processos de negócio.

No que se refere às atividads de *identificação de serviços* e *análise de gap*, ressalta-se que não é relevante para este método de estimação a abordagem ou técnica em si adotadas para se chegar àquela lista de funcionalidades. O essencial é que de “alguma forma” a lista chegue como entrada para o método de estimação poder ser iniciado / realizado.

O método foi concebido de forma alinhada tanto ao ciclo de vida como ao ciclo de governança SOA, além de envolver as estratégias de negócio da organização. Porém, como já realçado, o seu escopo funcional dá-se dentro da etapa de *Análise*.

Como mencionado no capítulo 1 e considerando a potencial complexidade e escopo de uma solução SOA, são vários os papéis que podem ser envolvidos no uso do método. Tomando como base os atores

e papéis usuais em engenharia de software (PAPAZAGLOU, 2012), eles seriam:

- **Cliente**: Expor necessidades do negócio;
- **Analista de Negócio**: Entender as necessidades do negócio, auxiliar na modelagem do negócio;
- **Analista de Requisitos**: Realizar o levantamento dos requisitos de software;
- **Arquiteto de Solução**: Entender as necessidades do negócio incluindo aspectos de, por exemplo, integração de soluções, reutilização de serviços, procurando identificar os serviços candidatos;
- **Arquiteto de Software**: Elaborar o projeto de aplicações de forma a viabilizar serviços de aplicação e a facilitar a composição destes em serviços de negócio;
- **Gestor**: Analisar o alinhamento da solução com a estratégia de negócio;
- **Gerente de Projetos**: Gerir o projeto de SOA;
- **Desenvolvedor**: Implementar a solução.

O principal objetivo da etapa de Análise é entender e avaliar as necessidades para o desenvolvimento de software. Conforme Lewis, Morris e Smith (2010), existem "gatilhos" para iniciar a etapa de análise no contexto de SOA/serviços de software:

- Desenvolvimento de um novo serviço;
- Desenvolvimento de um novo projeto SOA;
- Evolução e melhorias de serviço existente;
- Novos consumidores de serviços;
- Evolução e mudança em processos de negócio;
- Mudanças no modelo de negócio;
- Mudanças em sistemas legados que impliquem em mudanças nos serviços existentes;
- Adaptações de sistemas legados à arquitetura SOA;
- Mudanças na infraestrutura SOA.

Além desses citados, um outro tipo de gatilho percebido se refere a demandas oriundas de prospecções de mercado e inovações onde se deseje, por exemplo, fazer prototipações e estimativas de custos gerais. Isso é importante em cenários de inovação SOA e de inovação colaborativa em software, onde variados níveis de filtros de análise de

viabilidade são aplicados ao longo do processo de inovação (SANTANNA *et al.*, 2016).

Conforme Papazaglou (2012), a etapa de análise é subdividida nas atividades de "identificação do serviço", "análise de *gap*" e "análise de realização". Segundo Lewis, Smith e Kontogiannis (2010), a atividade de "identificação do serviço", quando se considera SOA, deve se basear em análises que considerem a estratégia da tecnologia da empresa, a sua estratégia e seus modelos de negócios, os seus processos em si de negócio ("BPM"), os requisitos destes processos e respectivos atores, e seus ativos em termos de sistemas legados existentes. No caso do método desenvolvido, nada é feito referente às atividades de "identificação do serviço" e "análise de *gap*". Como dito anteriormente, o método inicia tendo como entrada o resultado dessas atividades.

Tradicionalmente, o processo de estimação propriamente dito é todo executado na atividade de realização. Porém, desejou-se nesta tese melhor explicitar que o método proposto acaba por estender aquelas atividades na medida que, após a estimação (ao que se chamou de "medição"), há ainda um processo de análise multicritério que considera os aspectos de negócio e um comparativo entre custos gerais de implementação SOA / não-SOA (ao que se chamou de "ranqueamento"). Além disto, que ao final há um mais amplo processo de decisão diante deste ranqueamento, envolvendo gestão de projetos e recursos gerais disponíveis na empresa, e que ainda se pode usar o método para fazer algumas simulações de alternativas de decisão.

Com isto, as etapas do método proposto englobam (Figura 27):

- Quatro atividades (identificação de serviço e análise *gap*, análise da realização, ranqueamento e decisão), que são compostas por subatividades que detalham as etapas e artefatos envolvidos no desenvolvimento SOA. Eles são: "análise técnica", "medição", "técnicas multicritério" e "análise frente aos recursos disponíveis. Eles tem objetivo de refinar a seleção dos serviços até apontar quais funcionalidades-candidatas podem se tornar ativos.

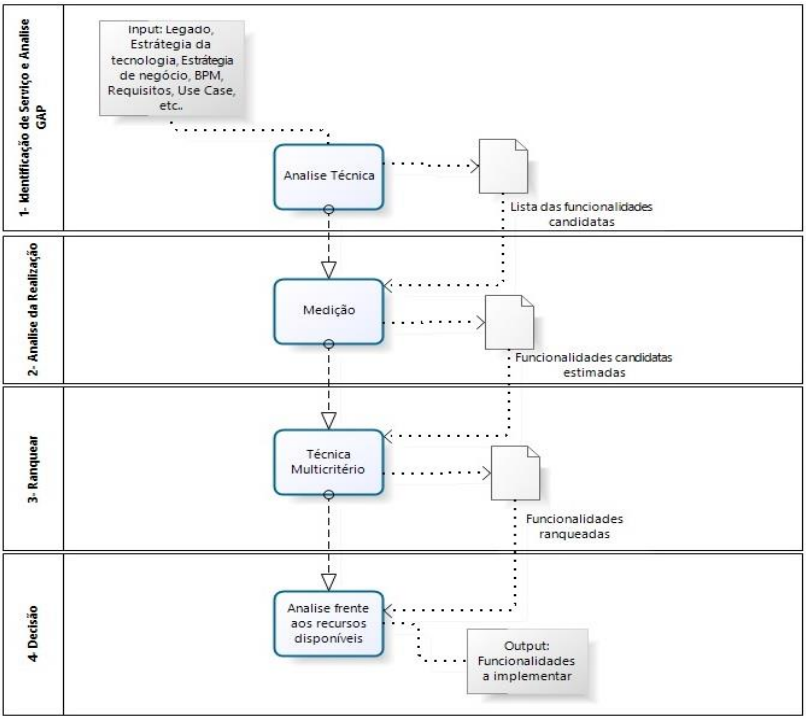


Figura 27: Sequência das atividades
Fonte: Autor.

5.2 FUNDAMENTAÇÃO TEÓRICA DE BASE

Como mencionado na seção anterior, cada um dos 15 fatores do método tem uma forma de cálculo. Estes fatores foram elicitados com base em inúmeros trabalhos (Capítulos 3 e 4) bem como junto a algumas empresas (ver Capítulo 7).

A

Figura 28 explicita mais especificamente a origem *principal* de cada fator e seus elementos, ao que se denominou de “mapa mental”. Nela, quando é mencionado "Adaptado de ...", se refere a quando o trabalho foi intensamente utilizado mas adaptado em alguns aspectos para o cenário SOA desejado e de comparação com um não-SOA. Quando escrito "Extraído parcialmente ...", se refere a quando apenas alguns elementos do trabalho foram aproveitados no método.

Os estudos gerais sobre os mais variados aspectos envolvidos numa análise SOA (explanados no Capítulo 3) também trouxeram complementações além daqueles trabalhos mencionados na

Figura 28. Abaixo são descritas suscintamente essas complementações, que são detalhadas na próxima seção:

- **Governança SOA:** o estudo sobre governança SOA serviu para identificar os itens que têm influência na avaliação da aderência do serviço avaliado perante necessidades organizacionais. Mais especificamente, o impacto da governança no fator experiência dos profissionais, no fator condição ambiental, e em todos os aspectos do fator aspectos de negócio. Em termos gerais, dado que governança influencia basicamente todas atividades de uma empresa e do setor de TI, tentou-se sempre verificar este aspecto em todos os fatores.
- **Estratégia SOA e Necessidades do Negócio (Börner & Goeken):** este trabalho serviu como base para identificar aspectos mais estratégicos a serem avaliados em projetos SOA.
- **Engenharia de software - QoS e SLA:** o estudo sobres SLAs contribuiu para identificar os fatores não funcionais que tem influência na escolha de um serviço.
- **Estimação de Software - Estimativa por analogia**: entendeu-se que esta técnica não se aplica ao método proposto. Isto porque verifica-se que não é comum haver necessidade de desenvolvimento de novos serviços similares aos já existentes.
- **Estimação de Software - Opinião especializada**: esta técnica teve contribuição no método proposto no sentido de tempo e de custo do desenvolvimento de software. Porem, devido a ser muito subjetiva, considerou-a como não muito adequada para o método proposto, pois este visa justamente diminuir a subjetividade na estimação. Essa técnica evidencia que é importante levar em consideração as experiências dos envolvidos no processo de implementação de serviço SOA, caso de envolver especialistas no momento de ponderar os pesos e valores dos fatores dos requisitos funcionais e não funcionais.

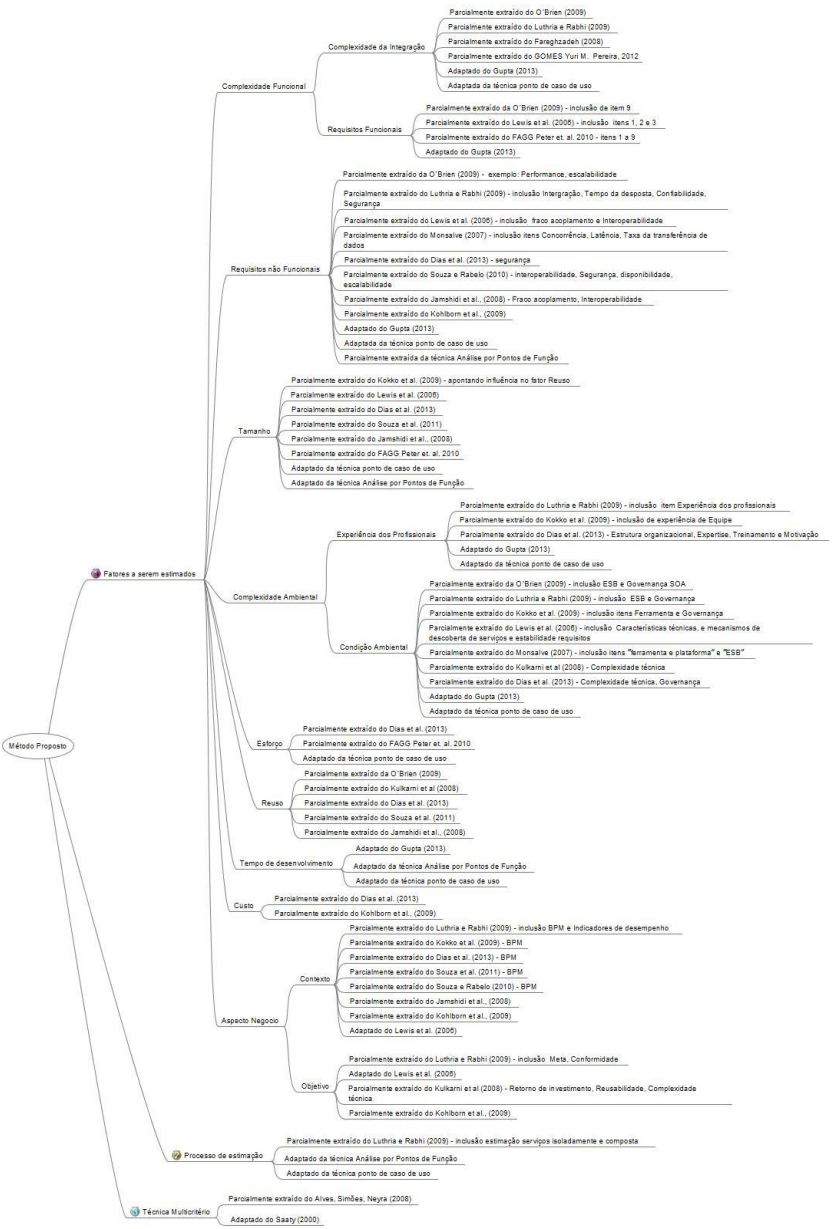


Figura 28: “Mapa mental” do método
Fonte: Autor

- **Estimação de Software - Estimativa de três pontos ou PERT**: esta técnica tem sido bastante aplicada na estimação.
- **Estimação de Software - Linhas de código**: esta técnica permitiu identificar os fatores e a inter-relação entre esforço, tamanho (quantidade de linhas de código) e experiência profissional, o que contribuiu para a definição da taxonomia do método proposto (Figura 29).
- **Estimação de Software - COCOMO II**: do modelo COCOMO II aproveitou-se o conhecimento de que a estimação do tempo de desenvolvimento é baseado no cálculo do tamanho do código e no esforço a ser dispendido (Figura 29). Porém, não se aproveitou do COCOMO II as suas fórmulas de estimação. Isto porque ele não é muito adequado para estimar projetos SOA uma vez que não considera, por exemplo, a capacidade de reutilização de serviços SOA (SETH; AGRAWAL; SINGLA, 2014) a despeito de haverem trabalhos que têm procurado adaptar o COCOMO II para SOA (TANSEY; STROULIA, 2007).
- **Estimação de Software - Análise por Pontos de Função (APF)**: o estudo da técnica Ponto de Função teve importantes contribuições na identificação dos fatores a serem estimados no método proposto. Um dos aspectos foi de se adotar 1 (um) como valor do fator de ajuste; se for igual a 1 a influência total das características gerais do sistema é neutra. Nesta situação, a contagem dos pontos de função ajustados equivale à contagem de pontos de função não ajustados. A Figura 30 ilustra os fatores envolvidos e suas inter-relações na estimação de software utilizando APF.

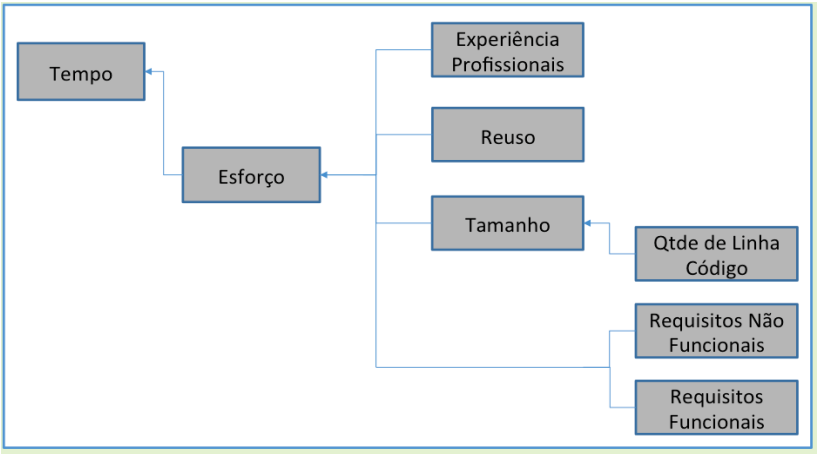


Figura 29: Relações entre fatores no COCOMO II
Fonte: Autor.

Nas figuras a direção das setas representam as dependências dos fatores. Por exemplo, o fator "tempo" depende do fator "esforço" que por sua depende dos fatores "experiência Profissionais", "Reuso", "Tamanho", "Requisitos não funcionais" e "Requisitos funcionais".

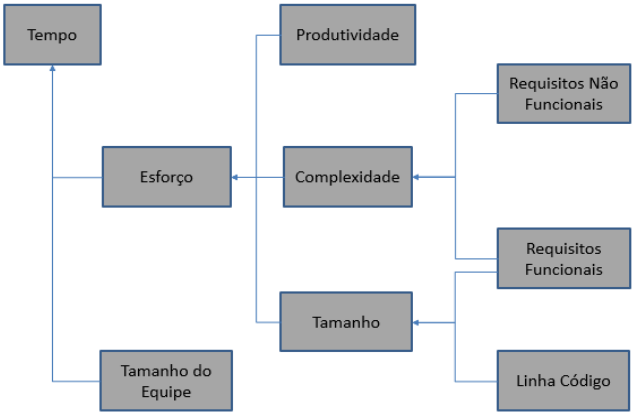


Figura 30: Fatores baseado APF
Fonte: Autor.

- **Estimação de Software - Pontos de Caso de Uso:** da mesma forma que no caso da técnica Análise de Ponto de Função, a Pontos de Caso de Uso teve grandes importantes contribuições na identificação e na inter-relação dos fatores a serem estimados no método proposto, resultando no que mostra a Figura 31.

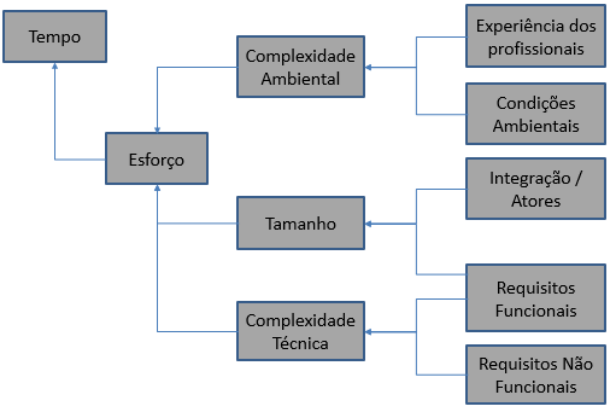


Figura 31: Fatores envolvidos na perspectiva Pontos de Caso de Uso
Fonte: Autor, Adaptado de Sommerville (2011); Pressman (2016)

- **Estimação de Software – PUTNAM:** este trabalho influenciou a organização do fator esforço e suas inter-relação com outros fatores, como mostra a Figura 32.

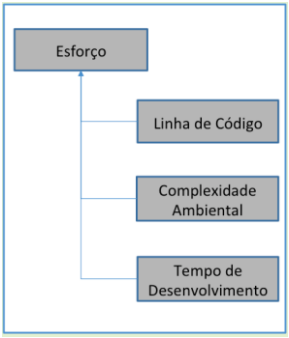


Figura 32: Fatores para calcular Esforço
Fonte: Autor.

- **Técnicas Multicriterio:** Neste tese será utilizado o método AHP, conforme mencionado e justificado no capítulo 3.

Considerando essa fundamentação teórica de base utilizada para o método, a Figura 33 mostra em termos gerais onde as diversas principais contribuições da literatura impactaram no modelo (‘taxonomia’) do método de estimação desenvolvido.

Os métodos tradicionais, de forma geral, estimam os fatores esforço, tempo, tamanho e custo de desenvolvimento de software, não levando em consideração os fatores que são envolvidos nas implementações SOA. Além desses fatores (e das adaptações nas suas formas de cálculo), foi incluído o fator de negócio, não encontrado de forma muito concreta em outros trabalhos.

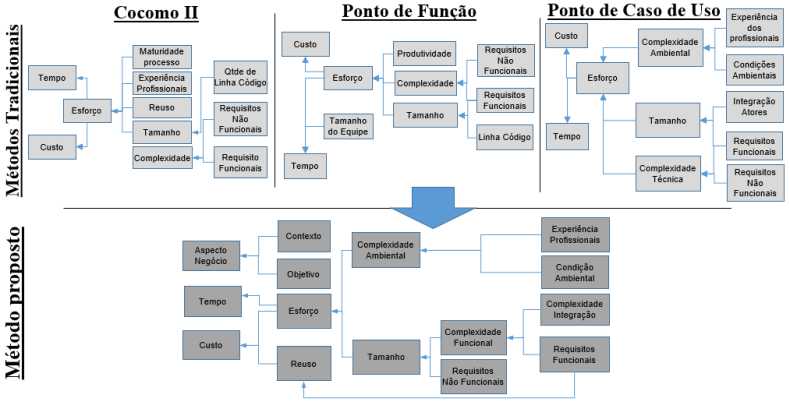


Figura 33: Contribuições principais da literatura para o método
Fonte: Autor

Como explanado no Capítulo 3, as métricas de software podem ser divididas em dois tipos: Métricas Orientadas à Função e Métricas Orientadas ao Tamanho. Nas métricas orientadas à função, a métrica se baseia nas funcionalidades do software (PRESSMAN, 2016). No caso das métricas orientadas ao tamanho, a medida é baseada na contagem de linhas de código.

Apesar de métricas orientadas a tamanho se mostrarem um modelo interessante, elas não são aceitas universalmente devido ao fato de haver diferença entre a quantidade de linhas de código de diferentes tipos de linguagens de programação, como também explanado no Capítulo 3 (PRESSMAN, 2016).

Dado a isso, as métricas orientadas ao tamanho foram descartadas, posto que no início do projeto não há uma definição do tamanho e isto aumenta a subjetividade da medição. Assim sendo, métricas orientadas à função foram escolhidas para o método, considerando os métodos de estimativa clássicos mais difundidos e consolidados, como Ponto de Função e Pontos de Caso de Uso. Além de poderem dar pelo menos parcialmente condições de medição de SOA (SANTILLO, 2007), elas fornecem uma estimativa de software na etapa inicial do desenvolvimento de software (PEREIRA et al. 2007).

Considerando que SOA tem várias características que diferem do software tradicional (mencionado anteriormente), a aplicação dos métodos tradicionais precisa de adaptações (KRAFZIG; BANKE; SLAMA, 2004). Como analisado no Capítulo 4, os trabalhos de Lewis (2009) e Farrag, Moawad (2014) foram outros que serviram de base para algumas das adaptações introduzidas nesta tese para o cenário de estimação SOA desejado.

Uma razão adicional para se optar por tentar usar métodos/técnicas "clássicas" de engenharia de software tem também a ver com um dos objetivos do método, que é o de fazer um comparativo com os custos "gerais" obtidos com uma implementação SOA versus com uma não-SOA. Desta forma, considera-se que a análise pelo gestor de TI fica facilitada dado que ele estará ponderando artefatos e conceitos clássicos com os quais tenha já um bom conhecimento ou pelo menos alguma familiaridade.

Análise de composição de serviços na estimação

O método assume que a implementação como serviço é apenas uma possibilidade (PRESSMAN, 2016) e que serviços podem ser implementados em diversas tecnologias (e.g. *web services*) e projetados para interoperar com outros serviços e softwares em geral (ERL 2009).

Na linha geral considerada em Erl (2009), o método desenvolvido parte do princípio que a medição deve considerar tanto um serviço de modo isolado como de modo agregado, de maior granularidade, para posterior composição. Esta agregação nesta tese é portanto considerada como a possibilidade (na análise / estimação) de "x" funcionalidades-candidatas serem implementadas como apenas um (1) serviço (de granularidade "x") e não como "x" serviços (de granularidade 1). O mesmo ocorre com a análise de software tradicional, onde uma daquelas funcionalidades-candidatas poderiam ser implementadas de forma tradicional (como uma pequena função) ou encapsuladas num mesmo

projeto de software, onde cada funcionalidade-candidata se tornaria uma funcionalidade do software tradicional.

As complexidades adicionais associadas a uma agregação (por exemplo, maiores dificuldades de integração interna entre as funcionalidades) deverão ser analisadas durante cada estimação, atribuindo devidamente valores às variáveis das fórmulas dos fatores de acordo com isso.

Independentemente da abordagem sugerida pelo método, as funcionalidades-candidatas são tratadas da seguinte forma na ótica dos quatro fatores básicos de estimação do método proposto:

- **Esforço:** Como este fator representa o esforço necessário para desenvolver cada funcionalidade, o esforço de composição de serviços ou funcionalidades será o somatório dos esforços das funcionalidades e serviços individualmente.
- **Custo:** Este fator representa o valor gasto para desenvolvimento de cada funcionalidade. Neste sentido, o custo do serviço ou funcionalidade composta será o somatório dos custos das funcionalidades e serviços isolados.
- **Tempo:** Como este fator representa o tempo de desenvolvimento de cada funcionalidade, o tempo de desenvolvimento do serviço ou funcionalidade composta será o somatório dos tempos de desenvolvimento das funcionalidades e serviços isolados.
- **Alinhamento de Negócio:** Este fator aponta como cada serviço ou funcionalidade está alinhada com os aspectos de negócio da empresa. Neste caso, como o aspecto de negócio de um serviço composto pode não ter relação com o aspecto de negócio dos serviços ou funcionalidades isoladas, para identificar aspectos de negócio de um serviço ou funcionalidade composto, o responsável precisa novamente atribuir pesos para itens deste fator para este serviço ou funcionalidade composto.

Sobre a subjetividade na medição

Uma questão sempre presente nas discussões sobre medições tem a ver com a subjetividade que pode existir, tanto quando do fornecimento de informações pelo gestor de TI da empresa para o método de estimação, quanto quando da interpretação dos resultados em si da estimação gerados (FUJITA, 2016; KAPUR et al., 2014; SVELTO

et al., 2016). Mesmo trabalhos que aplicam abordagens de otimização, como Kitchenham e Pfleeger (2002;2003), reconhecem que a interpretação final também tem uma parcela subjetiva de análise e decisão pelo gestor de TI.

Uma das abordagens que mais vêm sendo aplicadas para minimizar esse problema é na forma de tabulação de informações em escalas, onde o usuário pode expressar o que deseja dentro de limites em faixas de uma escala. Isto tem a ver também que para certos cálculos e análises um determinado número detalhado concreto em si não é essencial de ser conhecido, mas apenas uma "faixa" de valores.

Nesse sentido, a *Escala de Likert* é uma técnica das mais usadas (GLIEM; GLIEM, 2003), enquadrando faixas de valores dentro de categorias. As faixas e valores da escala são variáveis, mas tem sido muito comum se usar uma escala numérica de 0 a 5, o que é também seguido por este método de estimação desenvolvido. Assim, em grande parte das "tabelas de estimação" usadas no método proposto quando do fornecimento de informações pelo gestor de TI, inspirou-se no chamado "fator de influência", proposto em Sommerville (2011) (Tabela 18).

Tabela 18: Fator de influência

Grau	Fator de influência
0	Sem influência
1	Baixo
2	Moderado
3	Média
4	Alto
5	Forte

Essa questão da subjetividade é levantada aqui também para, de certa forma, "justificar" como aspectos mais subjetivos são considerados em certos fatores do método; mais especificamente, os aspectos de governança de TI e conformidade. Por exemplo, como avaliar o nível de governança em uma empresa? É fato que existem modelos de maturidade de governança que demonstram como fazer essa avaliação (ARSANJANI; GHOSH; ALLAM, 2008); (BROWN; LAIRD; CLIVE, 2009) (LEUSSE; DIMITRAKOS; BROSSARD, 2009) (KUPPURAJU, KUMAR, 2008), mas que também não calculam um valor em si de governança mas "apenas" o nível/categoria de governança que tem. Neste sentido, optou-se por utilizar escalas de avaliação para o gestor de TI no cálculo de alguns fatores, nomeadamente os fatores "Condição Ambiental" e "Aspecto de Negócio".

5.3 O MÉTODO DE ESTIMAÇÃO

Esta seção apresenta o método em suas quatro atividades, conforme Figura 27, a serem explanadas nas respectivas subseções a seguir

5.3.1 Atividade "Identificação do Serviço e Análise Gap"

As atividades "identificação de serviço" e "análise *gap*" têm como objetivo identificar os serviços candidatos através de certos métodos considerando os ativos de software e sistemas legados já existentes nos repositórios da empresa.

Conforme já abordado na Seção 5.1, a atividade de identificação de serviço recebe como entrada um conjunto de artefatos (estratégias de tecnologia e negócios, processos, requisitos e ativos de software) gerando uma lista de serviços candidatos como resultado. Tal lista corresponde, na realidade, à lista de funcionalidades necessárias de serem implementadas, seja como serviços/SOA, seja como software 'tradicional'.

Na atividade de identificação de serviço são consideradas a possibilidade de reuso e o eventual aproveitamento e adaptação de legados. Esta ação está alinhada ao processo de engenharia de requisitos, focado no aspecto da rastreabilidade. Segundo Sommerville (2007), um requisito é rastreável se é possível descobrir de onde ele surgiu (a fonte), porque o requisito existe (razão) e quais outros requisitos estão relacionados a ele (dependência). A rastreabilidade já é uma prática recomendada por diferentes normas e padrões, tais como CMMI (SEI, 2010), ISO/IEC-15504 (ISO, 2008) e MPS.BR (SOFTEX, 2013).

Conforme Pressman (2016) e Sommerville (2011), a identificação dos requisitos é baseada no levantamento e definição das necessidades da empresa. A necessidade identificada tem o objetivo de garantir o acompanhamento, a documentação das funcionalidades e a explicitação dos requisitos do software.

Na atividade de Análise de *Gap* a lista é reavaliada levando-se em consideração o repositório dos serviços e legados para verificar se existe legado que possa ser total ou parcialmente reusado para atender às necessidades das funcionalidades identificadas como necessárias de serem desenvolvidas. Se for identificado que já existe(m) serviço(s) que atende(m) completamente aos requisitos, será feito um reuso; se for

identificado que o serviço existente atende apenas parcialmente, isto será avaliado na atividade “analise da realização”.

5.3.2 Atividade “Análise da Realização”

O objetivo desta atividade é identificar as funcionalidades candidatas que podem se tornar ativos. Para isso o gestor executa a ação de “medição”. No caso do método proposto, significa basicamente “passar” cada uma das funcionalidades-candidatas pelos passos do método calculando os vários fatores envolvidos na estimação.

Como explicado na Seção 5.2, foram elicitados e inter-relacionados 15 fatores. Estes compõem o método desenvolvido e são expressos na forma de uma taxonomia (Figura 34).

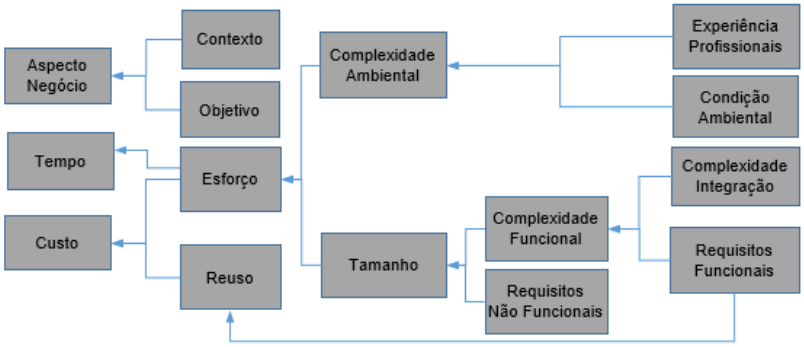


Figura 34: Taxonomia do Método de Estimação
Fonte: Autor.

O método é executado como um processo "*bottom-up*", a partir do lado direito para o lado esquerdo, iniciando por qualquer um dos quatro fatores à direita (e.g. o de requisitos funcionais). Ao final um valor de esforço, custos, tempo de desenvolvimento e nível de alinhamento com negócio é calculado para cada uma das funcionalidades-candidatas. A direção das setas indica a ligação de dependência entre os fatores. Por exemplo, o fator cálculo de complexidade ambiental depende diretamente dos valores calculados dos fatores condição ambiental e experiência profissionais.

A seguir cada um dos fatores do método é explicado, não apenas descrevendo como ele é calculado, mas também as principais fundamentações teóricas de base para cada um deles quando

considerado pertinente de ser complementado frente ao já exposto no Capítulo 3.

A sequência de apresentação dos fatores segue a taxonomia do modelo de estimação, ou seja, reflete basicamente a sequência que o gestor de TI forneceria as informações para o método.

Importante esclarecer que alguns fatores podem ficar com valor "zero". Como explanado até então, a adoção de SOA é um processo gradual e nem todas as empresas têm na sua cultura e processos de estimação vigentes a consideração de todos os fatores necessários para uma "completa" estimação SOA. Isto significa que alguns fatores para algumas empresas podem "não ter como" ser especificados já que não são contemplados nos seus modelos. Quando isso ocorrer, o gestor de TI pode atribuir zero ao fator em questão. Por outro lado, isso implica que todas as estimações da empresa devam adotar a mesma "convenção" sobre quais fatores devem ou não ser considerados de forma a se ter análises comparativas equivalentes entre diferentes estimações.

Ainda neste sentido do atribuir um valor zero a um dado fator, isso pode eventualmente também fazer sentido quando se desejar realizar uma estimação apenas "grosseira" ou numa dada estimação em particular, onde um dado fator – ou mesmo algum aspecto deste (i.e. uma dada variável na sua fórmula) – pode ser irrelevante ou inadequado de ser considerado. Apesar de ser uma opção permitida pelo método, o atribuir zero a algum fator ou aspecto exige igualmente experiência do gestor do TI.

5.3.3 Fatores a Serem Estimados

As seções a seguir descrevem cada um dos fatores da taxonomia.

5.3.3.1 Fator "Complexidade da Integração"

A integração dos serviços e componentes de um sistema SOA num ambiente heterogêneo pode ser muito complexa, gerando impactos negativos na qualidade geral dos sistemas e viabilidade tecnológica (RABELO; NORAN; BERNUS, 2015). Neste sentido os atores que tem interação com os serviços a serem desenvolvidos precisam ser identificados e estimados. A técnica Ponto de Caso de Uso ou *Use Case Poin*t é uma das técnicas mais utilizadas para estimar a complexidade da integração no processo de desenvolvimento de software (O'BRIEN; MERSON; BASS, 2007).

A técnica "*Use Case Point*" define uma classificação em três níveis de complexidade e seus respectivos pesos conforme mencionado anteriormente: 1-simples (para comunicação padronizada; por exemplo através de uma API), 2-média (quando a comunicação é realizada com um sistema externo por meio de um protocolo consolidado), e 3-complexos (geralmente envolvendo usuário final, via de regra por uma interface gráfica) (Tabela 19).

Tabela 19: Tipo de atores

Ator	Peso	Valor	Resultado
Simples	1		
Médio	2		
Complexo	3		
		Total	

A coluna *valor* representa a quantidade de ator(es) para cada classificação. A coluna *resultado* é obtida através da multiplicação da coluna pelo valor do *peso*. O valor total do fator Complexidade de Integração *FCI* é o resultado do somatório da coluna *resultado*.

$$FCI = \Sigma\, Resultado$$

Equação 23: Cálculo de Somatório

*5.3.3.2 **Fator "Requisitos Funcionais"***

Os requisitos funcionais (RF) representam informações sobre como o sistema deve se comportar e o que o sistema deve fazer (SOMMERVILLE, 2011), que no todo são expressas na dimensão complexidade. Estes aspectos também facilitam a identificação das funcionalidades comuns para potencial reutilização. Conforme Chung e Leite et al. (2009), RF é utilizado no método de estimação desenvolvido para ajudar a definir a *complexidade* e o *tamanho* da funcionalidade. Estes dois outros fatores são descritos nas próximas subseções.

De acordo com os levantamentos do estado da arte bem como os trabalhos de Gomes e Pereira (2012) e Gupta (2013), dez parâmetros foram identificados para medir a complexidade de RF.

Seguindo esses trabalhos, o valor de cada parâmetro é expresso dentro de uma gama de três possibilidades. Da mesma maneira do que

os outros fatores, estes parâmetros também podem ter um peso de 0 a 5 (ver Tabela 18). A Tabela 20 ilustra os itens calculados para identificar complexidade de RF.

O resultado final deste fator oferece as bases para o cálculo do tamanho da funcionalidade a ser implementada.

Tabela 20: Complexidade requisitos funcionais

1- Tamanho da mensagem da requisição (Request Message Size): (Dataset = Número de conjuntos de informações que são trocadas pela chamada de serviço. Por exemplo: ordem de serviço).				
Item	**Descrição**	**Valor (1-3)**	**Peso (0-5)**	**Resultado**
1	Dataset <= 1			
2	2<= Dataset <= 3			
3	Dataset > 3			
2- Tamanho da mensagem da resposta (Response Message Size): (Dataset = Número de conjuntos de informações retornadas como resultado durante a chamada de serviço).				
Item	**Descrição**	**Valor (1-3)**	**Peso (0-5)**	**Resultado**
1	Dataset <= 1			
2	2<= Dataset <= 3			
3	Dataset > 3			
3- Complexidade de tratamento de dados na requisição/resposta (Request Response Data Translation Complexity): Tratamento de formato e semântica: a complexidade do tratamento do formato e da semântica que deve ser aplicado antes da requisição ou da sua resposta.				
Item	**Descrição**	**Valor (1-3)**	**Peso (0-5)**	**Resultado**
1	Não há necessidade de tratamento de dados			
2	Tratamento de dados apenas em um sentido (requisição ou resposta)			
3	Tratamento de dados em dois sentidos (requisição e resposta)			
4- Complexidade de lógica de negócio (Core Business Logic Complexity): Complexidade da especificação.				
Item	**Descrição**	**Valor (1-3)**	**Peso (0-5)**	**Resultado**
1	Simples			

2	Media			
3	Complexo			

5- O número de objetos manipulados pela lógica: Pode ser diferente do número de objetos de parâmetros, uma vez que a aplicação pode utilizar objetos internos aos quais tem acesso para complementar sua lógica. Ex: produto, cliente.

Item	Descrição	Valor (1-3)	Peso (0-5)	Resultado
1	Objetos <= 2			
2	3<= Objetos <= 5			
3	Objetos > 5			

6- Complexidade de acesso aos dados

Item	Descrição	Valor (1-3)	Peso (0-5)	Resultado
1	Não há necessidade de gravar dados (somente leitura)			
2	Grava dados ou somente leitura			
3	Precisa executar várias operações de leitura e gravação de dados			

7- Complexidade da integração com outros serviços: Aplicado em casos onde são chamados outros serviços.

Item	Descrição	Valor (1-3)	Peso (0-5)	Resultado
1	Não tem interação com outros serviços (service atômico)			
2	Necessita realizer a busca de serviço a ser chamado atraves de séries			
3	Necessita de chamado de outro serviço transacional			

8- Complexidade de tratamento da falha e do erro

Item	Descrição	Valor (1-3)	Peso (0-5)	Resultado
1	Não há necessidade de tratamento			
2	Erros devem ser relatados dentro do escopo do serviço			
3	Erros devem ser reportados via interface externo			

9- Complexidade da infraestrutura da integração

Item	Descrição	Valor (1-3)	Peso (0-5)	Resultado

1	Através de API bem definida			
2	Através de API com tecnologias abertas como JMS e Apache CXF.			
3	Através de interface proprietária do software			
10-	**Quantidade de campos envolvidos**			
Item	**Descrição**	**Valor**	**Peso (0-5)**	**Resultado**
7	1<= quantidade <= 19			
10	20<= quantidade <= 50			
15	50<= quantidade			

A coluna *resultado* é calculada através da multiplicação da coluna *peso* por *valor*, conforme Equação 24.

$$FRF = \Sigma\, Resultado$$

Equação 24: Requisitos Funcionais
Fonte: Gupta (2013)

Novos requisitos funcionais podem ser adicionais e outros podem ser retirados (ou mesmo desconsiderados ao se atribuir o valor zero a eles). O método é flexível para considerar os requisitos ora em questão.

*5.3.3.3 **Fator "Complexidade Funcional"***

Conforme Kulkarni e Dwivedi (2008), a complexidade funcional de um serviço é o atributo que permite medir a facilidade da implementação de um software (ou parte dele), e tem um impacto no tamanho da funcionalidade.

Neste método presume-se que existe uma relação entre a complexidade funcional e o grau de dificuldade de interpretação, especificação e implementação de um dado conjunto de requisitos em relação à quantidade de suas variáveis e procedimentos (ROSTAMPOUR *et. al.*, 2011).

Como pode ser observado na

Figura 35, para estimação da complexidade funcional devem ser antes calculados os fatores de complexidade da integração e de requisitos funcionais. A

Figura 35 ilustra o uso das mesmas técnicas de estimação adotadas na estimação de complexidade ambiental para definir os aspectos a serem estimados para identificar complexidade funcional.

Um trabalho-chave para a identificação do que considerar nesta análise de complexidade foi o de Grupta (2013). Este trabalho tem foco em aplicações baseado em SOA e foi adicionado o item 10 (Tabela 21), extraído do método Ponto de Função, que tem o objetivo de mensurar a quantidade de campos manipulados no uso da funcionalidade. Estes itens permitem a aplicação do método proposto tanto no desenvolvimento baseado em SOA como no de desenvolvimento tradicional. No caso do fator Complexidade SOA também foram utilizados itens adotados na técnica Ponto de Caso de Uso devido a ter maior abrangência nas suas aplicações. A Tabela 21 e a Tabela 22 apresentam o resultado final dos fatores e seus itens a serem avaliados na implementação das funcionalidade independentemente do modo (SOA ou tradicional).

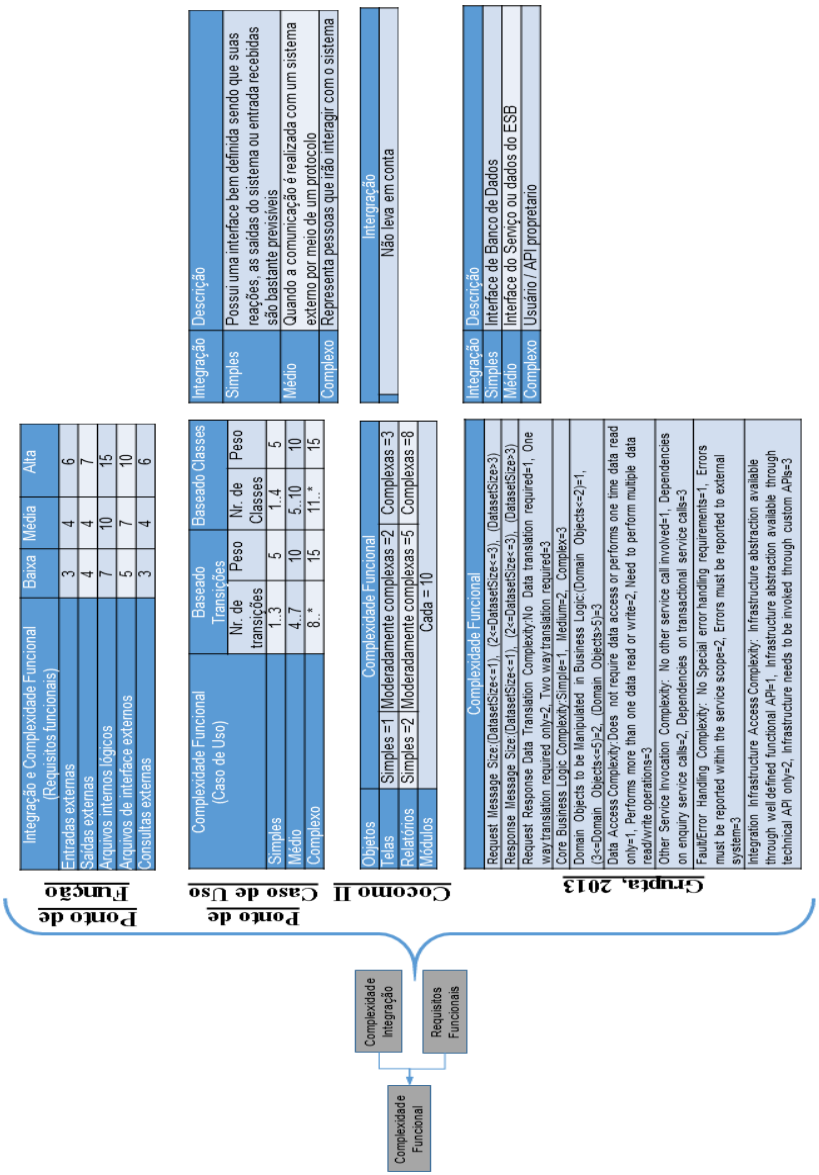


Figura 35: Complexidade Funcional
Fonte: Autor

Tabela 21: Requisitos Funcionais

1- Tamanho da mensagem da requisição		
Item	**Descrição**	**Valor(1-3)**
1	Dataset <= 1	
2	2<= Dataset <= 3	
3	Dataset > 3	
2- Tamanho da mensagem da resposta		
Item	**Descrição**	**Valor(1-3)**
1	Dataset <= 1	
2	2<= Dataset <= 3	
3	Dataset > 3	
3- Complexidade de tratamento de dados na requisição/resposta		
Item	**Descrição**	**Valor(1-3)**
1	Não há necessidade de tratamento de dados	
2	Tratamento de dados apenas em um sentido (requisição ou resposta)	
3	Tratamento de dados em dois sentidos (requisição e resposta)	
4- Complexidade de lógica de negócio		
Item	**Descrição**	**Valor(1-3)**
1	Simples	
2	Média	
3	Complexo	
5- O número de objetos manipulados pela lógica		
Item	**Descrição**	**Valor(1-3)**
1	Objetos <= 2	
2	3<= Objetos <= 5	
3	Objetos > 5	
6- Complexidade de acesso a dados		
Item	**Descrição**	**Valor(1-3)**
1	Não há necessidade de gravar dados (somente leitura)	
2	Grava dados ou somente leitura	
3	Precisa executar várias operações de leitura e gravação de dados	
7- Complexidade da integração com outros serviços		
Item	**Descrição**	**Valor(1-3)**
1	Não tem interação com outros serviços (serviço atômico)	
2	Necessita realizar a busca de serviço a ser chamado através de séries	

3	Necessita de chamado de outro serviço transacional	
8- Complexidade de tratamento da falha e do erro		
Item	**Descrição**	**Valor(1-3)**
1	Não há necessidade de tratamento	
2	Erros devem ser relatados dentro do escopo do serviço	
3	Erros devem ser reportados via interface externo	
9- Complexidade da infraestrutura da integração		
Item	**Descrição**	**Valor(1-3)**
1	Através API bem definido	
2	Através API técnico , exemplo: JMS, Apache CXF API.	
3	Atraves de interface específico do produto	
10- Quantidade de campos envolvidos		
Item	**Descrição**	**Valor(1-3)**
1	1<=Quantidade<= 19=7	
2	20<= Quantidade <= 50=10	
3	50<= Quantidade =15	

Tabela 22: Integração

Ator	Peso	Valor	Resultado
Simples	1		
Médio	2		
Complexo	3		

Para calcular o fator Complexidade Funcional do serviço são somados o fator RF (FRF), que é calculado baseado nos requisitos funcionais, e Complexidade de Integração - CI dos atores envolvidos no funcionamento do serviço (KARNER, 1993).

Complexidade funcional (FCF) = FRF + FCI

FRF = Fator Requisitos Funcionais
FCI = Fator Complexidade da Integração

Equação 25: Cálculo da complexidade
Fonte: Rostampour *et. al.* (2011)

*5.3.3.4 **Fator "Requisitos não Funcionais"***

Requisitos Não-Funcionais (RNF) definem restrições do sistema em aspectos qualitativos, como desempenho, segurança e usabilidade. A complexidade dos requisitos não funcionais afeta complexidade funcional (CLELAND; SETTIMI, 2007).

A Tabela 23 ilustra a lista dos atributos da RNF a serem avaliados no processo do desenvolvimento SOA, num total de 18.

Tabela 23: Requisitos Não Funcionais

Nome	**Descrição**	**Valor (0-5)**
Tempo de resposta	O tempo que demora para retornar a resposta	
Concorrência	O número de solicitações simultâneas	
Escalabilidade	A capacidade de receber cargas de trabalho variadas sem deixar de atender a todos os seus requisitos.	
Segurança	Nível de segurança em termos de confidencialidade	
Monitoramento de serviço	Ferramenta de monitoramento	
Latência	O tempo necessário para iniciar o atendimento de uma requisição de serviço	
Taxa da transferência de dados	Taxa de transferência de dados	
Disponibilidade	O tempo que a aplicação está operando	
Confiabilidade	A aplicação deve funcionar continuamente sem falha	
Precisão/Acurácia	Taxa de erros num intervalo do tempo	
Robustez	O nível de flexibilidade da aplicação para entradas incorretas	
Estabilidade	Capacidade de evitar efeitos inesperados decorrentes das alterações no interface	
Mensagens Confiáveis	A aplicação oferece mecanismos confiáveis de garantia de entrega de mensagens	
Integridade	Capacidade de manter a integridade transacional em cenários de alta concorrência	

Interoperabilidade	Capacidade do interagir com outros sistemas sem perdas ou distorções de informação	
Fraco acoplamento	O quanto a aplicação, depende de, ou está relacionada a, outras implementações	
Abstração	Necessidade de esconder quanto possível os detalhes da implementação	
Sem estado	Não manter informações	
Total (TRNF)		

O cálculo do fator ajustado RNF é mostrado na Equação 26 e é baseada no cálculo de Ponto de Função. O valor 0.01 é o Coeficiente por Grau de Influência e 0.65 é uma constante empírica que pode alterar a contagem de pontos de função numa amplitude de –35% até +35%. O valor final do fator de ajuste pode variar de 0,65 até 1,35. A descrição dos mesmos foi apresentada no Capítulo 3.

$$FRNF = (TRNF * 0.01) + 0.65$$

Equação 26: Cálculo de ajuste
Fonte: IFPUG (2010)

A equação acima é calculada com base na quantidade de 14 RNFs. No entanto, o método proposto considera 18 RNFs (Tabela 23) e assim a fórmula precisa ser adaptada não apenas para 18, mas para qualquer número de RNF e que continue garantindo a amplitude –35% até +35%). Isto permite que quando há necessidade de incluir mais RNFs ou até transformar um RNF "macro" em vários RNFs detalhados, a fórmula garanta a adequada amplitude conforme o método Ponto de Função. Para tal alterou-se o coeficiente (0.01) conforme a quantidade de RNFs, transformando a fórmula da equação 26 em variáveis, conforme abaixo:

0.01 = Coef,
QRNF = Qtde RNF = 18 no método proposto,
VRNF = Peso Máximo que pode atribuir para cada RNF = 5.

Com isso a fórmula da equação 26 para amplitude máxima (valor 1.35) passa a ser:

$$FRNF = ((QRNF * VRNF) * Coef) + 0.65 = 1.35$$

Equação 27: Cálculo de ajuste flexível
Fonte: Autor

Pode-se assim transformar a fórmula em:

$$\text{Coef} = \frac{1.35-0.65}{QRNF*VRNF} => \frac{0.70}{QRNF*VRNF}$$

Equação 28: Cálculo do Coeficiente
Fonte: Autor

Portanto, para utilizar a fórmula padrão de Ponto de Função para quantidades diferentes de RNFs, inicialmente é necessário calcular o coeficiente baseado na quantidade dos RNF. Após encontrar o valor do coeficiente pode-se utilizar a fórmula tradicional de ajuste de RNF do método Ponto de Função, substituindo o coeficiente 0.01 pelo valor encontrado.

No caso da Tabela 23, que tem 18 RNFs, o cálculo do coeficiente será conforme Equação 29:

$$\text{Coef} = \frac{0.70}{QRNF*VRNF} = \frac{0.70}{18*5} = 0.0077$$

Equação 29: Resultado Cálculo do Coeficiente
Fonte: Autor

"*Coef*" será igual a 0,0077 e assim a fórmula tradicional de Ponto de Função para ajustar Valor TRNF é:

$$FRNF = (\text{TRNF} * 0.0077) + 0.65$$

Equação 30: Cálculo ajuste final
Fonte: Autor

5.3.3.5 ***Fator "Tamanho"***

Levando em consideração os levantamos bibliográficos descritos no Capítulo 3 e ilustrados na Figura 34 (seção 5.3.2), o fator Tamanho é mensurado baseado na complexidade funcional e nos seus requisitos não funcionais. O fator complexidade funcional é calculado conforme mostrado anteriormente e tem influência no cálculo do tamanho.

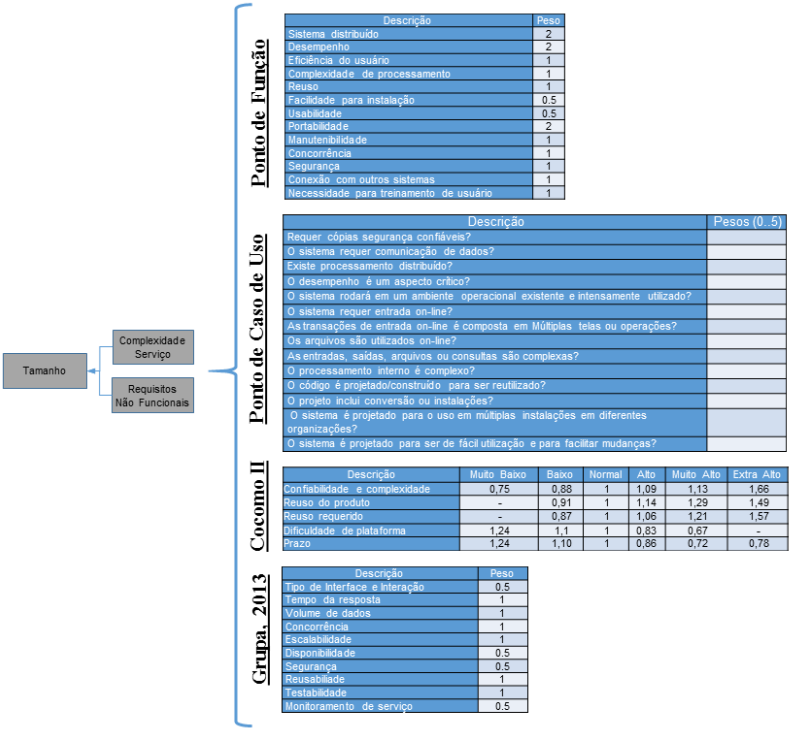
Conforme Karner (1993), este fator representa o tamanho da funcionalidade e que também se refere à granularidade. Ele é calculado com base nos requisitos não funcionais e complexidade funcional, conforma equação abaixo:

$$FTAM = FRNF * FCF$$

FRNF = Fator Requisitos Não Funcionais
FCF = Fator Complexidade Funcional

Equação 31: Cálculo do fator Tamanho
Fonte: Karner, 1993

Para definir os requisitos não funcionais foram avaliadas as 4 técnicas mencionadas anteriormente e mostradas na Figura 36. A Tabela 24 apresenta o resultado final da junção dos itens mencionados.



Descrição	Peso
Sistema distribuído	2
Desempenho	2
Eficiência do usuário	1
Complexidade de processamento	1
Reuso	1
Facilidade para instalação	0.5
Usabilidade	0.5
Portabilidade	2
Manutenibilidade	1
Concorrência	1
Segurança	1
Conexão com outros sistemas	1
Necessidade para treinamento de usuário	1

Descrição	Pesos (0..5)
Requer cópias segurança confiáveis?	
O sistema requer comunicação de dados?	
Existe processamento distribuído?	
O desempenho é um aspecto crítico?	
O sistema rodará em um ambiente operacional existente e intensamente utilizado?	
O sistema requer entrada on-line?	
As transações de entrada on-line é composta em Múltiplas telas ou operações?	
Os arquivos são utilizados on-line?	
As entradas, saídas, arquivos ou consultas são complexas?	
O processamento interno é complexo?	
O código é projetado/construído para ser reutilizado?	
O projeto inclui conversão ou instalações?	
O sistema é projetado para o uso em múltiplas instalações em diferentes organizações?	
O sistema é projetado para ser de fácil utilização e para facilitar mudanças?	

Descrição	Muito Baixo	Baixo	Normal	Alto	Muito Alto	Extra Alto
Confiabilidade e complexidade	0,75	0,88	1	1,09	1,13	1,66
Reuso do produto	-	0,91	1	1,14	1,29	1,49
Reuso requerido	-	0,87	1	1,06	1,21	1,57
Dificuldade de plataforma	1,24	1,1	1	0,83	0,67	-
Prazo	1,24	1,10	1	0,86	0,72	0,78

Descrição	Peso
Tipo de Interface e Interação	0.5
Tempo da resposta	1
Volume de dados	1
Concorrência	1
Escalabilidade	1
Disponibilidade	0.5
Segurança	0.5
Reusabilidade	1
Testabilidade	1
Monitoramento de serviço	0.5

Figura 36: Tamanho
Fonte: Autor

Tabela 24: Requisitos Não Funcionais

Nome	Valor (0-5)
Tempo da desposta	
Concorrência	
Escalabilidade	
Segurança	
Monitoramento de serviço	
Latência	
Taxa da transferência de dados	
Disponibilidade	
Confiabilidade	
Precisão/Acurácia	
Robustez	
Estabilidade	
Mensagens Confiáveis	
Integridade	
Interoperabilidade	
Fraco Acoplamento	
Abstração	
Sem estado	

*5.3.3.6 **Fator "Experiência dos Profissionais"***

Na implementação SOA, além do desenvolvedor, existem outros atores envolvidos, como o consumidor de serviços, desenvolvedor de serviços e provedor do serviços. Neste caso, conforme Azevedo et. al (2013), o desenvolvimento baseado em SOA envolve times multidisciplinares que não deveriam trabalhar isoladamente.

A experiência dos profissionais define a *expertize* real dos profissionais envolvidos no processo de desenvolvimento (LEWIS; SMITH; KONTOGIANNIS, 2010). Para definir um valor para a experiência profissional são avaliados 8 itens, oriundos do método Ponto de Caso de Uso (Tabela 25).

Tabela 25: Experiência dos Profissionais (EP)

Descrição	Peso	Valor (0-5)	Resultado
Equipe tem conhecimento no processo de desenvolvimento:	1.5		

Qual é o conhecimento da equipe com o processo de desenvolvimento do Software? Quanta maior seja a experiência maior é o valor.			
Experiência da equipe com o tipo de aplicação de negócio: Qual é o conhecimento da equipe no domínio do assunto? Quanto maior seja o conhecimento, maior deve ser o valor informado.	0.5		
Experiência da equipe com orientação a objetos: Quanto mais experiente a equipe com implementação orientada a objeto, maior deve ser o valor.	1		
Capacidade de análise: Quanto mais experiente é o líder maior é a representatividade do valor.	0.5		
Motivação da equipe: Qual é a motivação da sua equipe? Maior motivação representa maior valor.	1		
Trabalhadores em tempo parcial: Maior quantidade de funcionários em tempo parcial representa maior valor.	-1		
Dificuldade com a linguagem de programação escolhida: Maior dificuldade representa o valor maior.	-1		
Total (FEP)			

O fator experiência profissional (FEP) é a soma do campo resultado:

$$FEP = \Sigma\ EP\ Resultado$$

Equação 32: Cálculo fator Experiência Profissional
Fonte: Autor

*5.3.3.7 **Fator "Condição Ambiental"***

Esse fator indica a qualidade das ferramentas e dos processos de desenvolvimento de software na empresa (BIEBERSTEIN *et al*., 2005). O seu cálculo é semelhante ao fator experiência dos profissionais. De acordo com os levantamentos do estado da arte bem como o trabalho de Gupta (2013), para sua avaliação deve-se considerar os aspectos relacionados na Tabela 26.

Tabela 26: Condição Ambiental

Item	Descrição	Valor	Peso (0-5)	Resultado
1	Existe Governança SOA na organização?	1		
2	Existe ferramenta e plataforma adequada para o desenvolvimento?	1		
3	Existe descoberta dinâmica por serviços (services discovery)?	1		
4	A organização usa ESB e outros middlewares de suporte ?	1		
5	Os requisitos são estáveis: As mudanças nos requisitos causam aumento de retrabalho.	2		
Total (FCA)	**Total (FCA)**			

A Tabela 26 possui a coluna Peso (0-5). Para definir o valor a ser atribuído nesta coluna seguem-se as seguintes orientações:

Item 1: Os modelos para governança em SOA bastante conhecidos são os da IBM ((BROWN; LAIRD; CLIVE, 2009) e de (SCHEPERS; IACOB; VAN ECK, 2008). Baseado neles, governança SOA contem 6 estapas: *definição da estratégia SOA, alinhamento organizacional, gestão de portfólio de serviços, gestão do ciclo de vida de serviços, gestão de políticas SOA,* e *gestão de níveis de serviço*.
Uma sugestão para aplicação da medição da governança para este método é somar 1 ponto para cada etapa implementada. Porém, ainda assim pode ser que as etapas sejam implementadas apenas parcialmente. Neste caso sugere-se que seja feito um cálculo da cobertura em porcentagem. Por exemplo: se duas áreas tem 50% de implementação soma-se 0,5 ponto para cada área, totalizando 1 ponto. A etapa "Gestão

políticas SOA" do ciclo da vida da governança SOA não será utilizada nesta contagem já que esta etapa será avaliada separadamente no item "Conformidade" do Fator "Aspecto Negócio". Caso a organização não tenha nenhuma governança inclusa nos seus processos deve ser aplicado o valor zero.

Item 2: Entende-se que exista alguma ferramenta de suporte ou não. Neste caso pode-se atribuir como peso o valor 0 (ausência da ferramenta) ou 5 (existência de muito boa ferramenta).

Item 3: Entende-se que exista rastreabilidade ou não. Neste caso pode-se atribuir como peso o valor 0 (ausência da rastreabilidade) ou 5 (existência de completa rastreabilidade).

Item 4: Entende-se que existe ESB (ou middlewares equivalentes) ou não. Neste caso pode-se atribuir como peso o valor 0 (ausência do ESB) ou 5 (existência do ESB). No caso, 5 porque considera-se que se adotar um middleware de suporte do nível de um ESB é algo que traz grande qualidade ao ambiente geral mas também uma complexidade adicional.

Item 5: Como este item avalia mudanças nos requisitos (funcionais e não funcionais), para reduzir a subjetividade somente 2 valores podem ser atribuídos sobre este item. O valor 5 representa ausência de alterações nos requisitos; o valor 0 representa que os requisitos sofrem alterações durante projeto de desenvolvimento.

O cálculo é baseado na fórmula abaixo:

$$FCA = \Sigma\ Resultado$$

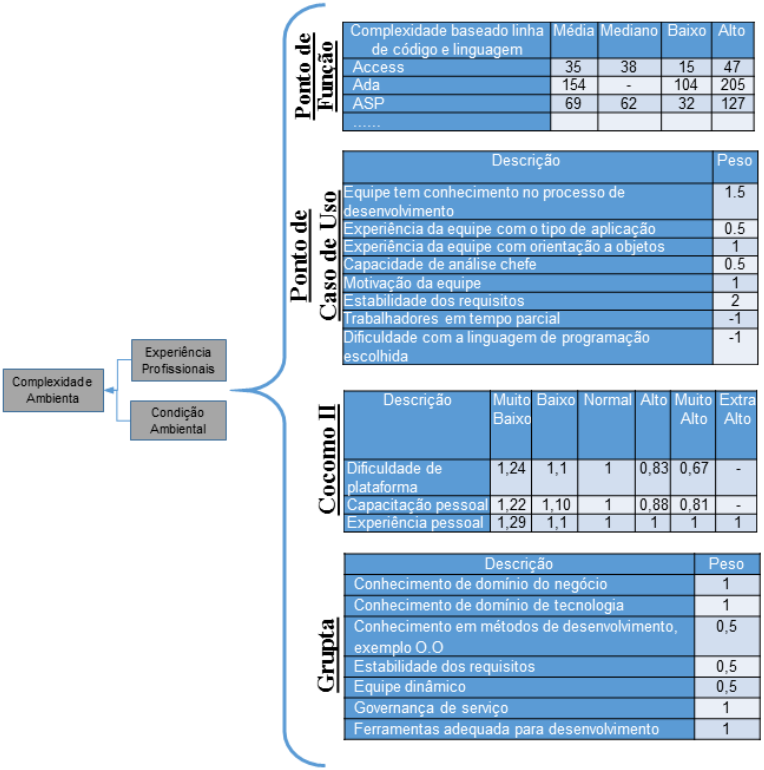
Equação 33: Cálculo do fator Condição Ambiental
Fonte: Autor

*5.3.3.8 **Fator "Complexidade Ambiental"***

SOA é uma estratégia global que afeta toda a organização, inclusive a sua cultura e métodos de trabalho (PAPAZOGLOU, 2012). Isso também é conhecido como Complexidade Ambiental (CAM). Este fator tem impactos nos esforços necessários para desenvolver o software. Além disto, este fator é determinante na estimativa do esforço (SOMMERVILLE, 2011).

O Fator Complexidade Ambiental está relacionado com o nível de experiência dos profissionais e as condições ambientais. No caso da experiência dos profissionais, pode-se citar a capacidade do líder de projeto, a motivação da equipe e a experiência com a aplicação de desenvolvimento. As condições ambientais indicam a qualidade das ferramentas e dos processos de desenvolvimento de software na empresa (BIEBERSTEIN *et al.*, 2005).

Para definição do cálculo de estimação da complexidade ambiental foram avaliados o método de Ponto de Função, o Ponto de Caso de Uso, o Cocomo II e a abordagem do Grupta (2013), conforme mostrado na Figura 37. Dentre esses, o Ponto de Caso de Uso é o que apresenta a maior abrangência no cálculo de complexidade ambiental.



Complexidade baseado linha de código e linguagem	Média	Mediano	Baixo	Alto
Access	35	38	15	47
Ada	154	-	104	205
ASP	69	62	32	127
.....				

Descrição	Peso
Equipe tem conhecimento no processo de desenvolvimento	1.5
Experiência da equipe com o tipo de aplicação	0.5
Experiência da equipe com orientação a objetos	1
Capacidade de análise chefe	0.5
Motivação da equipe	1
Estabilidade dos requisitos	2
Trabalhadores em tempo parcial	-1
Dificuldade com a linguagem de programação escolhida	-1

Descrição	Muito Baixo	Baixo	Normal	Alto	Muito Alto	Extra Alto
Dificuldade de plataforma	1,24	1,1	1	0,83	0,67	-
Capacitação pessoal	1,22	1,10	1	0,88	0,81	-
Experiência pessoal	1,29	1,1	1	1	1	1

Descrição	Peso
Conhecimento de domínio do negócio	1
Conhecimento de domínio de tecnologia	1
Conhecimento em métodos de desenvolvimento, exemplo O.O	0,5
Estabilidade dos requisitos	0,5
Equipe dinâmico	0,5
Governança de serviço	1
Ferramentas adequada para desenvolvimento	1

Figura 37: Complexidade Ambiental
Fonte: Autor

Considerando a taxonomia do método, para a estimação da complexidade ambiental, são avaliados os fatores de condição ambiental e experiência profissional.

Em relação a esses dois fatores, além dos itens mencionados na Figura 37, foram incluídos os itens "Existe descoberta dinâmica por serviços (services discovery)?" e "Organização tem estrutura ESB?", que têm relação com características de SOA e por sua vez têm influência no esforço da implementação do serviço.

Quando o método for utilizado para estimar o desenvolvimento de uma funcionalidade a ser desenvolvida tradicionalmente, o valor da relevância destes 2 itens e outros que são aplicados somente no desenvolvimento SOA podem ser atribuídas como zero, o que representa "não aplicável".

A Tabela 27 e Tabela 28 apresentam o resultado final dos itens a serem avaliados na implementação das funcionalidades caso elas sejam desenvolvidas no modo tradicional ou SOA.

Tabela 27: Experiência Profissional

Descrição	**Peso**
Equipe tem conhecimento no processo de desenvolvimento	1.5
Experiência da equipe com o tipo de aplicação e negócio	0.5
Experiência da equipe com orientação a objetos	1
Capacidade de análise	0.5
Motivação da equipe	1
Trabalhadores em tempo parcial	-1
Dificuldade com a linguagem de programação escolhida	-1

Tabela 28: Condição Ambiental

Descrição	**Peso**
Existe Governança SOA na organização	1
Existe ferramenta e plataforma adequado para desenvolvimento	1
Existe descoberta dinâmica por serviços (*services discovery*)?	1
Organização tem estrutura ESB?	1
Estabilidade dos requisitos	2

CAM é calculado somando o fator experiência dos profissionais com o fator condição ambiental (BIEBERSTEIN *et al*., 2005):

$$CAM = FEP + FCA$$

FEP = Fator Experiência Profissional
FCA = Fator Condição Ambiental

Equação 34: Cálculo Fator Complexidade Ambiental
Fonte: Autor

Este valor não é o valor final para representar a Complexidade Ambiental. Na verdade, um dos principais objetivos para calcular CAM é saber se o tamanho do serviço vai aumentar ou diminuir por conta de uma maior ou menor complexidade ambiental. Por isto, o valor CAM deve ser ajustado. Isto é feito através da aplicação da fórmula de ajuste da técnica Ponto de Caso de Uso, onde os valores 1,4 e -0,03 são constantes (KARNER, 1993) (ver Capítulo 3). Se o fator Complexidade Ambiental (FCAM) for inferior a 1 significa que o tamanho da funcionalidade irá diminuir devido à complexidade do ambiente. Se for maior do que 1, significa que o tamanho irá aumentar. Quando o resultado for igual a 1 significa que ele não tem nenhum impacto sobre o tamanho da funcionalidade.

$$FCAM = 1.4 + (-0.03 * CAM)$$

Equação 35: Ajuste do Fator complexidade ambiental
Fonte: Autor

5.3.3.9 ***Fator "Reuso"***

Um serviço é indicado para ser reutilizado por outros sistemas especialmente quando concebido como uma função lógica e fisicamente desacoplada. Além disto, o reuso é um critério fundamental para a adoção de SOA (LEWIS; SMITH; KONTOGIANNIS, 2010).

Projetar uma função para reuso exige, contudo, um esforço extra e consequentemente um maior custo para prepará-lo para poder ser integrado com / invocado por "qualquer" outro sistema (BENNETT, 2012). O valor atribuído ao fator de reuso (FR) corresponde ao número total dos Casos de Usos (ou outras funcionalidades) que podem usar essa função.

$$FR = \sum funcionalidade\ reusável$$

Equação 36: Cálculo fator Reuso
Fonte: Autor

5.3.3.10 ***Fator "Esforço"***

Conforme Karner (1993), este fator representa os esforços necessários para desenvolver a funcionalidade analisada. Esforço é representado por unidade de pessoa/tempo. A unidade de tempo costuma ser expressa em pessoa/horas, pessoa/dias ou pessoa/mês.

Levando em consideração a estrutura das técnicas Ponto de Função e Ponto de Caso de Uso, os seguintes fatores devem ser avaliados para calcular o esforço necessário para implementação de uma dada funcionalidade: Tamanho e Complexidade Ambiental.

Conforme Pressman (2016), nas técnicas Ponto de Função e Pontos de Caso de Uso, a produtividade representa a quantidade de pessoas por unidade de tempo necessário para atender 1 unidade de medição das técnicas Ponto de Função ou Pontos de Caso de Uso.

Para não causar dúvidas em relação ao método proposto com as unidades do Ponto de Função e Unidade Ponto de Caso de Uso, nesta proposta isto foi nomeado como *Unidade de Objeto*. Desta forma, a produtividade representa a quantidade de pessoas por hora para atender a 1 Unidade de Objeto (UO) e tem o objetivo de permitir estimar o tamanho de um sistema. Com isso a empresa precisa identificar qual é a sua produtividade em termos de UO, o que o faz utilizando os mesmos cálculos utilizados para identificar a produtividade por Ponto de Função ou Ponto de Caso de Uso.

No caso da empresa não ter dados históricos ou maturidade no processo da estimação, ela pode utilizar um valor padrão de 6 horas/dia para produtividade no processo de desenvolvimento, que segundo Jones (2017), é a produtividade média diária.

Com isso, o cálculo do esforço é calculado conforme Equação 37. O fator esforço é base para o cálculo dos fatores tempo de desenvolvimento e custo da implementação da funcionalidade, descritos nas próximas subseções.

$$FE_{homem/hora} = (FTAM * FCAM) * PROD$$

FTAM = Fator Tamanho
FCAM = Fator Complexidade Ambiental
PROD = Produtividade

Equação 37: Cálculo Fator Esforço
Fonte: Autor

*5.3.3.11 **Fator "Tempo de Desenvolvimento"***

Conforme Erl (2009), "estimar o custo, o tempo de desenvolvimento e os esforços para a implementação de serviços baseado em SOA é uma tarefa difícil devido a sua natureza diversificada, resultando uma estimativa imprecisa ".

Considerando a Figura 34, os cálculos dos fatores *Esforço, Custo* e *Tempo de Desenvolvimento* dependem dos cálculos de outros fatores, abordados/calculados anteriormente. A Figura 38 apresenta as fórmulas-base utilizadas para calcular esses três fatores por cada técnica, conforme levantamento bibliográfico (ver Capítulo 3).

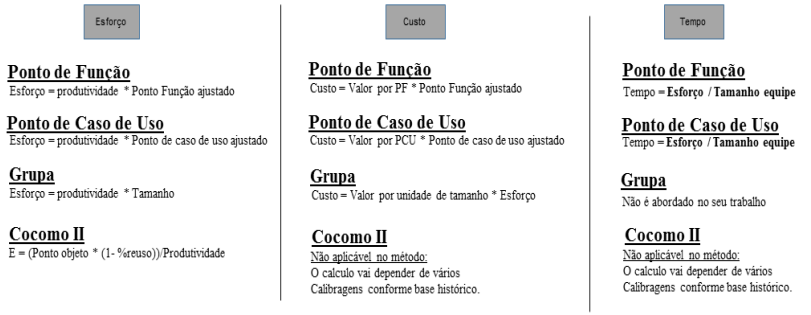


Figura 38: Fatores Esforço, Custo e Tempo
Fonte: Autor

O fator Tempo de Desenvolvimento (FTD) refere-se ao tempo estimado para desenvolver cada funcionalidade-candidata. Pode-se estima-lo por dia, hora ou outras medidas mais alinhadas com o processo de desenvolvimento de software da empresa. No método desenvolvido, adota-se a unidade *horas* como regra geral.

A experiência mostra que geralmente o esforço para desenvolver uma funcionalidade para ser reusada é duas vezes maior do que desenvolvê-la sem esta preocupação (BENNETT, 2012). Assim, no caso de reuso o cálculo é baseado na fórmula expressa na Equação 38.

$$FTD_{hora} = EF * 2 \ / TE$$

EF = *Esforço*
TE = *Tamanho do Equipe*

Equação 38: Cálculo fator tempo de desenvolvimento com Reuso
Fonte: Autor

Para o cálculo do tempo de desenvolvimento sem considerar o Reuso o tempo é calculado conforme Equação 39.

.

$$FTD_{hora} = EF \ / \ TE$$

EF = Esforço
TE = Tamanho do Equipe

Equação 39: Cálculo fator tempo de desenvolvimento sem Reuso
Fonte: Autor

5.3.3.12 ***Fator "Custo"***

Conforme Pressman (2016), os custos são considerados um dos fatores decisivos no desenvolvimento de um sistema. Como mencionado no cálculo do tempo de desenvolvimento, os fatores esforço e reuso são utilizados para medir os custos.

Para o cálculo do custo de funcionalidade com reuso, o custo é calculado dividindo o esforço total pela quantidade de reuso multiplicando-se pelo valor médio (VM) gasto por Unidade de Objeto (POULIN; HIMLER, 2006), conforme Equação 40. Como valor médio (VM), a empresa deve utilizar o mesmo valor Ponto de Função.

$$Custo = ((EF * 2) \ / FR) * VM$$

EF = Esforço
FR = Fator Reuso
VM = Valor Médio

Equação 40: Cálculo fator custo no desenvolvimento com reuso
Fonte: Autor

No caso de desenvolvimento sem considerar o reuso, o cálculo é expresso conforme Equação 41:

$$Custo = (EF \ / FR) * VM$$

EF = Esforço
FR = Fator Reuso
VM = Valor Médio

Equação 41: Cálculo fator custo no desenvolvimento sem reuso
Fonte: Autor

5.3.3.13 ***Fator "Negocio"***

Conforme Lewis, Smith e Kontogiannis (2010), a principal diferença do ciclo da vida de SOA com outros modelos de desenvolvimento software (como o RUP) é a necessidade de se definir no início do desenvolvimento as atividades relacionadas com a análise dos serviços e as suas relações com os objetivos de negócio.

De acordo com Gitman (2010), as decisões precisam estar alinhadas com o aspecto estratégico / de negócio da empresa. Isto aponta que, antes de mensurar outros fatores, a avaliação do aspecto negócio é prioritária. Porém, as técnicas da estimação tradicionais avaliam muito genericamente e subjetivamente o impacto do alinhamento ao negócio no desenvolvimento software (LEWIS; SMITH; KONTOGIANNIS, 2010).

Dada essa importância e essa lacuna, na atividade de medição foi incluído o fator "Aspecto Negócio". Além disto, e adotando-se o *framework* proposto por Lewis, Smith e Kontogiannis (2010), este aspecto é mensurado de modo paralelo aos demais fatores no método. Isto porque a estimação do desenvolvimento do software implica em um conjunto distinto de preocupações e de atividades para diferentes atores envolvidos. Por exemplo, para engenheiros de software, trata-se de requisitos funcionais e não funcionais, técnicas de integração, ferramentas, ambientes de desenvolvimento e middleware. Para pessoas de negócios, as maiores preocupações recaem sobre a implementação de estratégias de negócios. Isso se reflete inclusive no fato de que as "fórmulas" de tais cálculos não dependerem dos cálculos dos demais fatores. Daí que ele aparece de forma "desconectada" na taxonomia.

No entanto, o processo de análise do fator negócio precisa ser flexível e adaptável conforme as mudanças que possam ocorrer no planejamento estratégico, que por sua vez podem gerar alterações de valores nos aspectos de negócio (GITMAN, 2010).

Dois trabalhos foram usados como base para o cálculo do fator Aspecto Negócio: os de Lewis, Smith e Kontogiannis (2010) e de Börner e Goeken (2009). A Figura 39 ilustra de que forma esses trabalhos foram aproveitados e usados para tornar mais concreto e mensurável o aspecto do negócio numa estimação SOA.

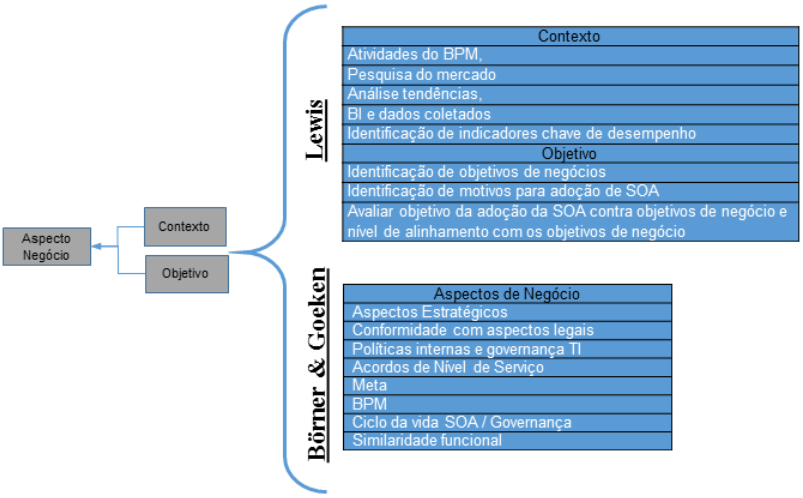


Figura 39: Aspecto de negócio
Fonte: Autor

Analisando esses trabalhos foi elaborada a Tabela 29, que lista os itens a serem avaliados no fator Aspectos de Negócio.

Tabela 29: Aspectos de Negócio

Contexto	**Valor (0-5)**
Atividades do BPM	
Pesquisa do mercado	
Análise tendências	
Business Intelligence e dados coletados associados	
Identificação de indicadores-chave de desempenho	
Objetivos	**Valor (0-5)**
Identificação de objetivos de negócios	
Identificação de motivos para adoção de SOA	
Avaliar objetivo da adoção da SOA em relação aos objetivos de negócio	
Meta	
Conformidade com aspectos legais e Politicas internas	

Lewis, Smith e Kontogiannis (2010) apontam duas atividades básicas que expressam os aspectos de negócio dentro de um processo SOA, aos que chama de *"Business Context Understanding"* e *"Business Objetives Specification"*. Em função disto esses aspectos foram considerados também como fatores do método e explicitamente modelados na taxonomia.

A atividade *"Business Context Understanding"* tem como objetivo entender o contexto no qual a empresa opera. Processos internos de negócios são documentados e analisados em relação às metas de negócios, do mercado e tendências. Para isto, são utilizadas principalmente informações sobre: processos da empresa ("BPM"), análise do mercado e tendências, inteligência de negócio (*BI – Business Intelligence*) e indicadores-chave de desempenho (*KPI – Key Performance Indicators*).

A atividade *"Business Objetives Specification"* visa formalizar os objetivos de negócios e analisar o impacto da adoção de SOA na concretização destes objetivos. As atividades envolvidas são: identificação de objetivos de negócios, identificação de motivos para adoção de SOA, avaliação de metas do SOA, e definição das métricas de avaliação de desempenho para determinar o nível de alinhamento com os objetivos de negócio.

No método proposto, com o objetivo de avaliar o aspecto de negócio de cada funcionalidade candidata, para cada item mencionado nos modelos de Lewis, Smith e Kontogiannis (2010) e Börner e Goeken (2009) foi criado um item equivalente em formato de pergunta. Cada um deve ser mensurado com base na Tabela 30, levando em consideração aspectos alinhados com os objetivos de negócio.

Para manter padronizada a forma da avaliação deste fator em relação aos outros fatores, cada pergunta recebe um valor entre 0 a 5. O valor 0 representa resposta negativa para a pergunta, enquanto que valores entre 1 a 5 representam o grau de intensidade de cobertura.

Tabela 30: Aspecto Negócio

Contexto	Fator	Valor (0-5)
Atividades do BPM	A funcionalidade está relacionada com atividades do BPM?	
Pesquisa do mercado	A funcionalidade faz parte da prioridade do mercado?	
Análise de tendência	A funcionalidade faz parte das tendências selecionada pela	

	empresa?	
BI e dados coletados	A funcionalidade faz parte do Roadmap?	
Identificação de indicadores chave de desempenho	A funcionalidade melhora KPI da organização?	
Objetivos	**Fator**	**Valor (0-5)**
Identificação de objetivos de negócios	A implementação faz parte dos objetivos da Empresa?	
Identificação de motivos	A funcionalidade faz parte dos motivos da adoção SOA?	
Avaliar objetivo da adoção da SOA contra objetivos de negócio e nível de alinhamento com os objetivos de negócio	Qual é nível de alinhamento de serviço com objetivos de negócio?	
Meta	Ajuda alcançar meta da organização	
Conformidade	A funcionalidade está conforme com os: – Aspectos legais: Existe leis ou normas para implementações das funcionalidades? – Politicas internas: Existem regras internas para implementações? (exemplo: dados do cliente devem ser mantidos na estrutura interna)	

Fonte: Lewis, Smith e Kontogiannis (2010)

Para o campo *valor* deve ser utilizada a Tabela 30, mas com a seguinte adaptação: o valor 0 representa a ausência do item; a faixa de 1 a 5 representa o grau de sua aplicação na organização. O valor 1 representa que o item está sendo utilizado fracamente e o valor 5 representa o uso do item com máxima intensidade.

No caso do item "Conformidade", baseado no trabalho de Börner e Goeken (2009), existe a necessidade de avaliar as seguintes conformidades no processo de estimação de desenvolvimento de SOA: Políticas internas e Aspectos legais. Assim, na Tabela 30, além do valor zero, há uma faixa de 1 a 5, dividida entre os dois itens mencionados. Neste caso, quando não há conformidade com nenhum deles o valor deve ser zero. Quando um deles é alinhado com a implementação o

valor deverá ser 3 (média entre 1 a 5). No caso dos dois tipos de conformidade serem aplicados o valor deve ser 5.

O Fator Negócio (FN) é calculado utilizando a média aritmética do valor total dos itens na tabela, conforme Equação 42.

$$FN = \sum valores\ dos\ itens\ /\ 10$$

Equação 42: Cálculo fator Negócio
Fonte: Autor

5.3.4 Atividade "Ranqueamento"

Com a lista das funcionalidades candidatas estimadas o próximo passo do método é analisar cada uma sob os diferentes pesos de importância que o gestor de TI dá a cada um dos quatro fatores principais de estimação (custo, tempo, esforço e alinhamento). Assim, há que se classificar cada funcionalidade conforme os pesos atribuídos e fazer um ranqueamento delas para posterior decisão final.

Importante esclarecer que não se trata de ranquear apenas os desenvolvimentos na forma de serviços (como em Kohlborn et al. (2009) e Marks e Bell (2006)), mas sim todas as funcionalidades-candidatas. As propostas de priorização existentes na literatura são subjetivas, pouco facilitando o processo decisório do gestor ao escolher quais serviços candidatos podem se tornar ativos de software (O´BRIEN, 2009) (LUTHRIA; RABHI, 2009).

Para tal foi aplicado uma abordagem multicritério. A decisão multicritério consiste em um conjunto de métodos e técnicas formais para auxiliar ou apoiar pessoas e organizações a tomarem decisões mais fundamentadas considerando subjetividades intrínsecas ao problema, esclarecendo o problema e avaliando as alternativas sob a influência de uma multiplicidade de critérios que são muitas vezes conflitantes entre si (ALMEIDA, 2011) (ALMEIDA, 2013).

Como comentado na seção 3.4, há vários métodos de análise multicritério. A escolha do método mais apropriado depende de diversos fatores, tais como: as características do problema analisado, o contexto considerado, o tipo de informação disponível, o seu grau de precisão e a estrutura de preferência do decisor (ALMEIDA, 2011).

Considerando a técnica de multicritério, Marks e Bell (2006) sugerem a elaboração de matriz de priorização de acordo com as

diretrizes da organização. Já Jones (2006) propõe o uso de matriz de ordenação de serviços baseada no valor que o serviço disponibiliza. Estas propostas de priorização são subjetivas e precisam ter um método da priorização com maior objetividade. Neste sentido, Hafezz, Zhang e Malak (2002) consideram o uso do método AHP mais adequado para priorização de serviços, o que neste trabalho também se considerou adequado, inclusive como corroborado em outros trabalhos, como (KARLSSON; WOHLIN; REGNELL,1998), (KARATZAS; DIOUDI; MOUSSIOPOULOS, 2003), (KARLSSON, 1996), (ALLEN et al., 2008), e (LEE; JOSHI; BAE, 2008), mencionados no Capítulo 3.

Na abordagem metodológica proposta, para a utilização do método AHP para a priorização das funcionalidades candidatas, há necessidade de seguir as seguintes passos:

1) Definição dos critérios da avaliação; no caso, aqueles quatro principais fatores de estimação do método desenvolvido.
2) Preencher as tabelas de avaliação para cada funcionalidade.

A Figura 40 ilustra a estrutura hierárquica da aplicação do modelo multicritério. No topo há o problema-foco, que é o da priorização das funcionalidades candidatas. O segundo nível corresponde aos critérios definidos, aqueles quatro fatores. O último nível da estrutura hierárquica refere-se às funcionalidades 'S' candidatas, previamente identificadas e que então necessitam ser priorizadas / ranqueadas.

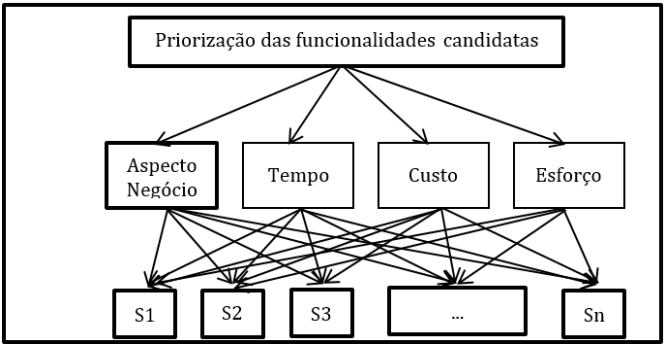


Figura 40: Aplicação do modelo multicritério
Fonte: Autor

O passo de preenchimento das tabelas de avaliação informa os valores nas tabelas do método AHP e encontra a prioridade dos serviços. A avaliação inicia pela determinação do peso relativo dos grupos dos fatores custo, esforço, tempo de desenvolvimento e negócio, avaliados dois a dois (Tabela 31). Para cada célula deve ser atribuído um peso, que representa a relação entre o fator mencionado na linha e o fator mencionado na coluna. Por exemplo, na linha 2 e coluna 1 (esforço x custo), o peso informado será a relevância do fator "esforço" sobre o fator "custo".

Tabela 31: Priorização dos Fatores

	Custo	Esforço	Tempo	Negócio
Custo	1			
Esforço		1		
Tempo			1	
Negócio				1

5.3.5 Atividade "Decisão"

Na perspectiva da estimação de software dentro da etapa de Análise SOA, esta última atividade do método está fortemente atrelada à gestão do posterior projeto de implementação das funcionalidades de acordo com o ranqueamento multicritério final calculado através do método.

O objetivo desta atividade é atuar como espaço de *decisão final* do gestor de TI sobre quais das funcionalidades candidatas elicitadas serão realmente desenvolvidas como serviços (e com que grau de granularidade / composição) ou como software tradicional, levando em conta os seus custos financeiros, tempo de desenvolvimento, esforço de desenvolvimento e grau de alinhamento ao negócio estimados.

Conforme Gitman (2010), a análise de recursos é o princípio econômico usado em administração para tomadas de decisões na escolha de projetos a serem implementados. Este princípio significa basicamente considerar se há os recursos financeiros para garantir o pleno desenvolvimento das funcionalidades necessárias dentro do horizonte temporal estimado (também conhecido como 'conciliação financeira').

Complementarmente, há que se analisar se existem na empresa os recursos humanos em número ou em disponibilidade disponíveis para aquele horizonte temporal. Com isso se pode inclusive analisar a possibilidade de subcontratação de desenvolvimento de alguma parte do

projeto. O gestor pode ainda decidir pelo cancelamento de desenvolvimentos (conforme seus custos) e assim mudar prioridades em função da estimação de forma mais alinhada aos objetivos estratégicos comerciais da empresa.

O método desenvolvido permite ainda fazer simulações de custos e alocação de recursos, ajudando o gestor de TI em traçar cenários alternativos de desenvolvimento, de alocação de recursos e de cronograma de desembolso financeiro ao longo do tempo considerando as prioridades da empresa.

Importante ressaltar que o método não decide; ele *recomenda*. Cabe ao gestor de TI da empresa (e demais *stakeholders* considerados importantes) tomar a decisão final, porém agora com valores rigorosamente calculados, de forma mais completa e adequada a SOA, e considerando todas as principais perspectivas de análise na estimação.

As informações utilizadas para alimentar os fatores podem sofrer alterações em processos de estimação. Por exemplo, profissionais muito experientes podem sair da empresa, novas tecnologias podem ser introduzidas, novos processos gerais de melhoria podem modificar métricas anteriores, etc.

5.4 RESUMO DAS CONTRIBUIÇÕES DO MÉTODO

O método proposto é o resultado de uma generalização de métodos de estimação para software ‘tradicional’, de fatores de decisão e suas adaptações / extensões para um cenário de estimação de serviços de software em uma perspectiva SOA.

O método é composto por 15 fatores e cada fator tem uma fórmula. Cada fórmula tem num número diferente de variáveis e algumas destas tem vários aspectos a serem ponderados. Globalmente, são 99 (noventa e nove) “aspectos de análise” que devem ser averiguados e preenchidos no método.

A Figura 41 mostra de forma resumida todas essas adaptações e extensões realizadas. O lado esquerdo da figura apresenta todas as adaptações efetuadas nas fórmulas em relação às adotadas pelas técnicas tradicionais (comentadas nas seções 3.3 e 5.2). O lado direto mostra as adaptações em cada um dos fatores envolvidos no processo da estimação de desenvolvimento tradicional para o desenvolvimento de software baseado SOA. Neste sentido, exceto o fator “Aspectos de Negócio”, que é aplicado tanto no processo da estimação de desenvolvimento tradicional quanto no desenvolvimento baseado SOA, outros campos

mencionados no lado direto precisam ser informados numa estimação SOA.

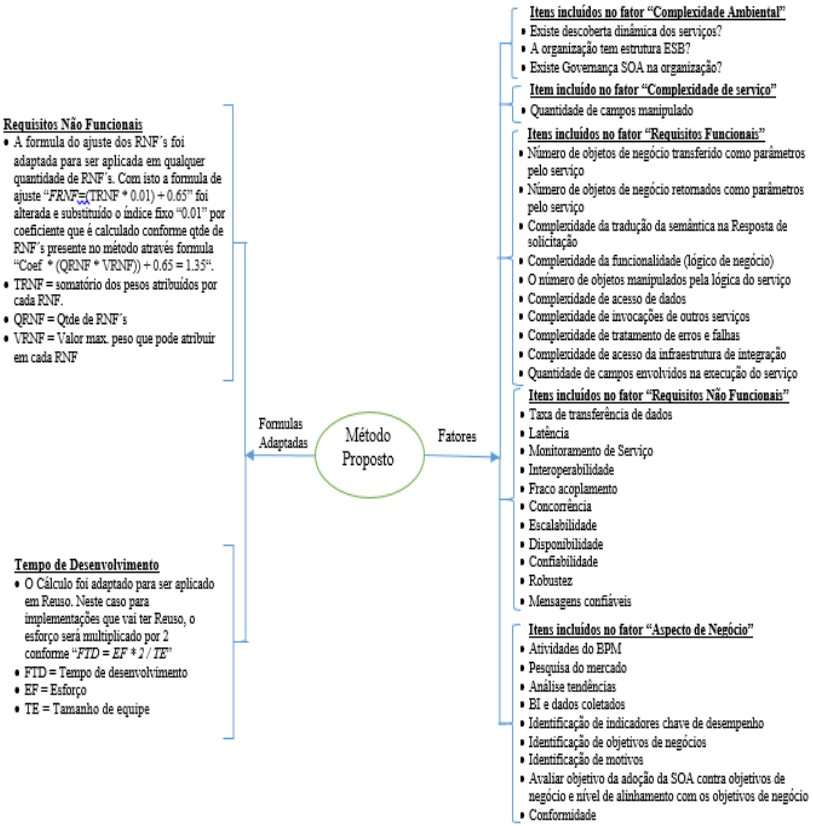


Figura 41: Contribuições para o método
Fonte: Autor

6 EXEMPLO DO USO DO MÉTODO E PROTÓTIPO

Após feita a apresentação conceitual detalhada do método de estimação desenvolvido, este capítulo tem por objetivo mostrar um exemplo do seu uso de forma a clarear suas potencialidades e sua delimitação. Neste sentido, este capítulo está organizado em cinco seções.

A primeira seção descreve o Caso de Uso a ser tomado como base na exemplificação.

A segunda seção mostra de forma sucinta o software-protótipo que foi desenvolvido para parcialmente automatizar a execução do método e que corresponde a um dos objetivos específicos desta tese no que toca a assistir os gestores no uso do método.

A terceira seção mostra o cálculo completo da estimação de uma (1) das funcionalidades do caso de uso usado como exemplo, provendo e comparando tanto o "custo geral" de implementação da funcionalidade como serviços como de desenvolvimento tradicional. Este cálculo é o que é efetuado pelo software-protótipo desenvolvido e cujo gestor simplesmente insere dados sobre o projeto SOA.

A quarta seção é análoga à terceira. A diferença é que nesta é mostrado o cálculo completo da estimação de quando mais do que uma funcionalidade será considerada na granularidade de um serviço.

A quinta e última seção faz uma análise geral dos resultados do exemplo.

6.1 CASO DE USO DE EXEMPLO

O caso tomado como base se refere a uma grande empresa nacional com larga experiência em BPM e SOA e com a qual este autor teve a oportunidade de interagir durante o desenvolvimento desta tese.

Esse caso é real e tem a ver com um processo de gerenciamento de incidentes no sistema atual da empresa. O objetivo do sistema de gerenciamento de incidente é solucionar o mais breve possível qualquer anomalia em um sistema de TI, com o mínimo de interrupção tanto do negócio como do usuário. A modelagem BPMN da Figura 42 ilustra as atividades e atores envolvidos nesse processo.

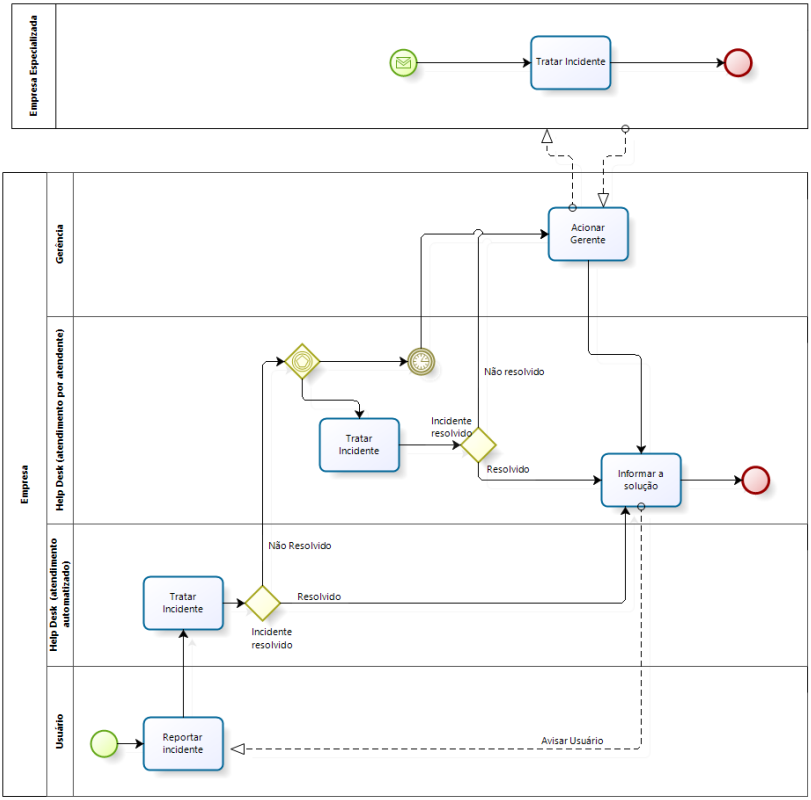


Figura 42: Gerenciamento de Incidentes
Fonte: Autor.

O fluxo desse processo começa pelo usuário relatando o incidente, encaminhando para o *help desk* automatizado. O help desk tem o objetivo de tratar o incidente buscando sua resolução na base de conhecimento da empresa de modo automatizado. Para isto, o sistema faz reconhecimento do texto que foi inserido pelo usuário e, com isso, o sistema compara o texto com padrões para encontrar uma solução. Se não conseguir, o incidente é escalado para o help desk via atendente, que trabalhará na solução do incidente. O *gateway* exclusivo modelado significa que somente um dos fluxos possa ser executado. Se a pessoa do help desk não atender o incidente dentro do tempo estabelecido, o gerente será acionado e, neste caso, o incidente será repassado para uma empresa terceira especializada no tratamento do assunto.

No caso, se o tratamento do incidente for iniciado por atendente, existem duas possíveis situações: na primeira, quando é encontrada a solução, isto é registrado na base de conhecimento para futuras consultas. Para este registro o texto precisa ser antes normalizado para que os textos sejam padronizados. Posteriormente, é enviado um aviso para o usuário e o processo é finalizado. A segunda opção é quando o atendente não encontrou a solução; neste caso o assunto é novamente encaminhado para a gerência, que encaminhará o assunto para empresa especializada tratar. Com isso, a solução do assunto será registrada na base de conhecimento pelo sistema de gerenciamento de incidentes para futuras consultas e enviado aviso para o usuário e o processo é finalizado. A Figura 43 ilustra uma visão global das funcionalidades desse Caso e suas inter-relações.

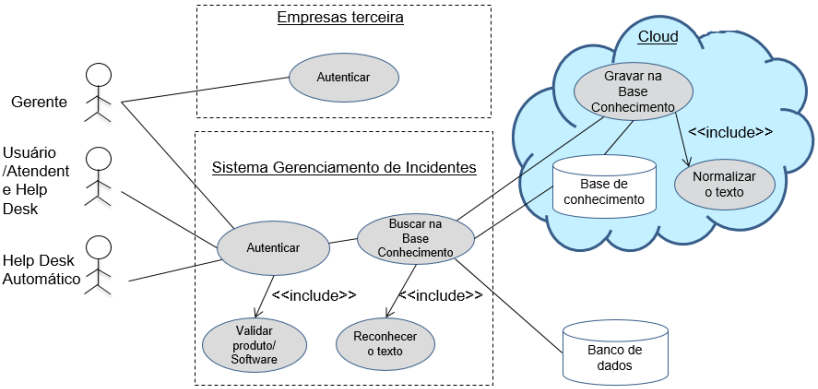


Figura 43: Exemplo da aplicação do método
Fonte: Autor.

Na figura, as elipses representam as funcionalidades candidatas enquanto os retângulos tracejados representam as empresas. As funcionalidades e base de conhecimento inseridas na nuvem representam serviços existentes que podem ser acessados como SaaS.

A lista completa das funcionalidades candidatas identificadas do Caso de Uso para atender o modelo de processo de negócio acima descrito é mostrada na Tabela 32.

Tabela 32: Funcionalidades identificados

Funcionalidade	Ator
- Autenticar	- Usuário
	- Gerente
	- Help Desk Automático
	- Atendente Help Desk
- Validar produto/software	- Usuário
	- Gerente
	- Help Desk Automático
	- Atendente Help Desk
- Reconhecer o texto	- Sistema através API
- Buscar na Base Conhecimento	- Sistema através API
- Gravar na Base Conhecimento	- Sistema através API
- Normalizar texto	- Sistema através API

Fonte: Autor.

Neste exemplo as funcionalidades “autenticar” e “validar produto/software” fazem parte dos legados e são implementados de forma ‘tradicional’, i.e. não orientada a serviços. Tais funcionalidades foram identificadas na atividade de análise de *gap* do ciclo de vida SOA e, suscintamente, servem para:

- Autenticar: validar usuário e suas permissões;
- Validar o produto: validar quais produtos de software que o usuário e atendente podem utilizar;
- Reconhecer o texto: interpretar o texto escrito pelo usuário e facilitar a busca na base de conhecimento;
- Buscar na base de conhecimento: esta funcionalidade é baseada nas regras e busca as informações na base de conhecimento;
- Gravar na base de conhecimento: esta funcionalidade segue as regras e padrões necessários para inserir as informações na base de conhecimento;
- Normalizar texto: normalizar o texto escrito pelo atendente para facilitar a inserção na base de conhecimento.

Considerando a lista das funcionalidade candidatas, o gestor pode avaliar as opções, listadas na Tabela 33.

Tabela 33: As opções das implementações

Opção	Ator
1	Implementar as novas funcionalidades como serviços, em uma ótica SOA.
2	Implementar as novas funcionalidades no modo tradicional, inseridas num software monolítico.
3	Avaliar as implementações das funcionalidades de modo isolado e também em conjunto, ou seja, cada funcionalidade como um (1) serviço ou mais do que uma funcionalidade com um (1) serviço.
4	Implementar algumas funcionalidades como serviço e outras como software tradicional.

O método se aplica a cada funcionalidade individualmente, que depois é estendida para análises em conjunto. Para efeitos de exemplificação do método se tomará a funcionalidade "*gravar na base de conhecimento*", sendo avaliada a sua implementação no modo tradicional e baseado em serviços já que uma outra facilidade do método é o de estimar o custo geral nos dois modelos de implementação e posterior comparação para fins de apoio à decisão.

Ao final será apresentada a estimativa dessa funcionalidade em conjunto com a "*normalizar texto*" (modo 'composto'). A Tabela 34 e Tabela 35 apresentam um resumo da especificação da cada funcionalidade mencionada.

Tabela 34: Detalhes da funcionalidade *Gravar na base conhecimento*

Informação	Descrição
Nome	Gravar na Base Conhecimento (GBC)
Funcionalidade	Criar integração entre os sistemas e Base de Conhecimento, gravando as informações já normalizadas na base de conhecimento.
Dados Entrada	Texto normalizado
Dados Saída	Confirmação do sucesso na gravação ou erro

Tabela 35: Detalhes da funcionalidade *Normalizar texto*

Informação	Descrição
Nome	Normalizar texto (NT)
Funcionalidade	Normalizar o texto baseado em padrões e regras na base de conhecimento
Dados Entrada	Texto gerado pela usuário ou sistema
Dados Saída	Confirmação do sucesso ou mensagem do erro

6.2 SOFTWARE DE APOIO À EXECUÇÃO DO MÉTODO

Como pode ser observado no Capítulo 5, são muitas informações que o gestor tem que saber e calcular para percorrer todas as atividades (cada um dos fatores do método na devida sequência) do processo completo de estimação de um caso de uso, incluindo as análises de implementação como serviço, como software tradicional, uma combinação de ambos, o ranqueamento multicritério e finalmente a geração de um quadro comparativo.

Além de trabalhoso, esta atividade acaba por ser naturalmente sujeita a erros de cálculos e tomar um tempo considerável do gestor.

Para facilitar e assistir o gestor no uso do método (além de ajudar nas etapas de verificação, avaliação e validação desta tese) foi desenvolvido um pequeno software, como protótipo. Ele foi implementado na linguagem *Visual Basic* e utilizado o *Excel* como ferramenta de base de cálculo dos vários fatores do método e comparativos de estimação.

Conceitualmente, este software pode ser visto como um artefato de apoio à *base practice* (o método desenvolvido) do processo de *Análise* no âmbito do ciclo de vida e modelo de governança SOA. Além disto, pode ser utilizado para realizar simulações e comparações com base histórica que será gerada com o uso da ferramenta.

A Figura 44 mostra a tela principal do aplicativo. Observa-se a sequência dos passos a serem seguidos para inserção de dados do método.

Figura 44: Tela principal da ferramenta
Fonte: Autor.

Ao clicar na opção “Definir” o aplicativo pede para se informar dados comparativos dos fatores Custo, Esforço, Tempo e Negócio (conforme técnica multicritério AHP) (Figura 45). Nesta avaliação é preciso atribuir o valor, comparando os fatores de dois a dois, conforme já explicado no Capítulo 5.

Na mesma tela é necessário inserir os dados referente ao valor gasto por Ponto de Função e o valor da produtividade da equipe.

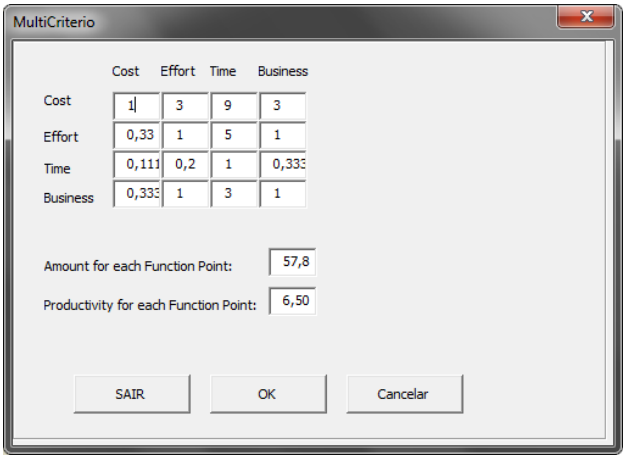


Figura 45: Dados do Multicritério
Fonte: Autor.

Ao sair da tela de multicritério e voltando para tela principal (Figura 44), deve-se clicar no botão “Preencher” onde o aplicativo apresentará a tela de cadastro de funcionalidades (Figura 46).

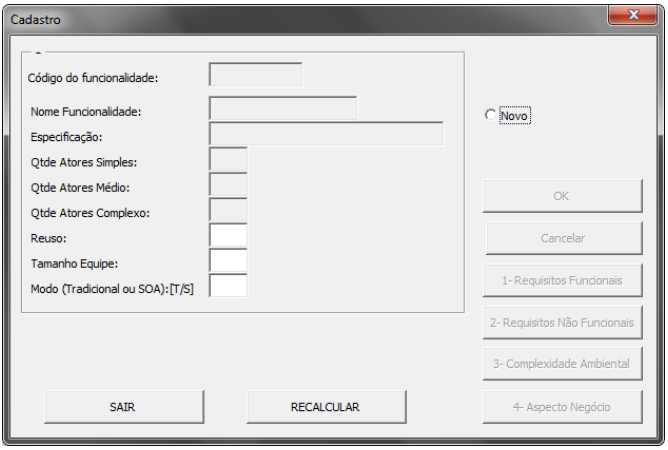


Figura 46: Cadastro dos dados Input
Fonte: Autor

Para realizar o cadastro o gestor inicialmente deve clicar na opção "novo" e inserir os dados solicitados. Após preencher os dados solicitados nesta tela, deve ser clicado no botão OK e depois clicar no botão "1-Requisitos Funcionais", que apresentará a tela com itens relacionados aos requisitos funcionais (Figura 47).

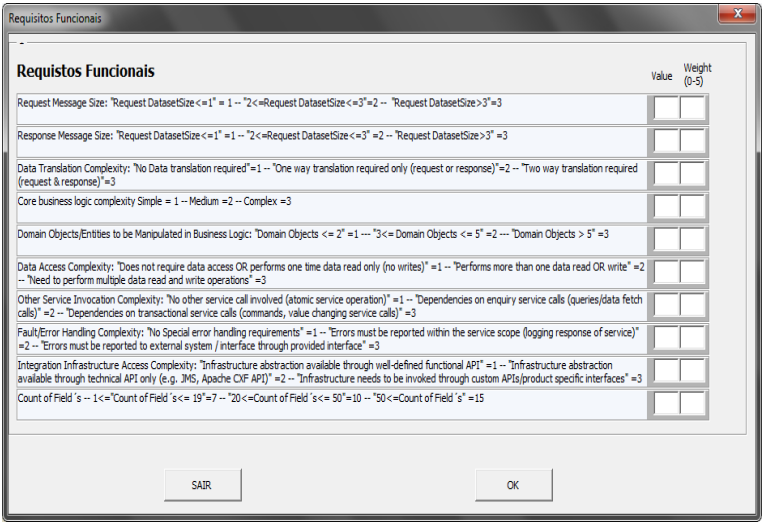


Figura 47: Requisitos Funcionais
Fonte: Autor

Ao finalizar as atribuições dos valores e pesos clicando no botão "OK", ao sair o usuário volta para tela de cadastro (Figura 46). Na tela de cadastro deve ser clicado no botão "2-Requisitos Não Funcionais", informando pesos para cada item (Figura 48).

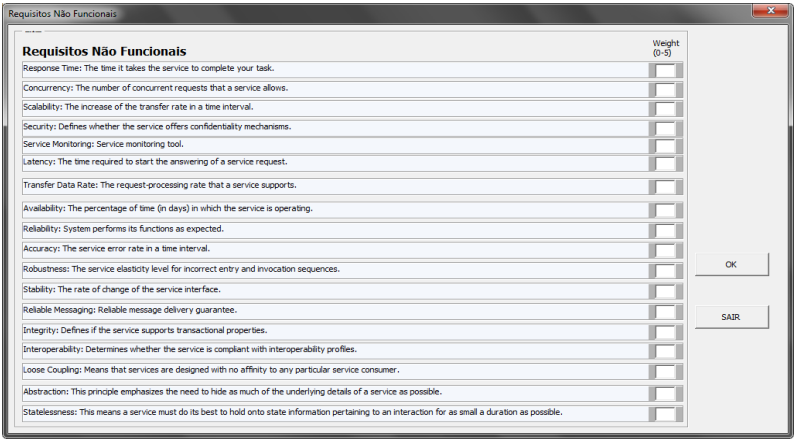


Figura 48: Requisitos Não Funcionais
Fonte: Autor

Ao finalizar as atribuições dos pesos, clicando no botão "OK" e depois "Sair", volta para tela de cadastro (Figura 46). Na tela de cadastro deve ser clicado no botão "3-Complexidade Ambiental" informando os pesos para cada item (Figura 49).

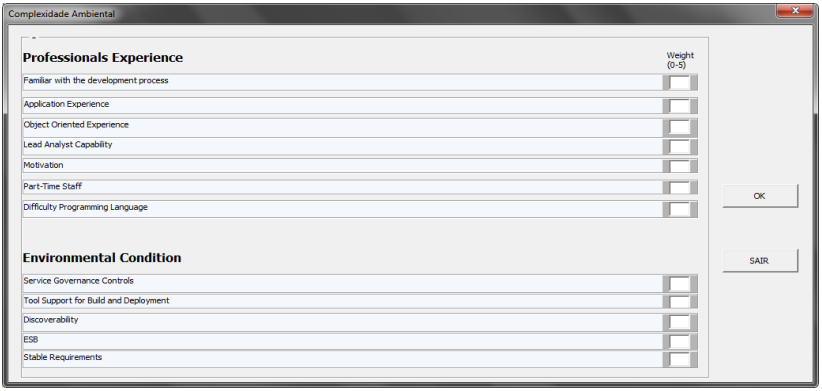


Figura 49: Dados Complexidade Ambiental
Fonte: Autor.

Ao finalizar as atribuições dos pesos, clicando no botão “OK”, ao sair, o usuário volta para tela de cadastro (Figura 46). Na tela de cadastro deve ser clicado no botão “4-Aspecto Negócio” informando pesos para cada item (Figura 50).

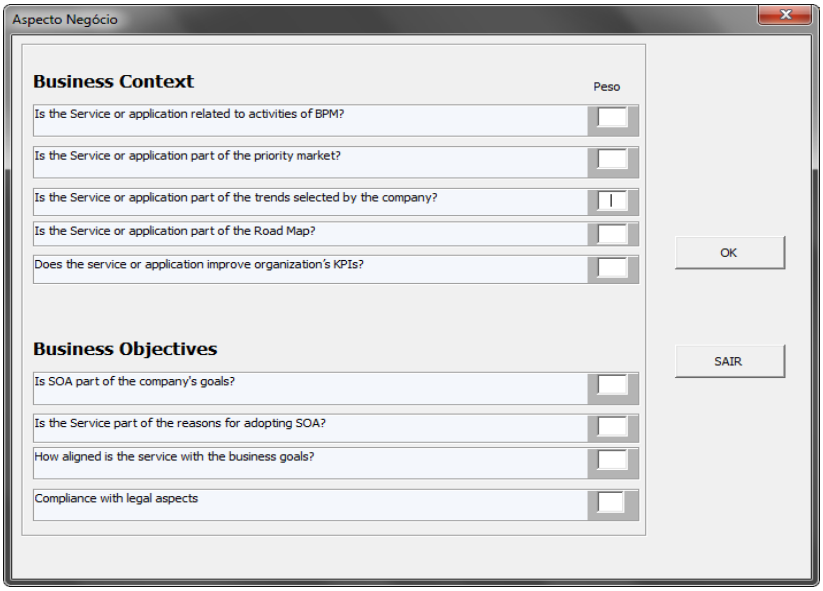


Figura 50: Dados Aspecto Negócio
Fonte: Autor.

Ao finalizar as atribuições dos pesos e sair, o usuário deve voltar para tela principal (Figura 44).

A empresa pode cadastrar várias funcionalidades e eventualmente pode não ter interesse em estimar todas. Portanto, na tela principal ao clicar no botão “Medir” aparece a planilha (Figura 51) com a lista das funcionalidades cadastradas anteriormente, permitindo ao usuário a seleção das quais devem ser estimadas, inserindo “x” no seus campo “Selection”. A coluna “modo de desenvolvimento” é informada pelo usuário do método, e o objetivo de indicar se as informações da funcionalidade são referentes ao desenvolvimento tradicional (T) ou serviço (S).

ID	Selection	Name of Functionality	Especification	Modo de desenvolvimento (T- Tradicional ou S-SOA)
1	☒	GBC - Tradicional	Gravar na Base Conhecimento	T
2	☒	GBC - Serviço	Gravar na Base Conhecimento	S
3		Calendário	Calendário dos treinamentos	S
4	☒	NT-Serviço	Normalizar Texto	S
5	☒	NT-Tradicional	Normalizar Texto	T
5	☒	BBC- Tradicional	Busca na Base Conhecimento	T
6		Agendamento	Agendamento da mensagens	S
6	☒	BBC-Serviço	Busca na Base Conhecimento	S
7	☒	RT - Tradicional	Reconhecimento do texto	T
8	☒	RT - Serviço	Reconhecimento do texto	S

Figura 51: Seleção das funcionalidades
Fonte: Autor.

Por fim, ao clicar no botão "Calcular" na tela principal, o aplicativo vai calcular os fatores e vai automaticamente fazer o ranqueamento multicritério em função do maior percentual de alinhamento ao negócio das funcionalidades (Figura 52).

Perc.	Func.	Custo (R$)	Esforço (Homem/H)	Tempo (H)	Negócio	Reuso	Tam. Equipe
22,90%	RT - Serviço	6035,31	208,55	417,09	4	4	1
18,20%	BBC - Serviço	8389,5	217,42	434,84	3,67	3	1
13,30%	BBC - Tradicional	13644,47	235,74	235,74	2,11	1	1
11,10%	GBC - Serviço	13987,18	483,32	966,63	3,44	4	1
10,50%	RT - Tradicional	17189,62	296,99	296,99	1,56	1	1
9,90%	NT- Serviço	17806,66	461,47	922,94	3,44	3	1
7,50%	NT- Tradicional	25979,2	673,27	1346,54	3,33	3	1
6,50%	GBC - Tradicional	36888,16	637,32	1274,64	3,33	2	1

Figura 52: Resultado final
Fonte: Autor.

Avaliando a Figura 52, observa-se que a funcionalidade “RT – Serviço” tem maior prioridade perante as outras funcionalidade. Na sequencia surge a funcionalidade “BBC - Serviço” e assim por diante. Isto quer dizer que, baseado nos pesos e nos valores atribuídos durante a aplicação do método, é sugerido que a empresa deva iniciar pela implementação do RT-Serviço.

A implementação das demais funcionalidades e naquela sequência sugerida dependeria da disponibilidade de recursos financeiros e recursos humanos.

6.3 EXEMPLO DE ESTIMAÇÃO *ISOLADA* DA FUNCIONALIDADE *“GBC”*

A seguir serão detalhados os cálculos da estimação da implementação da funcionalidade *GBC*, individualmente, tanto no modo tradicional como no baseado em serviços (opções 1 e 2 da Tabela 33). Em outras palavras, detalha os cálculos que o software desenvolvido automatiza.

A sequência dos itens corresponde à do método em si, onde o cálculo de certos fatores depende do cálculo anterior de outros fatores, conforme taxonomia e fórmulas de cálculo do método apresentadas no Capítulo 5.

- **Cálculo do Fator *Complexidade da Integração***

Segue abaixo os dados do exemplo aplicado no fator complexidade da integração da funcionalidade GBC (Tabela 36).

Tabela 36: Tipo de atores

Actor	**Weight**	**Traditional**		**SOA Based**	
		Value	***Result***	***Value***	***Result***
Simples	1	3	3	2	2
Média	2	1	2	0	0
Complexo	3	0	0	0	0

$$FCI = \Sigma\ Resultado$$

$$ICF_{Tradicional} = 3 + 2 + 0 = 5$$
$$ICF_{SOA} = 2 + 0 + 0 = 2$$

- **Cálculo do Fator *Requisitos Funcionais***

A Tabela 37 apresenta os dez aspectos a serem avaliados e os valores atribuídos em cada um deles.

Tabela 37: Complexidade dos requisitos funcionais

1- Tamanho da mensagem da requisição							
Rating	***Description***	**Tradicional**			**Baseado SOA**		
		Valor (1-3)	**Peso (0-5)**	**Res.**	**Valor (1-3)**	**Peso (0-5)**	**Res.**
1	Dataset<=1						
2	2<= Dataset <=3						
3	Dataset>3	1	1	1	1	1	1
2- Tamanho da mensagem da resposta							
Faixa	***Descrição***	**Tradicional**			**Baseado SOA**		
		Valor (1-3)	**Peso (0-5)**	**Res.**	**Valor (1-3)**	**Peso (0-5)**	**Res.**
1	Dataset<=1						
2	2<= Dataset <=3						
3	Dataset>3	1	1	1	1	3	3
3- Complexidade de tratamento de dados na requisição/resposta							
Faixa	***Descrição***	**Tradicional**			**Baseado SOA**		
		Valor (1-3)	**Peso (0-5)**	**Res.**	**Valor (1-3)**	**Peso (0-5)**	**Res.**
1	Não há necessidade de tratamento de dados						
2	Tratamento de dados apenas em um sentido (requisição ou resposta)	3	3	9	3	5	15
3	Tratamento de dados em dois sentidos (requisição e resposta)						
4- Complexidade de lógica de negócio							
Rating		**Tradicional**			**Baseado SOA**		
		Valor (1-3)	**Peso (0-5)**	**Res.**	**Valor (1-3)**	**Peso (0-5)**	**Res.**
1=Simples, 2=Média, 3=Complexo		3	5	15	3	5	15
5- O número de objetos manipulados pela lógica							
Rating	***Description***	**Tradicional**			**Baseado SOA**		
		Valor (1-3)	**Peso (0-5)**	**Res.**	**Valor (1-3)**	**Peso (0-5)**	**Res.**
1	Objetos<=2						
2	3<= Objetos<=5	3	5	15	3	4	12
3	Objetos>5						

6- Complexidade de acesso aos dados							
Rating	***Description***	**Tradicional**			**Baseado SOA**		
		Valor (1-3)	**Peso (0-5)**	**Res.**	**Valor (1-3)**	**Peso (0-5)**	**Res.**
1	Não há necessidade de gravar dados (somente leitura)	3	5	15	3	4	12
2	Grava dados ou somente leitura						
3	Precisa executar várias operações de leitura e gravação de dados						

7- Complexidade da integração com outros serviços							
Rating	***Description***	**Tradicional**			**Baseado SOA**		
		Valor (1-3)	**Peso (0-5)**	**Res.**	**Valor (1-3)**	**Peso (0-5)**	**Res.**
1	Não tem interação com outros serviços (serviço atômico)	1	3	3	2	2	4
2	Necessita realizar a busca de serviço a ser chamado através de séries						
3	Necessita de chamado de outro serviço transacional						

8- Complexidade de tratamento da falha e do erro							
Rating	***Description***	**Tradicional**			**Baseado SOA**		
		Valor (1-3)	**Peso (0-5)**	**Res.**	**Valor (1-3)**	**Peso (0-5)**	**Res.**
1	Não há necessidade de tratamento	3	5	15	2	3	6
2	Erros devem ser relatados dentro do escopo do serviço						
3	Erros devem ser reportados via interface externo						

9- Complexidade da infraestrutura da integração							
Rating	*Description*	Tradicional			Baseado SOA		
		Valor (1-3)	Peso (0-5)	Res.	Valor (1-3)	Peso (0-5)	Res.
1	Através API bem definido	3	3	9	2	1	2
2	Atraves API técnico , exemplo: JMS, Apache CXF API						
3	Atraves de interface específico do produto						
10- Quantidade de campos envolvidos							
Rating	*Description*	Tradicional			Baseado SOA		
		Valor (1-3)	Peso (0-5)	Res.	Valor (1-3)	Peso (0-5)	Res.
7	1<= quantidade <= 19	7	5	35	7	5	35
10	20<=quantidade<= 50						
15	50<= quantidade						

$$FRF = \Sigma\, Resultado$$

$FRF_{Tradicional} = 118$
$FRF_{SOA} = 105$

- **Cálculo do Fator *Complexidade Funcional***

Para calcular a complexidade funcional é somada a complexidade de requisitos funcionais com a complexidade dos atores.

$$Complexidade\ funcional\ (FCF) = FRF + FCI$$

$FCF_{Tradicional} = 5 + 118 = 123$
$FCF_{SOA} = 2 + 105 = 107$

- **Cálculo do Fator *Requisitos Não Funcionais***

A Tabela 38 apresenta os requisitos não funcionais e os valores atribuídos em cada um deles.

Tabela 38: Requisitos não Funcionais

Name	Peso(0-5)	
	Tradicional	***Baseado SOA***
Tempo de resposta	1	3
Concorrência	1	5
Escalabilidade	3	2
Segurança	5	2
Monitoramento de serviço	1	2
Latência	2	2
Taxa da transferência de dados	3	2
Disponibilidade	3	2
Confiabilidade	5	2
Precisão	5	4
Robustez	2	4
Estabilidade	3	5
Mensagens Confiáveis	5	1
Integridade	3	0
Interoperabilidade	5	3
Fraco acoplamento	1	3
Abstração	5	5
Sem estado	1	5
Total (TNFR)	**54**	**52**

Como explicado nos Capítulos 3 e 5, o valor total TRNF precisa ser ajustado. Para tal é preciso identificar o coeficiente a ser inserido na formula de ajuste. Por exemplo, dado que a quantidade de requisitos não funcionais é 18, o coeficiente da formula de ajuste é = 0.0077.

Desta forma, o fator complexidade dos requisitos não funcionais é calculado da seguinte forma:

$$FRNF = (0.0077 * \text{TRNF}) + 0.65$$

$$FRNF_{Tradicional} = (0.0077*54)+0.65 = 1.070$$
$$FRNF_{SOA} = (0.0077*52)+0.65 = 1.054$$

- **Cálculo do Fator *Tamanho***

O tamanho do software é calculado por meio da fórmula abaixo:

$$FTAM = FRNF * FCF$$

$$FTAM_{Tradicional} = 1.07 * 123 = 131.61$$

$$FTAM_{SOA} = 1.054 * 107 = 112.83$$

- **Cálculo do Fator *Experiência dos Profissionais***

Considerando a Tabela 39 para calcular o fator complexidade funcional, os valores da coluna *resultado* são somados.

Tabela 39: Experiência dos profissionais

Descrição	Tradicional			Baseado SOA		
	Peso	Valor (0-5)	*Result*	Peso	Valor (0-5)	*Result*
Equipe tem conhecimento no processo de desenvolvimento	1.5	3	4.5	1.5	3	4.5
Experiência da equipe com o tipo de aplicação e negócio	0.5	5	2.5	0.5	2	1
Experiência da equipe com orientação a objetos	1	2	2	1	3	3
Capacidade de análise	0.5	4	2	0.5	2	1
Motivação da equipe	1	3	3	1	2	2
Trabalhadores em tempo parcial	-1	1	-1	-1	1	-1
Dificuldade com a linguagem de programação escolhida	-1	3	-3	-1	4	-4

O fator experiência dos profissionais (FEP) é calculado por meio da seguinte formula e seu resultado apontado abaixo:

$$FEP = \Sigma\ EP\ Resultado$$

$$FEP_{Tradicional} = 10$$

$$FEP_{SOA} = 6.5$$

- **Cálculo do Fator *Condição Ambiental***

Considerando a Tabela 40 para calcular o fator condição ambiental, os valores da coluna *resultado* são somados.

Tabela 40: Condição Ambiental

Descrição	Tradicional			Baseado SOA		
	Peso	Valor (0-5)	Res.	Peso	Valor (0-5)	Res.
Existe Governança SOA na organização	1	5	5	1	5	5
Existe ferramenta e plataforma adequado para desenvolvimento	1	3	3	1	4	4
Existe descoberta dinâmica por serviços (services discovery)?	1	2	2	1	5	5
Organização tem estrutura ESB?	1	1	1	1	3	3
Estabilidade dos requisitos	2	2	4	2	2	4

$$FCA = \Sigma\ Resultado$$

$$FCA_{Traditional} = 15$$
$$FCA_{SOA} = 21$$

- **Cálculo do Fator *Complexidade Ambiental***

Este fator é calculado por meio da formula abaixo e seu resultado é apesentado abaixo:

$$CAM = FEP + FCA$$

$$CAM_{Tradicional} = 10 + 15 = 25$$
$$CAM_{SOA} = 6{,}5 + 21 = 27.5$$

O valor precisa ser ajustado considerando a formula abaixo (ver Capítulos 3 e 5) e o resultado é apresentado abaixo:

$$FCAM = 1.4 + (-0.03 * CAM)$$

$$FCAM_{Tradicional} = 1.4 + (-0.03 * 25) = 0.65$$

$$FCAM_{SOA} = 1.4 + (-0.03 * 27.5) = 0.575$$

- **Cálculo do Fator *Reuso***

O valor do reuso corresponde ao número de funcionalidades dos outros Casos de Uso que podem usar essa função:

$$FR_{Tradicional} = 2$$
$$FR_{SOA} = 4$$

- **Cálculo do Fator *Esforço***

O esforço é calculado com duas finalidades: apoio na estimativa do tempo de desenvolvimento e do custo. O fator esforço é calculado multiplicando a produtividade estimada pelo tamanho do serviço em Ponto de Função.

No exemplo em questão, a produtividade é igual 7,45 homens/hora, ou seja, a empresa precisa de 7,45 pessoas trabalhando durante 1 hora para atender cada 1 Unidade Objeto. O resultado do cálculo esforço segue abaixo:

$$FE_{homem/hora} = (FTAM * FCAM) * PROD$$

$$FE_{Tradicional} = 131.61 * 0.65 * 7,45 = 637.32$$

$$FE_{SOA} = 112.83 * 0.575 * 7,45 = 483.32$$

- **Cálculo do Fator *Tempo de Desenvolvimento***

Pode-se estimar o tempo de desenvolvimento por dia, hora ou outras medidas mais adequadas às práticas de desenvolvimento de software da empresa. Neste sentido, é necessário apontar o tamanho de equipe. No caso do exemplo, o tamanho de equipe envolvido é de 2 pessoas no projeto SOA e de três pessoas no projeto tradicional. Além disto, esta funcionalidade será reutilizada dez vezes no modo tradicional e doze vezes por meio do SOA. Então o cálculo será:

$$FTD_{hora} = EF * 2 \ / TE$$

$$FTD_{hora\ Tradicional} = (637.32 * 2) /1 = 1274.64 \text{ horas}$$
$$FTD_{hora\ SOA} = (483.32 * 2) /1 = 966.63 \text{ horas}$$

- **Cálculo do Fator *Custo***

O Fator Custo (FC) é calculado multiplicando-se o esforço (*EF*) pelo Valor médio (M) gasto por Unidade Objeto. Além disto, como no nosso exemplo em questão existe Reuso, o esforço será multiplicado por 2. Considerando o exemplo, a empresa tem custo de R$57.88 por Unidade Objeto. Então o cálculo será:

$$Custo = ((EF * 2) \ / FR) * VM$$

$$FC_{Tradicional} = ((637.32 * 2)/ 2) * 57.88 = R\$\ 36.888,08$$
$$FC_{SOA} = ((483.32 * 2) / 4) * 57.88 = R\$\ 13.987,18$$

- **Cálculo do Fator *Negócio***

Para calcular o fator negócio foram atribuídos os valores para *Business Context* e *Business Objectives* (Tabela 41):

Tabela 41: Aspecto Negócio

Contexto	**Descrição**	**Tradicional**	**SOA**
		Valor	**Valor**
Atividades do BPM	A funcionalidade está relacionado com atividades do BPM?	1	2
Pesquisa do mercado	A funcionalidade faz parte da prioridade do mercado?	5	2
Análise tendências,	A funcionalidade faz parte das tendências selecionada pela empresa?	2	3
BI e dados coletados	A funcionalidade faz parte do Roadmap?	3	5
Identificação de indicadores chave de desempenho	A funcionalidade melhora KPI da organização?	5	2
Objetivos	**Descrição**	**Tradicional**	**SOA**

		Valor	Valor
Identificação de objetivos de negócios	A implementação faz parte dos objetivos da Empresa?	3	3
Identificação de motivos	A funcionalidade faz parte dos motivos da adoção SOA?	2	5
Avaliar objetivo da adoção da SOA contra objetivos de negócio e nível de alinhamento com os objetivos de negócio	Qual é nível de alinhamento de serviço com objetivos de negócio?	5	5
Conformidade	A funcionalidade está conforme com os: – Aspectos legais: Existe leis ou normas para implementações das funcionalidades? Politicas internas: Existe regras alinhadas com o implementado na organização? (por exemplo, dados do cliente devem ser mantidos na estrutura interna ou há regras para manipulação dos dados dos clientes)	4	4

O Fator Negócio (FN) é calculado utilizando a média aritmética do valor total dos itens na Tabela 41. O valor máximo possível é cinco e o mínimo 0. Este intervalo define até que ponto o serviço é alinhado com as estratégias da organização.

$$FN = \sum valores\ dos\ itens\ /\ 9$$

$$FN_{Tradicional} = 30 / 9 = 3.33$$

$$FN_{SOA} = 31 / 9 = 3.44$$

A Tabela 42 apresenta o cálculo de todas as funcionalidades candidatas e os valores finais dos quatro fatores principais de tomada de decisão pelo gestor: *esforço, custo, tempo de desenvolvimento* e *grau de alinhamento ao negócio*, tanto para um desenvolvimento como software tradicional como para um baseado em serviços.

Para efeitos ilustrativos, essa tabela mostra o cálculo das demais funcionalidades do caso de uso, e não apenas o da funcionalidade *GBC* (*Gravar na Base de Conhecimento*).

Tabela 42: Resultado final dos 4 fatores

Serviço	Modo (T= Tradicional ou S=SOA)	Fator esforço (Homem/hora)	Fator Custo (R$)	Tempo desenvolvimento (horas)	Aspecto negocio
Gravar na Base Conhecimento	T	637,32	36888,16	1274,64	3,33
Gravar na Base Conhecimento	S	483,32	13987,18	966,63	3,44
Normalizar Texto	T	673,27	25979,20	1346,54	3,33
Normalizar Texto	S	461,47	17806,66	922,94	3,44
Buscar na Base Conhecimento	T	235,74	13644,47	235,74	2,11
Buscar na Base Conhecimento	S	217,42	8389,50	434,84	3,67
Reconhecer o texto	T	296,99	17189,62	296,99	1,56
Reconhecer o texto	S	208,55	6035,31	417,09	4,00

- **Cálculo Multicritério – O Ranqueamento das Funcionalidades**

Depois de calcular todos os fatores os tomadores de decisão podem refinar a análise ponderando pesos de importâncias entre aqueles quatro fatores entre si. Isto é considerado bastante importante como opção de análise na medida em que tomadas de decisão podem variar de acordo com cada demanda e cliente. Com isso, o ranqueamento inicial pode se alterar.

Para isto o método utiliza uma abordagem multicritério e adota método AHP.

A avaliação inicia pela determinação do peso relativo dos grupos dos fatores Custo, Esforço, Tempo de Desenvolvimento e Negócio, avaliados dois a dois. A Tabela 43 apresenta os dados de peso relativo entre os critérios para o exemplo utilizado. Estes pesos são definidos pela equipe executiva da empresa.

A linha *soma* representa o somatório dos fatores. Este valor é utilizado na Tabela 44 para calcular o percentual da importância de cada fator. Por exemplo, vê-se na tabela que *custo* é mais importante que os fatores *esforço* e *negócio*:

Tabela 43: Priorização dos Fatores

	Custo	Esforço	Tempo	Negócio
Custo	1	3	9	3
Esforço	1/3	1	5	1
Tempo	1/9	1/5	1	1/3
Negócio	1/3	1	3	1
Soma	**1,77**	**5,2**	**18**	**5,33**

Ainda de acordo com o método AHP, uma normalização é feita pela divisão entre cada valor individual com o total de cada coluna. A Tabela 44 apresenta o cálculo e os dados normalizados. A determinação da contribuição de cada critério na meta global é calculada a partir desses dados. O vetor *peso* apresenta os pesos relativos entre os critérios, obtidos através da média aritmética dos valores dos fatores.

Tabela 44: Vetor Prioridade

	Custo	Esforço	Tempo	Negócio	Peso (%)
Custo	1/1,77=0,57	3/5,2=0.58	9/18=0,5	3/5,33=0,56	55,1%
Esforço	0,3/1,77=0,17	1/5,2=0.19	5/18=0,28	1/5,33=0,19	21,1%
Tempo	0,1/1,77=0,05	0,2/5,2=0.04	1/18=0,05	0,33/5,33=0,06	5,4%
Negócio	0,3/1,77=0,28	1/5,2=0,19	3/18=0,17	1/5,33=0,19	18,3%

O valor do peso determina a participação daquele critério no resultado total da meta. Por exemplo, o critério custo tem um peso de 55.1% dentro da meta global.

Após isto é realizada a avaliação dois a dois entre as funcionalidades, baseado nos fatores Custo (Tabela 45), Esforço (Tabela 46), Tempo de Desenvolvimento (Tabela 47), e Aspectos de Negócio (Tabela 48). Nas tabelas a seguir, a última coluna representa o percentual da importância da cada uma das funcionalidades dentro do fator avaliado. Esse percentual é calculado da mesma forma que é calculado na Tabela 44. Por exemplo, no caso $GBC_{Tradicional}$, o cálculo será: (1,0/22,5) + (0,38/8,5) + (0,48/10,8) + (0,70/15,8) + (0,37/8,3) + (0,23/5,1) + (0,47/10,5) + (0,16/3,7) = 4,4.

Tabela 45: Aplicação AHP baseado fator Custo

Funcionalidades	GBC - T	GBC - S	NT- S	NT- T	BBC - T	BBC - S	RT - T	RT - S	Percentual - %
GBC - T	36888,16/ 36888,16 =1,0	13987,18/ 36888,16 =0,38	17806,66/ 36888,16 = 0,48	25979,20/ 36888,16 =0,70	13644,47/ 36888,16 =0,37	8389,50/ 36888,16 = 0,23	17189,62/ 36888,16 = 0,47	6035,31/ 36888,16 = 0,16	4,4
GBC - S	36888,16/ 13987,18 = 2,64	13987,18/ 13987,18 = 1,00	17806,66/ 13987,18 = 1,27	25979,20/ 13987,18 = 1,86	13644,47/ 13987,18 = 0,98	8389,50/ 13987,18 = 0,60	17189,62/ 13987,18 = 1,23	6035,31/ 13987,18 = 0,43	11,7
NT- S	36888,16/ 17806,66 =2,07	13987,18/ 17806,66 =0,79	17806,66/ 17806,66 =1,00	25979,20/ 17806,66 =1,46	13644,47/ 17806,66 =0,77	8389,50/ 17806,66 =0,47	17189,62/ 17806,66 =0,97	6035,31/ 13987,18 =0,34	9,2
NT- T	36888,16/ 25979,20 =1,42	13987,18/ 25979,20 =0,54	17806,66/ 25979,20 =0,69	25979,20/ 25979,20 =1,00	13644,47/ 25979,20 =0,53	8389,50/ 25979,20 =0,32	17189,62/ 25979,20 =0,66	6035,31/ 25979,20 =0,23	6,3
BBC - T	36888,16/ 13644,47 =2,70	13987,18/ 13644,47 =1,03	17806,66/ 13644,47 =1,31	25979,20/ 13644,47 =1,90	13644,47/ 13644,47 =1,00	8389,50/ 13644,47 =0,61	17189,62/ 13644,47 =1,26	6035,31/ 13644,47 =0,44	12
BBC - S	36888,16/ 8389,50 =4,40	13987,18/ 8389,50 =1,67	17806,66/ 8389,50 =2,12	25979,20/ 8389,50 =3,10	13644,47/ 8389,50 =1,63	8389,50/ 8389,50 =1,00	17189,62/ 8389,50 =2,05	6035,31/ 8389,50 =0,72	19,6
RT - T	36888,16/ 17189,62 =2,15	13987,18/ 17189,62 =0,81	17806,66/ 17189,62 =1,04	25979,20/ 17189,62 =1,51	13644,47/ 17189,62 =0,79	8389,50/ 17189,62 =0,49	17189,62/ 17189,62 =1,00	6035,31/ 17189,62 =0,35	9,5
RT - S	36888,16/ 6035,31 =6,11	13987,18/ 6035,31 =2,32	17806,66/ 6035,31 =2,95	25979,20/ 6035,31 =4,30	13644,47/ 6035,31 =2,26	8389,50/ 6035,31 =1,39	17189,62/ 6035,31 =2,85	6035,31/ 6035,31 =1,00	27,2
Soma	**22,5**	**8,5**	**10,8**	**15,8**	**8,3**	**5,1**	**10,5**	**3,7**	

Tabela 46: Aplicação AHP baseado fator Esforço

Funcionalidades	GBC - T	GBC - S	NT- S	NT- T	BBC - T	BBC - S	RT - T	RT - S	Percentual - %
GBC - T	1,00	0,76	0,72	1,06	0,37	0,34	0,47	0,33	6,45
GBC - S	1,32	1,00	0,95	1,39	0,49	0,45	0,61	0,43	8,52
NT- S	1,38	1,05	1,00	1,46	0,51	0,47	0,64	0,45	8,92
NT- T	0,95	0,72	0,69	1,00	0,35	0,32	0,44	0,31	6,11
BBC - T	2,70	2,05	1,96	2,86	1,00	0,92	1,26	0,88	17,46

BBC - S	2,93	2,22	2,12	3,10	1,08	1,00	1,37	0,96	18,93
RT - T	2,15	1,63	1,55	2,27	0,79	0,73	1,00	0,70	13,86
RT - S	3,06	2,32	2,21	3,23	1,13	1,04	1,42	1,00	19,74

Tabela 47: Aplicação AHP baseado fator Tempo de desenvolvimento

Funcionalidades	**GBC - T**	**GBC - S**	**NT- S**	**NT- T**	**BBC - T**	**BBC - S**	**RT - T**	**RT - S**	**Percentual - %**
GBC - T	1,00	0,76	0,72	1,06	0,18	0,34	0,23	0,33	4,92
GBC - S	1,32	1,00	0,95	1,39	0,24	0,45	0,31	0,43	6,49
NT- S	1,38	1,05	1,00	1,46	0,26	0,47	0,32	0,45	6,79
NT- T	0,95	0,72	0,69	1,00	0,18	0,32	0,22	0,31	4,66
BBC - T	5,41	4,10	3,92	5,71	1,00	1,84	1,26	1,77	26,59
BBC - S	2,93	2,22	2,12	3,10	0,54	1,00	0,68	0,96	14,42
RT - T	4,29	3,25	3,11	4,53	0,79	1,46	1,00	1,40	21,11
RT - S	3,06	2,32	2,21	3,23	0,57	1,04	0,71	1,00	15,03

Tabela 48: Aplicação AHP baseado Fator Aspecto de Negócio

Funcionalidade	**GBC - T**	**GBC - S**	**NT- S**	**NT- T**	**BBC - T**	**BBC - S**	**RT - T**	**RT - S**	**Percentual - %**
GBC - T	1,00	0,97	0,97	1,00	1,58	0,91	2,14	0,83	13,39
GBC - S	1,03	1,00	1,00	1,03	1,63	0,94	2,21	0,86	13,84
NT- S	1,03	1,00	1,00	1,03	1,63	0,94	2,21	0,86	13,84
NT- T	1,00	0,97	0,97	1,00	1,58	0,91	2,14	0,83	13,39
BBC - T	0,63	0,61	0,61	0,63	1,00	0,58	1,36	0,53	8,48
BBC - S	1,10	1,06	1,06	1,10	1,74	1,00	2,36	0,92	14,73
RT - T	0,47	0,45	0,45	0,47	0,74	0,42	1,00	0,39	6,25
RT - S	1,20	1,16	1,16	1,20	1,89	1,09	2,57	1,00	16,07

Com base nestes valores obtidos, observa-se que ainda não é possível determinar quais serviços atendem melhor aos objetivos da empresa. É então necessário atribuir prioridades com base nos valores calculados via AHP (Tabela 49):

Tabela 49: Aplicação dos pesos nas funcionalidades

	Custo	Esforço	Tempo	Negócio
GBC - T	4,4	6,45	4,92	13,39
GBC - S	11,7	8,52	6,49	13,84
NT- S	9,2	8,92	6,79	13,84
NT- T	6,3	6,11	4,66	13,39
BBC - T	12	17,46	26,59	8,48
BBC - S	19,6	18,93	14,42	14,73
RT - T	9,5	13,86	21,11	6,25
RT - S	27,2	19,74	15,03	16,07

x

Fator	%
Custo	55%
Esforço	21,1%
Tempo	5,4%
Negócio	18,3%

=

%
6,5%
11,1%
9,9%
7,5%
13,3%
18,2%
10,5%
22,9%

Esse resultado indica a porcentagem de relevância de cada funcionalidade candidata para o perfil da empresa. Considerando este exemplo, em SOA, a funcionalidade "Reconhecer o Texto - RT" apresentou o maior peso (22,9%) entre as demais analisadas (Tabela 50). Além disso, casualmente todas as funcionalidades no modo serviços SOA tiveram uma percentagem mais elevada em comparação com a mesma implementação no modo tradicional.

Tabela 50: Resultado final com percentual

Perc.	**Funcionalidade**	**Custo (R$)**	**Esforço (Homem/H)**	**Tempo (H)**	**Negócio**	**Reuso**	**Tam. Equipe**
22,9%	RT - Serviço	6035,31	208,55	417,09	4,00	4	1
18,2%	BBC - Serviço	8389,50	217,42	434,84	3,67	3	1
13,3%	BBC - Tradicional	13644,47	235,74	235,74	2,11	1	1
11,1%	GBC - Serviço	13987,18	483,32	966,63	3,44	4	1
10,5%	RT - Tradicional	17189,62	296,99	296,99	1,56	1	1
9,9%	NT- Serviço	17806,66	461,47	922,94	3,44	3	1
7,5%	NT- Tradicional	25979,20	673,27	1346,54	3,33	3	1
6,5%	GBC - Tradicional	36888,16	637,32	1274,64	3,33	2	1

6.4 EXEMPLO DE ESTIMAÇÃO *DA COMPISIÇÃO* DAS FUNCIONALIDADES *"GBC"* E *"NT"*

Conforme Erl (2009), estimar SOA pode ser baseado na avaliação de cada serviço isoladamente ou composto. O serviço composto é definido como um serviço formado por vários serviços, de maneira que eles sejam consumidos em uma atividade.

O Erl (2009) afirma que à medida que a sofisticação das soluções orientadas a serviços continuem aumentando há também o aumento da complexidade das configurações de composição dos serviços.

Esta complexidade pode dificultar a decisão do que é realmente mais vantajoso desenvolver, ou seja, a solução como serviço composto ou de desenvolver isoladamente. Neste sentido, o método permite a simulação em diferentes cenários, criando composições entre os serviços que atendam atividades do mesmo negócio, facilitando a decisão do avaliador. Este avaliador que inclusive pode tomar decisão de quais serviços isolados podem fazer parte da composição. O avaliador pode ser qualquer gestor de projeto, ou seja, gerente de projeto, gerente de portfólio, analista de negócio ou diretoria de tecnologia.

A seguir serão detalhados os mesmos cálculos da estimação do tópico anterior , porém, agora das funcionalidades GBC e NT em conjunto atendendo a opção 3 da Tabela 33. Tendo o resultado das estimações destas funcionalidades de modo isolado e agora composto, dará possibilidade do gestor avaliar se é mais vantajoso desenvolver as funcionalidades em conjunto (serviço com maior granularidade) ou isoladamente (serviço de granularidade 1).

- **Cálculo Fator *Complexidade da Integração***

Segue abaixo os dados do exemplo aplicado no fator complexidade da integração das funcionalidades GBC e NT (Tabela 51).

Tabela 51: Tipo de atores

Actor	**Weight**	**GBC**		**NT**	
		Value	***Result***	***Value***	***Result***
Simples	1	2	2	1	1
Média	2	0	0	0	0
Complexo	3	0	0	0	0

$$ICF_{GBC} = 2 + 0 + 0 = 2$$
$$ICF_{NT} = 1 + 0 + 0 = 1$$

- **Cálculo Fator *Requisitos Funcionais***

A Tabela 52 apresenta os dez itens a serem avaliados e os valores atribuídos em cada um deles.

Tabela 52: Complexidade dos requisitos funcionais

1- Tamanho da mensagem da requisição							
Faixa	***Descrição***	**GBC**			***NT***		
		Valor (1-3)	***Peso (0-5)***	***Res.***	***Valor (1-3)***	***Peso (0-5)***	***Res.***
1	DatasetSize<=1	1	1	1	1	1	1
2	2<= Dataset Size<=3						
3	DatasetSize>3						
2- Tamanho da mensagem da resposta							
Faixa	***Descrição***	**GBC**			***NT***		
		Valor (1-3)	***Peso (0-5)***	***Res.***	***Valor (1-3)***	***Peso (0-5)***	***Res.***
1	DatasetSize <= 1	1	3	3	1	1	1
2	2<= DatasetSize <= 3						
3	DatasetSize > 3						
3- Complexidade de tratamento de dados na requisição/resposta							
Faixa	***Descrição***	**GBC**			***NT***		
		Valor (1-3)	***Peso (0-5)***	***Res.***	***Valor (1-3)***	***Peso (0-5)***	***Res.***
1	Não há necessidade de tratamento de dados	3	5	15	3	4	12
2	Tratamento de dados apenas em um sentido (requisição ou resposta)						
3	Tratamento de dados em dois sentidos (requisição e resposta)						
4- Complexidade de lógica de negócio							
Faixa		**GBC**			***NT***		
		Valor (1-3)	***Peso (0-5)***	***Res.***	***Valor (1-3)***	***Peso (0-5)***	***Res.***
1=Simples, 2=Média, 3=Complexo		3	5	15	3	4	12

5- O número de objetos manipulados pela lógica							
Faixa	*Descrição*	**GBC**			***NT***		
		Valor (1-3)	*Peso (0-5)*	*Res.*	*Valor (1-3)*	*Peso (0-5)*	*Res.*
1	Domain Objetos<=2	3	4	12	1	4	4
2	3<= Domain Objetos<=5						
3	Domain Objetos>5						
6- Complexidade de acesso aos dados							
Faixa	*Descrição*	**GBC**			***NT***		
		Valor (1-3)	*Peso (0-5)*	*Res.*	*Valor (1-3)*	*Peso (0-5)*	*Res.*
1	Não há necessidade de gravar dados (somente leitura)	3	4	12	2	3	6
2	Grava dados ou somente leitura						
3	Precisa executar várias operações de leitura e gravação de dados						
7- Complexidade da integração com outros serviços							
Faixa	*Descrição*	**GBC**			***NT***		
		Valor (1-3)	*Peso (0-5)*	*Res.*	*Valor (1-3)*	*Peso (0-5)*	*Res.*
1	Não tem interação com outros serviços (serviço atômico)	2	2	4	2	2	4
2	Necessita realizar a busca de serviço a ser chamado através de séries						
3	Necessita de chamado de outro serviço transacional						
8- Complexidade de tratamento da falha e do erro							
Faixa	*Descrição*	**GBC**			***NT***		
		Valor (1-3)	*Peso (0-5)*	*Res.*	*Valor (1-3)*	*Peso (0-5)*	*Res.*
1	Não há necessidade de tratamento	2	3	6	2	3	6
2	Erros devem ser relatados dentro do escopo do serviço						
3	Erros devem ser reportados via interface externo						
9- Complexidade da infraestrutura da integração							
Faixa	*Descrição*	**GBC**			***NT***		

		Valor (1-3)	*Peso (0-5)*	*Res.*	*Valor (1-3)*	*Peso (0-5)*	*Res.*
1	Através API bem definido	2	1	2	2	1	2
2	Atraves API técnico , exemplo: JMS, Apache CXF API.						
3	Atraves de interface específico do produto						
10- Quantidade de campos envolvidos							
Faixa	***Descrição***	**GBC**			***NT***		
		Valor (1-3)	*Peso (0-5)*	*Res.*	*Valor (1-3)*	*Peso (0-5)*	*Res.*
7	1<= Quantidade <= 19	7	5	35	7	3	35
10	20<= Quantidade<= 50						
15	50<= Quantidade						

$FRF_{GBC} = 105$
$FRF_{NT} = 83$

- **Cálculo Fator *Complexidade Funcional***

Para calcular a complexidade funcional são somados os valores de complexidade de requisitos funcionais e da dos atores.

$FCF_{GBC} = 2 + 105 = 107$
$FCF_{NT} = 1 + 83 = 84$

- **Cálculo Fator *Requisitos Não Funcionais***

A Tabela 53, apresenta os requisitos não funcionais e os valores atribuídos a cada um deles.

Tabela 53: Requisitos não Funcionais

Nome	**Peso(0-5)**	
	GBC	***NT***
Tempo de resposta	3	3
Concorrência	5	4
Escalabilidade	2	3
Segurança	2	3

Monitoramento de serviço	2	2
Latência	2	2
Taxa da transferência de dados	2	2
Disponibilidade	2	2
Confiabilidade	2	5
Precisão	4	3
Robustez	4	4
Estabilidade	5	5
Mensagens Confiáveis	1	2
Integridade	0	2
Interoperabilidade	3	4
Fraco acoplamento	3	5
Abstração	5	5
Sem estado	5	3
Total (TNFR)	**52**	**59**

Como já explanado, o valor total TRNF precisa ser ajustado. No caso, considerando que a quantidade de requisitos não funcionais é 18, o coeficiente da formula de ajuste é = 0.0077:

$$FRNF_{GBC}=(0.0077*52)+0,65 = 1,056$$

$$FRNF_{NT}= (0.0077*59)+0,65 = 1,110$$

- **Cálculo do Fator *Tamanho***

O tamanho do software é calculado por meio da seguinte fórmula e o resultado baseado nos valores do exemplo:

$$FTAM_{GBC} = 1,054 * 107= 112,83$$

$$FTAM_{NT} = 1,109 * 84 = 93,15$$

- **Cálculo Fator *Experiência dos Profissionais***

Considerando a Tabela 54 para calcular o fator complexidade funcional, os valores da coluna "*result*" são somados.

Tabela 54: Experiência dos profissionais

Descrição	**GBC**			**NT**		
	Peso	***Valor (0-5)***	***Result***	***Peso***	***Valor (0-5)***	***Result***
Equipe tem conhecimento no processo de desenvolvimento	1,5	3	4,5	1,5	3	4,5
Experiência da equipe com o tipo de	0,5	2	1	0,5	2	1

aplicação e negócio						
Experiência da equipe com orientação a objetos	1	3	3	1	3	3
Capacidade de análise	0,5	2	1	0,5	2	1
Motivação da equipe	1	2	2	1	2	2
Trabalhadores em tempo parcial	-1	1	-1	-1	1	-1
Dificuldade com a linguagem de programação escolhida	-1	4	-4	-1	4	-4

O fator experiência dos profissionais (FEP) é calculado por meio da seguinte formula:

$$FEP_{GBC} = 6,5$$
$$FEP_{NT} = 6,5$$

- **Cálculo Fator *Condição Ambiental***

Os dados do exemplo aplicado às duas funcionalidades a serem avaliadas são (Tabela 55):

Tabela 55: Condição Ambiental

Descrição	**GBC**			**NT**		
	Peso	***Valor (0-5)***	***Res.***	***Peso***	***Valor (0-5)***	***Res.***
Existe Governança SOA na organização	1	5	5	1	4	4
Existe ferramenta e plataforma adequado para desenvolvimento	1	4	4	1	3	3
Existe descoberta dinâmica por serviços (services discovery)?	1	5	5	1	4	4
Organização tem estrutura ESB?	1	3	3	1	3	3
Estabilidade dos requisitos	2	2	4	2	2	4

$$FCA_{GBC} = 21$$
$$FCA_{NT} = 18$$

- **Cálculo Fator *Complexidade Ambiental***

Este fator é calculado baseado na seguinte fórmula:

$$CAM_{GBC} = 6,5 + 21 = 27,5$$
$$CAM_{NT} = 6,5 + 18 = 24,5$$

O valor precisa ser ajustado por meio da fórmula abaixo:

$$FCAM_{GBC} = 1,4 + (-0,03 * 27,5) = 0,575$$
$$FCAM_{NT} = 1,4 + (-0,03 * 24,5) = 0,665$$

- **Cálculo Fator *Reuso***

O valor do reuso corresponde ao número de funcionalidades dos outros Casos de Uso que podem usar essa função:

$$FR_{GBC} = 4$$
$$FR_{NT} = 3$$

- **Cálculo Fator *Esforço***

O fator de esforço é calculado multiplicando a produtividade estimada pelo tamanho do serviço em Ponto de Função. No exemplo, a produtividade é igual 7,45 homens/hora. O resultado do calculo do é:

$$FE_{GBC} = 112,83 * 0,575 * 7,45 = 483,31$$
$$FE_{NT} = 93,15 * 0,665 * 7,45 = 461,47$$

O esforço total das 2 funcionalidades é o somatório dos valores:

$$FE_{GBC + NT} = 483,31 + 461,47 = 944,79$$

- **Cálculo Fator *Custo***

O fator Custo (FC) é calculado multiplicando-se o esforço (*EF*) pelo valor médio (M) gasto por Unidade Objeto. Além disto, como no exemplo existe reuso, o esforço será multiplicado por 2. No exemplo a empresa tem custo de R$57.88 por Unidade Objeto:

$$FC_{GBC} = ((483,31 * 2) * 57,88) / 4 = R\$\ 13.987,18$$
$$FC_{NT} = ((461,47 * 2) * 57,88) / 3 = R\$\ 17.806,66$$

O custo total das 2 funcionalidades é somatório dos valores:

$$FC_{GBC + NT} = 13987,18 + 17806,66 = R\$\ 31.793,84$$

- **Cálculo Fator *Negócio***

Como a estimação é referente à composição de duas funcionalidades que serão desenvolvidas em conjunto para calcular o

fator negócio, os pesos a serem atribuídos precisam levar em consideração qual é a relevância e o alinhamento destes dois serviços com aspecto negócio. Por isto, será necessário novamente o gestor atribuir valores para fatores *Business Context*, *Business Objectives* e tamanho de equipe que vão estar envolvidos na implementação destas duas funcionalidades (Tabela 56):

Tabela 56: Aspecto Negócio

Contexto	**GBC/NT**
	Valor
Atividades do BPM	5
Pesquisa do mercado	5
Análise tendências,	5
BI e dados coletados	5
Identificação de indicadores chave de desempenho	5
Objetivas	**GBC/NT**
	Valor
Identificação de objetivos de negócios	4
Identificação de motivos	5
Avaliar objetivo da adoção da SOA contra objetivos de negócio e nível de alinhamento com os objetivos de negócio	5
Conformidade	5

O fator Negócio (FN) é calculado utilizando a média aritmética do valor total dos itens na tabela. O valor máximo seria 5 e valor mínimo 0. Este intervalo define até que ponto o serviço é alinhado com estratégias da organização:

$$FN_{GBC/NT} = 44 / 9 = 4,89$$

- **Cálculo Fator *Tempo de Desenvolvimento***

Como as duas funcionalidades serão implementadas em conjunto o esforço é o somatório do esforço das duas funcionalidades:

$$FE_{GBC+NT} = 483,31 + 461,47 = 944,79$$

Além disto, há necessidade de novamente apontar o tamanho de equipe. No caso do exemplo, o tamanho de equipe envolvida para implementação de duas funcionalidades são de 3 pessoas:

$$FTD_{hora\ GBC} = (944,78 * 2) / 3 = 629,86\ horas$$

A Tabela 57 apresenta o cálculo de todas as funcionalidades candidatas e valores finais dos quatro fatores. Baseado nestes valores será aplicada o método AHP para ranquear as funcionalidades.

Tabela 57: Resultado dos 4 fatores

Serviço	Modo (T= Tradicional ou S=SOA)	Fator esforço (Homem/hora)	Fator Custo (R$)	Tempo desenvolvimento (horas)	Aspecto negocio
Gravar na Base Conhecimento	T	637,32	36888,16	1274,64	3,33
Normalizar Texto	T	673,27	25979,20	1346,54	3,33
Buscar na Base Conhecimento	T	235,74	13644,47	235,74	2,11
Buscar na Base Conhecimento	S	217,42	8389,50	434,84	3,67
Reconhecer o texto	T	296,99	17189,62	296,99	1,56
Reconhecer o texto	S	208,55	6035,31	417,09	4,00
Gravar na Base Conhecimento + Normalizar Texto - Serviço	S	944,79	31793,84	629,86	4,89

- **Cálculo Multicritério – O Ranqueamento das Funcionalidades**

Assim como no caso anterior (Seção 6.3), depois de calcular todos os fatores os tomadores de decisão podem refinar a análise ponderando pesos de importâncias entre aqueles quatro fatores entre si, o que é feito utilizando uma abordagem multicritério e adotando o método AHP.

Para este análise serão utilizadas as mesmas tabelas utilizadas no tópico anterior.

Segue abaixo a avaliação dois a dois entre as funcionalidades, baseado nos fatores Custo (Tabela 58), Esforço (Tabela 59), Tempo de desenvolvimento (Tabela 60) e Aspectos de Negócio (Tabela 61). Nas

tabelas a seguir a última coluna representa o percentual da importância da cada uma das funcionalidades dentro do fator avaliado.

Tabela 58: Aplicação AHP baseado Fator Custo

Funcionalidades	GBC - T	GBC - S	NT- S	NT- T	BBC - T	BBC - S	RT - T	Percentual - %
GBC - T	1,00	0,70	0,37	0,23	0,47	0,16	0,86	5,28%
NT- T	1,42	1,00	0,53	0,32	0,66	0,23	1,22	7,50%
BBC - T	2,70	1,90	1,00	0,61	1,26	0,44	2,33	14,28%
BBC - S	4,40	3,10	1,63	1,00	2,05	0,72	3,79	23,22%
RT - T	2,15	1,51	0,79	0,49	1,00	0,35	1,85	11,33%
RT - S	6,11	4,30	2,26	1,39	2,85	1,00	5,27	32,27%
NT + GBC - Serviço	1,16	0,82	0,43	0,26	0,54	0,19	1,00	6,13%

Tabela 59: Aplicação AHP baseado Fator Esforço

Funcionalidades	GBC - T	GBC - S	NT- S	NT- T	BBC - T	BBC - S	RT - T	Percentual - %
GBC - T	1,00	1,06	0,37	0,34	0,47	0,33	1,48	7,43%
NT- T	0,95	1,00	0,35	0,32	0,44	0,31	1,40	7,03%
BBC - T	2,70	2,86	1,00	0,92	1,26	0,88	4,01	20,09%
BBC - S	2,93	3,10	1,08	1,00	1,37	0,96	4,35	21,78%
RT - T	2,15	2,27	0,79	0,73	1,00	0,70	3,18	15,95%
RT - S	3,06	3,23	1,13	1,04	1,42	1,00	4,53	22,71%
NT + GBC - Serviço	0,67	0,71	0,25	0,23	0,31	0,22	1,00	5,01%

Tabela 60: Aplicação AHP baseado Fator Tempo de desenvolvimento

Funcionalidades	GBC - T	GBC - S	NT- S	NT- T	BBC - T	BBC - S	RT - T	Percentual - %
GBC - T	1,00	1,06	0,18	0,34	0,23	0,33	0,49	5,09%
NT- T	0,95	1,00	0,18	0,32	0,22	0,31	0,47	4,82%
BBC - T	5,41	5,71	1,00	1,84	1,26	1,77	2,67	27,51%
BBC - S	2,93	3,10	0,54	1,00	0,68	0,96	1,45	14,91%
RT - T	4,29	4,53	0,79	1,46	1,00	1,40	2,12	21,83%
RT - S	3,06	3,23	0,57	1,04	0,71	1,00	1,51	15,55%
NT + GBC - Serviço	2,02	2,14	0,37	0,69	0,47	0,66	1,00	10,30%

Tabela 61: Aplicação AHP baseado Fator Aspecto de Negócio

Funcionalidade	GBC - T	GBC - S	NT- S	NT- T	BBC - T	BBC - S	RT - T	Percentual - %
GBC - T	1,00	1,00	1,58	0,91	2,14	0,83	0,68	14,56%
NT- T	1,00	1,00	1,58	0,91	2,14	0,83	0,68	14,56%
BBC - T	0,63	0,63	1,00	0,58	1,36	0,53	0,43	9,22%
BBC - S	1,10	1,10	1,74	1,00	2,36	0,92	0,75	16,02%
RT - T	0,47	0,47	0,74	0,42	1,00	0,39	0,32	6,80%
RT - S	1,20	1,20	1,89	1,09	2,57	1,00	0,82	17,48%
NT + GBC - Serviço	1,47	1,47	2,32	1,33	3,14	1,22	1,00	21,36%

Com base nestes valores obtidos, pode-se notar que ainda não é possível determinar quais serviços atendem melhor aos objetivos da empresa. É então necessário atribuir prioridades com base nos valores calculados via AHP (Tabela 62):

Tabela 62: Aplicação dos pesos nas funcionalidades

	Custo	Esforço	Tempo	Negócio
GBC - T	5,28	7,43	5,09	14,56
NT- T	7,50	7,03	4,82	14,56
BBC - T	14,28	20,09	27,51	9,22
BBC - S	23,22	21,78	14,91	16,02
RT - T	11,33	15,95	21,83	6,80
RT - S	32,27	22,71	15,55	17,48
NT + GBC - Serviço	6,13	5,01	10,30	21,36

X

Fator	%
Custo	55%
Esforço	21,1%
Tempo	5,4%
Negócio	18,3%

=

%
7,4%
8,6%
15,3%
21,2%
12,1%
26,6%
8,9%

Esse resultado indica a porcentagem de relevância de cada funcionalidade candidata para o perfil da empresa. Considerando este exemplo, em SOA, o "Reconhecer o Texto - RT" apresentou o maior peso (26.9%) entre as demais aplicações analisadas (Tabela 62). A Tabela 63 ilustra os valores de forma ordenada, para facilitar a análise.

Tabela 63: Resultado final com percentual

Perc.	**Funcionalidade**	**Custo (R$)**	**Esforço (Homem/H)**	**Tempo (H)**	**Negócio**	**Reuso**	**Tam. Equipe**
26,6%	RT - Serviço	6035,31	208,545	417,091	4,00	4	1,0
21,2%	BBC - Serviço	8389,50	217,420	434,839	3,67	3	1,0
15,3%	BBC - Tradicional	13644,47	235,737	235,737	2,11	1	1,0
12,1%	RT - Tradicional	17189,62	296,987	296,987	1,56	1	1,0
8,9%	NT + GBC - Serviço	31793,84	944,788	629,859	4,89	--	3,0
8,6%	NT- Tradicional	25979,20	673,269	1346,538	3,33	3	1,0
7,4%	GBC – Tradicional	36888,16	637,321	1274,643	3,33	2	1,0

6.5 COMPARAÇÃO DOS RESULTADOS DO EXEMPLO

Analisando a possibilidade de implementar as funcionalidades isoladamente ou algumas "compostos" uma com outras, a Tabela 64 apresenta a comparação entre os valores e percentuais nas duas situações diferentes analisadas.

Tabela 64: Comparação entre implementação isolada x composto

Funcionalidade Com composição	**Percentual**
RT - Serviço	26,6%
BBC - Serviço	21,2%
BBC - Tradicional	15,3%
RT - Tradicional	12,1%
NT + GBC - Serviço (composto)	8,9%
NT- Tradicional	8,6%
GBC - Tradicional	7,4%

Por meio da análise da Tabela 64, percebe-se que perante a organização, bem como pelos valores informados na tabela multicritério, é mais vantajoso desenvolver a funcionalidade "RT" no modo Serviço (o maior percentual, 26,6%) do que desenvolver RT no modo tradicional (12,1%). Além disto, percebe-se que é mais vantajoso implementar as funcionalidades NT (serviço) e GBC (serviço) de forma composto (8,9%) do que as implementar isoladamente (com 8,6% e 7,4% de alinhamento ao negócio, respectivamente).

Importante mencionar que os valores calculados podem ser comparados ou servirem de referência quando de eventuais terceirizações, apoiando a decisão de possível desenvolvimento por terceiros.

Na perspectiva de planejamento de gestão de projetos, deve ser feita uma análise de recursos humanos e financeiros Gitman (2010). Isto parte do princípio econômico usado em administração e usado para tomadas de decisões na escolha de projetos a serem implementados, de que uma empresa inicia um projeto desde que tenha recursos disponíveis para sua finalização.

Neste sentido, é possível que a empresa não possua recursos financeiros para implementar todos os serviços. Outra questão é a alocação de pessoal para atuar no projeto, que normalmente são escassos e disputados entre diversos projetos e atividades operacionais, tornando necessário também balancear a utilização desses recursos (ZIMMERMANN; KROGDAHL; GEE, 2004).

Assim sendo, e conforme os cálculos do método proposto sobre custo e esforço, a empresa analisa seus recursos financeiros e humanos disponíveis no momento, verificando quais funcionalidades candidatas são viáveis de serem implementadas, quer como serviços (forma uma perspectiva SOA) ou um software tradicional.

Neste caso, a empresa, iniciando pela funcionalidade-candidata com maior percentual de aderência, vai somando quantidade de recursos humanos (fator esforço) e recursos financeiro (fator custo) necessários para a implementação da funcionalidade em questão com as outras funcionalidades da lista. Com isso a empresa tem uma base para definir prioridades perante seus recursos disponíveis e então definir quantas e quais funcionalidades mensuradas e ranqueadas podem ser desenvolvidas e ao final se tornarem ativos de software da empresa.

Por exemplo, considerando que a empresa tivesse 7 funcionalidades em avaliação R$ 30 mil de recursos financeiros e 15 homens/dia disponíveis. Porém, como a terceira e quarta opções se referem às funcionalidades BBC e RT no modo tradicional e elas seriam desenvolvidas como serviço (primeira e segunda opções), o gestor pode "pular" para a quinta opção, que é das duas funcionalidades compostas. Porém, mesmo a empresa tendo os recursos humanos disponíveis, os recursos financeiros disponíveis seriam insuficientes, menores que o somatório dos custos de três funcionalidades "RT-Serviço", "BBC-Serviço" e "NT+GBC-Serviço" (R$ 401,836.34). Ficaria a critério da empresa aumentar a verba para completar a implementação das três funcionalidades, ou apenas autorizar a implementação das duas primeiras funcionalidades.

7 AVALIAÇÃO DO MÉTODO

A fim de avaliar se a proposta do método de estimativa proposto é aderente aos objetivos desta tese, este capítulo visa apresentar os procedimentos de avaliação bem como os resultados obtidos.

Como descrito no Capítulo 2, da Metodologia de Pesquisa, os procedimentos procuraram ser coerentes com o projeto da pesquisa e com a avaliação incremental e participativa de empresas e de especialistas desde o seu início. E ao se avaliar o modelo em si e seus vários elementos, procurou-se tanto avaliar a corretude das ações efetuadas para a concepção do método como, ao final, verificar se a pergunta da tese fora respondida e seus objetivos atingidos.

Dado que se trata de um trabalho aplicando o método *Design Science*, procurou-se envolver típicos "usuários do método" (que na tese são genericamente chamados de 'gestores de TI') desde o início, de forma a ajudarem nos refinamentos do método e sua verificação em ciclos de evolução. Neste sentido, a avaliação desta tese foi feita via quatro procedimentos metodológicos, descritos nas seções a seguir.

7.1 PUBLICAÇÃO DE ARTIGOS NO MEIO CIENTÍFICO

Foram aceitos e publicados dois artigos científicos em renomadas conferências internacionais diretamente relacionadas à área da tese, permitindo uma avaliação do método pela comunidade científica especializada.

Os artigos foram os seguintes:

- Ghoddosi, N. ; Rabelo, R. J., *A Method for Evaluating the Feasibility of SOA Projects*. Anais 12th IEEE International Conference on Service Systems and Service Management, Guangzhou, China, 2015. p. 1-8.
- Ghoddosi, N. ; Rabelo, R. J., *A systematized analysis process for SOA services estimation*. Anais 11th International Conference on Computer Engineering & Systems (ICCES), Cairo, Egito, 2016. p. 384-389.

7.2 VERIFICAÇÃO

Esta etapa visou testar o método em condições totalmente controladas pelo autor, onde se buscou aplicar o método em inúmeros casos de uso (concebidos pelo próprio autor, desde simples a muito complexos) e verificar se ele de fato estaria calculando todos os fatores de forma correta. Neste processo algumas correções foram efetuadas.

No escopo desta etapa houve também o envolvimento direto de profissionais. Em meados de 2013 foram contatadas três empresas de TI de Blumenau com o objetivo de avaliarem o método, que naquela altura estava ainda na sua "primeira versão". No entanto, apenas uma empresa acabou por ser selecionada para um contato mais aprofundado devido à sua maturidade em SOA, estrutura e qualidade de seu legado, tendo mais conhecimento para analisar o método e sugerir melhorias no que fosse necessário.

Dois workshops foram realizados nessa empresa no final de 2013 (Figura 53), coordenado pelo autor desta tese.

Figura 53: Foto de um workshop realizado na empresa escolhida
Fonte: Autor

Cada *workshop* teve uma duração aproximada de 4 horas, incluindo um tempo de explicação do método junto a analistas de sistemas, analistas de negócio, arquitetos de TI, gerentes de projetos e programadores, num total de 11 a 15 pessoas, todas com experiência nas suas áreas de atuação.

Esta heterogeneidade também foi usada para emular um ambiente com perfis de profissionais abrangente, como é o que acaba ocorrendo ao se conceber uma solução real SOA.

O método foi apresentado na forma de *slides* e a seguir foi discutido. Cada profissional pode expor seus pontos de vistas e discutir entre os participantes. Na moderação do workshop procurou-se entender em mais profundidade como os gestores de TI tomavam decisões sobre SOA, os requisitos de um método de estimação SOA assim como os critérios usados para tal. Ao final, vários aspectos do método e da sua importância/utilidade foram ratificados e algumas contribuições foram dadas e posteriormente incorporadas ao método.

Importante ressaltar que comentários dos três revisores de cada um daquelas conferências (Seção 7.1) também foram considerados e de alguns destes resultou igualmente em melhorias no método.

7.3 AVALIAÇÃO

Considerando que o método estava preliminarmente correto, ele foi submetido para uso em condições perto do real. Isto ocorreu entre outubro de 2016 e agosto de 2017 junto a uma empresa sediada na cidade do Rio de Janeiro e que tem sólido departamento de TI, além de experiência e maturidade em SOA.

A empresa aplicou o método em quatorze dos seus Casos de Uso reais, de simples a complexos. Houveram várias interações com um dos analistas-chefe de TI da empresa e este autor. Ao final alguns refinamentos foram efetuados no método. Devido a distância geográfica, a comunicação ocorreu via correio eletrônico (mais de 70 mensagens) e *Skype* (aproximadamente 6 horas de conversas).

Complementarmente, o método foi aplicado a um grupo de 30 alunos de pós-graduação em Automação da UFSC que fizeram a disciplina de SOA do curso assim como avaliado por dois experientes pesquisadores do GSIGMA, grupo de pesquisas coordenado pelo orientador desta tese. Da mesma forma, ao final destas alguns comentários foram fornecidos e alguns deles acabaram por ser introduzidos no método.

Todos esses ciclos de melhoramentos (*fluxos de conhecimento*) estão previstos na metodologia *Design Science*.

Essas duas primeiras estapas, de *verificação* e *avaliação*, visaram fundamentalmente averiguar a *corretude* em si do método, ou seja, observar as eventuais faltas, falhas e erros técnicos no método.

Após considerar que o método estava pronto, partiu-se para a última etapa, de *validação*.

7.4 VALIDAÇÃO

Esta etapa visou averiguar o cumprimento dos objetivos da tese bem como sua proposição de valor.

A definição de 'validação' é um tanto variável na literatura e difere de 'verificação'. Nesta tese três definições básicas serviram de referencial para se considerar esta etapa como *validação*.

Sucintamente, Gil (2010) define *verificar* como testar para ver se ficou conforme o planejado, enquanto que *validar* é testar para ver se o que foi feito vai funcionar para o propósito desejado. Num contexto de software, para Pressman (2016), validação significa um conjunto de atividades que garante que o software construído corresponde aos requisitos do cliente/negócio/objetivo final. Para Summerville (2011), é demonstrar que o software atende às expectativas dos clientes, à real necessidade final. Portanto, nesta teste a etapa de validação visa comprovar que o método de estimação para SOA atende aos objetivos para os quais foi criado, tanto do ponto de vista de requisitos e corretude (expressos no Capítulo 5) como de proposição de valor (listados no Capítulo 1).

Como explanado no Capítulo 1, a adoção de SOA em sua plenitude é um "processo", gradual, de alguma complexidade e geralmente bastante longo, e impacta uma organização em diversos níveis, fazendo com que empresas que o adotam acabem por ter variados níveis de "maturidade" em SOA.

Como também ressaltado no Capítulo 1, nesta tese tem se enfatizado que adotar SOA não é simplesmente ter apenas um pequeno conjunto de softwares tipo *web services*, invocados numa ótica de integração par-a-par de forma equivalente a uma *API*. Ao contrário, que tais softwares em uma perspectiva SOA são na verdade um dos resultados: de um mais complexo e amplo desdobramento de visão estratégica de TI (e de negócios) de uma empresa, englobando tanto aspectos de ciclo de vida SOA como de governança SOA; de uma ótica de integração desacoplada e baseada em reuso, incluindo os inúmeros aspectos de interoperabilidade com sistemas legados e distribuídos; e da existência de inúmeras TIs heterogêneas e de variadas gerações tecnológicas.

Portanto, o método foi primordialmente concebido para este tipo de desenvolvimento de serviços / cenário mais complexo e completo SOA, que, na verdade, não é ainda o mais usual de ser encontrado nas empresas que têm começado a adotar SOA.

Além disso, a adoção de SOA por empresas é relativamente recente, de forma que o tamanho da amostra viável para realização de validações não pôde ser muito grande, considerando também a viabilidade e custos de aplicar o método em outras localidades do estado de Santa Catarina e em outros estados.

Verificou-se ainda que 80% das empresas contatadas para possibilidade de validação do método estavam na verdade num estágio muito inicial de SOA, isso quando não se enquadravam naquele cenário simplista de *web services* acima relatado. Portanto, não apenas não empregavam práticas mais formais de estimação, como não tinham conhecimento suficiente para apropriadamente considerar os inúmeros aspectos afetos aos vários fatores do método desenvolvido; ou seja, não eram empresas suficientemente preparadas para avaliar o método. Algumas inclusive afirmaram diretamente que não teriam interesse no uso do método ao perceberem a amplitude dos aspectos a serem considerados e verificarem que suas empresas estavam apenas no início da "cultura" de serviços de software. Outras empresas alegaram problemas de agenda e excesso de trabalho das pessoas que poderiam avaliar para não aceitar o pedido de validação, mesmo após esclarecimentos sobre as potenciais vantagens do método para a empresa e que ela poderia usar os seus próprios casos de uso para avaliar.

A literatura apresenta vários métodos para a avaliação e validação de trabalhos relacionados a tecnologia (ZELKOWITZ, 2007). Dado que não se tratava de uma pesquisa tipo 'Estudo de Caso' (de comparação entre o estado anterior do processo numa dada empresas e o como teria ficado após a plena implantação do artefato desenvolvido), que não seria viável efetuar um estudo longitudinal de avaliação do método após ser usado por empresas após um longo período de tempo, e diante das dificuldades acima mencionadas, optou-se por aplicar a técnica *Expert Panel*.

O *Expert Panel* serve para validar estudos baseados no consenso de especialistas (ZELKOWITZ, 2007), ditos *experts* no assunto e, assim, consideradas aptas a avaliar o trabalho. É adequada para avaliações classificadas como "*self-assessment*", onde a empresa faz uma auto avaliação (EL-MAADDAWY, 2017) (TUDEVDAGVA; LKHAGVASUREN, 2016).

As características básicas da técnica são (ZELKOWITZ, 2007):

- Contexto é controlado – os procedimentos utilizados para se apresentar o objeto de avaliação são previamente estabelecidos, as premissas são conhecidas e a intervenção de ações é temporal;
- Os dados são coletados a partir dos especialistas – especialistas são a única fonte de dados do experimento e há alguma subjetividade nas respostas;
- Aplicação no contexto real – deve-se efetuar a avaliação considerando o ambiente ou condições que possam de alguma forma reproduzir cenários reais do objeto da avaliação.

Em termos gerais, a técnica é executada em três etapas principais: i) escolha criteriosa dos experts (incluindo a quantidade deles); ii) coleta de respostas dos experts (incluindo os procedimentos e meios de coleta); e iii) análise das respostas (incluindo o tratamento estatístico delas).

Sobre a primeira etapa, e como esclarecido acima, a amostra acabou por ter que ficar reduzida, privilegiando-se a maturidade SOA das empresas. Após inúmeras tentativas e contatos, apenas três empresas puderam participar da etapa da validação, sendo a primeira do Rio de Janeiro e as duas últimas de Florianópolis. As duas primeiras são de grande porte e a terceira de médio porte. As duas primeiras tem já um sólido conhecimento em SOA enquanto a terceira um menor conhecimento em relação a essas duas. A primeira não é uma empresa de software e tem um grande departamento de TI enquanto que as duas demais são de desenvolvimento de produtos de software e de prestação de serviços associados.

Sobre a segunda etapa, o instrumento utilizado foi um *questionário*, composto de um conjunto ordenado de perguntas para os especialistas.

Questionário é um instrumento de investigação via coleta de informação utilizado em sondagens ou inquéritos. É composto por um número de questões apresentadas por escrito com o objetivo de propiciar determinado conhecimento ao pesquisador (GIL 2010). Um questionário deve ser objetivo, limitado em extensão e estar acompanhado de instruções. As perguntas que compõem o questionário devem estar atreladas aos objetivos específicos do trabalho (GIL, 2002).

O questionário foi do tipo estruturado que, em suma, visou fazer refletir nas perguntas os vários aspectos desejados de serem avaliados no método (as *medidas de desempenho*, conforme o método *Design*

Science), considerando os seus objetivos e proposição de valor.

Antes de aplicar o questionário este autor esteve em contato direto, seja pessoalmente, via e-mail ou via *Skype*, com as três empresas para que o método aqui proposto fosse devidamente compreendido e, assim, que as respostas do questionário fossem efetivamente realistas às empresas, aos seus casos de negócio. Cada contato teve um tempo médio de 2 hora e foram realizadas de 3 a 4 vezes em cada empresa.

Nesses contatos foi feita a explicação do método e executada a sua aplicação em diferentes cenários. Além da explicação, o autor do método sugeriu cenários de aplicação, dando indicativos aos entrevistados sobre as variedades desejadas de situações (casos de uso) a serem avaliadas com o uso do método.

Algumas perguntas foram formuladas de forma a serem respondidas dentro de 'categorias', adotando-se a consolidada *Escala Likert* (LIKERT, 1932, apud GLIEM, 2003). Essa escala é utilizada em pesquisas de opinião e permite extrair diferentes níveis de concordância para cada uma das afirmativas, permitindo aferir em que medida o entrevistado concorda com a afirmativa apresentada e quantificar a concordância dos entrevistados com os objetivos em questão.

Outras perguntas do questionário foram feitas de forma aberta, onde o entrevistado pôde responder com suas próprias palavras de forma a captar aspectos subjetivos da opinião dos avaliadores.

Para apoiar a montagem dos questionários utilizou-se o método GQM (*Goal, Question, Metrics*), que prevê a criação de métricas a partir de objetivos da pesquisa e suas respectivas perguntas, propostas para atingir os objetivos (BASILI; CALDIERA; ROMBACH, 1994).

Seguindo o método GQM, deve-se primeiramente definir os objetivos da medição. Para isso dois conjuntos de perguntas correlacionadas foram preparadas. O primeiro e principal conjunto teve como objetivo verificar se a premissa dessa tese fora satisfeita com o modelo desenvolvido. O segundo avaliou questões de viabilidade geral do modelo. Finalmente, apresentaram-se os dados compilados das avaliações subjetivas do painel de especialistas bem como a compilação das perguntas de forma aberta.

Na primeira etapa primeiramente se traçaram 2 objetivos:

> **Objetivo 1:** Avaliar se o método tem potencial de melhorar o processo de estimação de SOA.
>
> **Objetivo 2**: Avaliar a viabilidade do método pelo ponto de vista dos especialistas.

Nesta etapa foram definidas as questões e as métricas para cada objetivo. As tabelas 65 e 66 apresentam a descrição do resultado dessa etapa. As 3 primeiras perguntas do questionário foram a respeito do conhecimento da pessoa que estava respondendo. Todas as questões mencionadas são do questionário e estão descritas no **Apêndice B.**

Tabela 65: Questões/métricas para o objetivo 1

Objetivo 1	
Questão 4	Você considera que a sistematização provida pelo método no processo de estimação contribui para uma padronização deste na empresa e assim pode se tornar uma prática dentro de um processo maior de governança de SOA & TI?
Métrica	Impressão objetiva do especialista sobre a padronização dos métodos tradicionais da estimação.
Questão 5	Você considera que o método desenvolvido estima com o devido rigor técnico / corretude um projeto de software baseado em serviços em uma perspectiva SOA?
Métrica	Impressão objetiva do especialista sobre o rigor técnico do método, oferecendo confiabilidade nos resultados.
Questão 6	Você considera que o método desenvolvido estima com a devida abrangência os vários aspectos de um projeto de software baseado em serviços em uma perspectiva SOA?
Métrica	Impressão objetiva do especialista sobre a importância da flexibilidade do método.
Questão 7	Você considera que o método desenvolvido diminui a subjetividade na estimação e assim dá maior confiança na decisão?
Métrica	Impressão objetiva do especialista sobre redução das subjetividades, oferecendo confiabilidade nas decisões.

Tabela 66: Questão/métrica para o objetivo 2

Objetivo 2	
Questão 13	Quais são as principais dificuldades que você vislumbra na implementação do método proposto na sua empresa ou nas empresas de software em geral?
Métrica	Impressão subjetiva do especialista sobre a aplicabilidade do método em diferentes tamanhos e tipos de empresa.

Aplicação dos Questionários

O questionário final teve sua aplicação iniciada em maio de 2017 e finalizada em julho de 2017.

Resultados compilados

Após cada empresa ter aplicado o método usando seus casos de uso o questionário foi aplicado a cada uma delas. Os resultados são apresentados a seguir.

Respostas das perguntas sobre o avaliador, considerando de 1 (pouco) a 5 (muito), apresentadas na tabela a seguir.

Por questões de sigilo, o nome das empresas serão colocados apenas na forma da sua letra inicial: empresa ‘S’ (empresa de software de grande porte), empresa ‘C’ (empresa de software de médio de porte) e empresa ‘P’ (empresa de não software de grande porte).

Tabela 67: Respostas às questões 1 e 2

Pergunta	Respostas		
	S	P	C
1. Seu conhecimento sobre o processo geral de estimação de software	3	4	3
2. Seu conhecimento em SOA ou em sistemas / modelos de negócios baseados em serviços de software	5	5	3

Resposta da pergunta sobre a empresa, considerando as opções sim/não e observações em cada uma delas, apresentadas na Tabela 68.

Tabela 68: Respostas à questão 3

Pergunta	Respostas		
	S	P	C
3. A sua empresa utiliza algum método formal ou sistematizado de apoio aos gestores de TI na estimação do esforço, custo, tempo de desenvolvimento e/ou de alinhamento ao negócio de projetos SOA?	não	sim	não
*os comentários foram avaliados pelo autor			

Respostas às perguntas de múltipla escolha com as opções na Escala Likert: *"Concordo totalmente", "Concordo Parcialmente", "Indiferente", "Discordo Parcialmente"* e *"Discordo totalmente",* apresentadas na tabela 69.

Tabela 69: Respostas às questões de 4 a 9

Pergunta	Respostas		
	S	P	C
4. Você considera que a sistematização provida pelo método no processo de estimação contribui para uma padronização deste na empresa e assim pode se tornar uma prática dentro de um processo maior de governança de SOA & TI?	Concordo Totalmente	Concordo Totalmente	Concordo Parcialmente
5. Você considera que o método desenvolvido estima com o devido rigor técnico / corretude um projeto de software baseado em serviços em uma perspectiva SOA?	Concordo Totalmente	Concordo Parcialmente	Concordo Parcialmente
6. Você considera que o método desenvolvido estima com a devida abrangência os vários aspectos de um projeto de software baseado em serviços em uma perspectiva SOA?	Concordo Totalmente	Concordo Parcialmente	Concordo Totalmente
7. Você considera que o método desenvolvido diminui a subjetividade na estimação e assim dá maior confiança na decisão?	Concordo Totalmente	Concordo Totalmente	Concordo Parcialmente
8. Você considera que o método desenvolvido ajuda no refinamento e/ou definição da granularidade dos serviços e decisões de reuso ?	Concordo Totalmente	Concordo Parcialmente	Indiferente
9. Você acha que faz sentido poder dar pesos de importância diferentes aos vários fatores do método para cada projeto SOA em específico?	Concordo Totalmente	Concordo Totalmente	Concordo Totalmente
*os comentários foram avaliados pelo autor			

Respostas às perguntas que avaliam diversos fatores do método. Dentro de cada opção, foi colocado o nome da empresa que escolheu aquele item. São duas perguntas desse tipo, a 10 e a 12:

Pergunta:
10. Avaliando o método proposto e considerando de forma genérica um dado projeto SOA, como você classificaria o grau de importância dos fatores cobertos pelo método no processo de estimação?

Tabela 70: Resposta à questão 10

Fator / Avaliação	Essencial	Importante	Pouco relevante	Não necessário	Incorreto
Requisitos Funcionais	S, P	C			
Complexidade Integração	S, P		C		
Experiência Profissional		S, P, C			
Condição Ambiental		S, P		C	
Complexidade Ambiental		S, P		C	
Requisitos Não Funcionais	P	S	C		
Complexidade Funcional	P	C, P			
Tamanho	S	C, P			
Reuso	S, P		C		
Esforço	S, P	C			
Custo	S	C, P			
Tempo de desenvolvimento	S	C	P		
Alinhamento Negócio	S	C, P			

12. Você acha que o método tem o potencial de proporcionar as seguintes melhorias?

Tabela 71: Resposta da questão 12

Melhoria / Avaliação	Concordo totalmente	Concordo Parcialmente	Não sei	Discordo Parcialmente	Discordo totalmente
Maior rapidez no processo de estimação	S, P	C			
Melhor base para gestão de projetos	S	P	C		
Ajuda como referencial de custos e tempos numa terceirização ou subcontratação	S, P	C			
Se tornar uma ferramenta de apoio a treinamento em estimação	S, P	C			

A seguir serão apresentadas as respostas para as três perguntas discursivas (perguntas 11, 13 e 14), com as suas respostas compiladas e sumarizadas pelo autor.

Tabela 72: Respostas das questões 11, 13 e 14

11. Você acha que haveria outros fatores necessários de serem mensurados para estimação de projetos SOA? Se sim, quais?

S	Impacto da não entrega de um serviço
P	Regulamentação e Legislações específicas
C	Não respondeu
13. Quais são as principais dificuldades (de qualquer tipo) que você vislumbra na implementação do método proposto na sua empresa ou nas empresas de software em geral?	
S	Capacitação
P	Profissional capacitado para usar o método
C	Convencer as pessoas da utilização
14. Se desejar fique livre para comentar sobre o método de estimação proposto.	
S	Excelente trabalho. O Método traria confiança na justificativa de priorização dos projetos
P	Excelente trabalho
C	Método faz sentido para fábricas de Software

7.5 DISCUSSÃO SOBRE OS RESULTADOS

Esta seção retrata a terceira fase da técnica *Expert Panel*, de análise dos resultados. Dado ao número muito pequeno de empresas participantes não foi feita uma análise estatística, mas qualitativa.

A Tabela 73 relaciona a proposição do valor apontado no tópico 1.5 e a questão que tratar ele.

Tabela 73: Proposições do valor x questões do questionário

	Proposição de valor	**Questão**
1	Sistematização do processo	4
2	Menor subjetividade, potencializando tomada da decisão	7
3	Decisão mais rápidas	12
4	Ponderar importâncias nos fatores de decisão	9
5	Comparar modo desenvolvimento tradicional com SOA	12
6	Permite priorização das demandas	9
7	Avalia desenvolvimento composto e isolado	12
8	Considera as vantagens e o custo do reuso	8
9	Avalia o impacto da adição de um ou mais colaboradores na equipe	12
10	Oferece base para decisão da possibilidade de subcontratações ou terceirizações de desenvolvimento de software.	12

Constatou-se que as respostas foram positivas em relação a praticamente todos os itens. Observa-se que basicamente até a questão 9 todas as respostas foram positivas, entre *concordo* e *concordo*

totalmente. Esse resultado corrobora com a efetividade do objetivo geral desta tese, que foi o de conceber um método de estimação para projetos SOA de maneira a permitir que ele atue como um guia para melhorar a qualidade geral do processo de análise e decisão.

Primeiramente, analisando as questões 4, 5, 6 e 7, que respondem ao objetivo 1 definido no GQM. Para as questões 4, 6 e 7 que tratam respectivamente da padronização, da abrangência na sua aplicação e da redução da subjetividade do processo de estimação, a maioria dos especialistas concordaram totalmente que o modelo tem potencial de padronização e redução da subjetividade no processo da estimação dentro da sua organização, além de atender a vários aspectos de um projeto de software. Outros especialistas concordaram apenas parcialmente com essa afirmação, o que estimula a evolução do método em um futuro próximo. Estes resultados demonstram que o método teve boa aceitação para ser implementado na empresa.

A questão 5 avalia o rigor técnico do método em relação a confiabilidade do resultado final. Neste caso somente um especialista considerou "Concordo Totalmente" e as empresas P e C responderam "Concordo Parcialmente". Isto por que consideraram que o rigor técnico e o resultado final do método irão depender da experiência de quem preencher e usar o método. Portanto, estas empresas consideram que é necessário se ter um usuário com grande conhecimento técnico.

A questão 8 avaliou a contribuição do método na definição da granularidade dos serviços e decisões de reuso das funcionalidades. No caso da empresa P, que respondeu "Concordo Parcialmente", apontou que apesar dos resultados do método serem importantes, seriam necessário outros artefatos, como modelos de dados e de processos na definição de reuso. No caso da empresa C, em alguns itens achou indiferente. Essa resposta parece ser muito vinculada ao tipo de empresa que é e à metodologia de trabalho utilizada. A empresa C é uma empresa que nasceu de uma *startup* e que tem seus métodos de trabalho baseados somente em metodologias ágeis e características do *kanban*, na qual a estimativa não é um ponto exaustivamente explorado. Nesse sentido, ela se mostrou pouco atraída com o método por considerá-lo um pouco burocrático.

A questão 9, que trata da importância de atribuir peso para criar flexibilidade de uso do método em projetos diferentes, todos os avaliadores tiverem resposta "concordo totalmente", representando a importância de existir um método adaptável para cada tipo de projeto de software.

Já as perguntas que abrangeram os itens/fatores que compõe o método em si (perguntas 10 e 11), foi possível notar que as empresas S e P foram mais positivas na aceitação dos fatores. Esse resultado corrobora a pergunta de pesquisa desta tese, na qual desejou-se saber quais são os fatores a serem considerados num processo de estimação de um projeto SOA e como tais fatores podem ser organizados de forma a sistematizar o processo de análise e decisão sobre implementar ou não um software na forma de serviços em uma perspectiva SOA. No caso da empresa C, talvez devido ao seu perfil, esta identificou alguns fatores como pouco relevantes ou mesmo desnecessários.

A questão 12, que avalia outras contribuições e proposição de valor, teve resultado similar às questões 10 e 11, onde as empresas P e S concordaram totalmente com as contribuições. Somente no item relacionado com a base para gestão de projeto a empresa P concordou parcialmente por acreditar que, além de dados do método, é importante ter outros artefatos para alcançar uma maior eficiência na gestão de projetos.

Sobre as perguntas finais (questões 13 e 14), abertas, todas as empresas se mostraram satisfeitas com o método. Porém, foi possível notar a preocupação em ter um especialista em aplicar o método. Na verdade, tal necessidade é muito comum na análise de outros problemas, como por exemplo em análises financeiras e de viabilidade, na qual também se exige profissionais altamente capacitados na área.

Na compilação dos resultados do questionário e das avaliações preliminares realizadas foi possível observar a grande aceitação do método e da importância de um método como este. O resultado final e o ranqueamento apresentados pelo método foram considerados importantes/úteis para os gestores das empresas assim como os resultados coerentes com a realidade delas. Os especialistas envolvidos destacaram o potencial de melhoria ao se sistematizar o processo de mediação e, com isso, melhorar a qualidade do processo geral de estimação.

8 CONCLUSÕES

O advento das arquiteturas orientadas a serviços (SOA) trouxe às empresas uma série de benefícios, mas igualmente uma série de outras dificuldades. Uma delas se refere a estimação de software e, mais precisamente, de *serviços* de software.

Projetos SOA vão muito além da questão somente tecnológica, impactando as empresas em várias dimensões. Restrito de certa forma à etapa de *Análise* de projetos SOA, esta tese apresentou uma contribuição àquele problema, representada na forma de um método de estimação. Essencialmente, este método visa dar às empresas uma melhor base para suas decisões sobre a implementação de certas funcionalidades baseadas em serviços de software sob uma perspectiva SOA, ou se na forma de software 'tradicional', monolítico. Em se tendo a qualidade da decisão melhorada, potencializa-se uma diminuição de risco dos projetos SOA, incluindo o risco de se desenvolver serviços não suficientemente alinhados aos negócios da empresa.

A área de estimação de software tem já artefatos conceituais muito consolidados na literatura e em muitas práticas de empresas. No caso da estimação de serviços em uma perspectiva SOA, observou-se que, dada a ausência de um método sistematizado de estimação de serviços, as empresas fazem tal estimação basicamente usando aqueles mesmos artefatos mas sem a devida consideração e adaptação dos inúmeros aspectos mais ligados às características do SOA (resumidamente mostrados na figura 41). Como consequência, a qualidade das estimações acaba por ser baixa, o que é inclusive apontado na literatura como uma das razões dos fracassos de muitos projetos SOA. Neste sentido, o método desenvolvido consiste fundamentalmente de uma "releitura" daqueles artefatos, adaptando-os a SOA assim como adicionando outros aspectos. Isto também foi reflexo de uma visão estratégica quando da concepção do método, visto que um método "radicalmente" diferente dos modelos e técnicas clássicas de estimação poderia dificultar sobremaneira a adoção do método desenvolvido pelas empresas, de alguma forma já acostumadas com aquelas clássicas.

Referente a esse ponto acima mencionado, não se trata, portanto, de então "forçar" as empresas a abandonarem os métodos/técnicas clássicas, mas sim de integrar e complementar tais métodos/técnicas para um contexto de características de SOA.

Além disto, o método transformou a análise de projetos SOA num processo, sistematizado, composto por 15 passos (fatores de estimação), rigorosamente identificados, fazendo com que os gestores de TI os "percorram", forneçam vários dados (99 dados ao total na versão atual) e, ao final, resultados concretos mensuráveis para tempo de desenvolvimento, custos, esforços e grau de alinhamento ao negócio sejam calculados. Tais cálculos são feitos com base em técnicas consolidadas e melhores práticas assim como com o apoio de um sistema computacional. Este sistema auxilia o gestor no uso do método e gera tabelas comparativas sobre o desenvolvimento de funcionalidades como serviço e como software tradicional, sugerindo prioridades de desenvolvimento consoante às prioridades da empresa ou as de uma demanda de negócio em particular. Salienta-se que o método indiretamente ajuda o gestor a não esquecer de nenhum aspecto numa análise "completa", considerando que são quase 100 os aspectos a terem que ser analisados ao total.

A um nível mais estratégico, o uso do método pode trazer outros benefícios, como o de facilitar a avaliação da terceirização de parte do desenvolvimento e de ajudar na definição de valores (venda ou aluguel) que a empresa possa cobrar ou ser cobrada de softwares oferecidos na modalidade SaaS. Tais benefícios podem ser alargados ao se considerar que as demandas de software a serem analisadas pelo método podem envolver demandas apenas ainda vislumbradas. Por exemplo, na estimação de novas soluções de software ou na prototipações de iniciativas de inovação.

Considerando a proposição de valor do método, não se tratou de assumir que até então as empresas não estimavam ou que não teriam como estimar seus softwares sem fazer uso de um modelo mais formalizado de processos de estimação. A questão é que, com o método proposto, aumenta-se a qualidade técnica da estimação, a confiança nos seus resultados, se diminui a subjetividade e empirismo no processo, e se tem a possibilidade de automaticamente comparar os "custos gerais" de implementação como serviços e como software tradicional, incluindo o de fazer simulações para fins de apoio à gestão de projetos.

Considera-se que a introdução de um método abrangente como este deva ser igualmente encarada como um "processo". Isto porque o seu pleno uso parte do princípio que as empresas já conheçam uma série de conceitos e informações relativas à estimação, o que significa já terem uma razoável cultura SOA. Porém, a prática tem mostrado que isso não é uma realidade muito forte e que a adoção do SOA leva tempo. Isto se torna ainda evidente ao se considerar que mais de 90% das

empresas de TI do Brasil são micro e pequenas empresas de software. Portanto, é visto como natural não apenas o método evoluir ao longo do tempo e talvez ter que ser adaptado à maturidade SOA de cada empresa usuária, mas também que ele possa se transformar numa prática dentro de um processo maior de governança de TI.

Por outro lado, apesar dessas usuais limitações das empresas, um método como este pode igualmente ajudá-las a adotar *algum* processo de estimação, mesmo que a empresa não tenha todos os dados requeridos para o uso pleno do método. Isto porque, e como acima escrito, a grande maioria das empresas de software são de pequeno porte e geralmente não dispõem de gestores de TI especializados em estimação e lidam com informações pouco confiáveis de medição e de históricos. Também como citado, a maioria dos gestores fazem estimação de forma empírica, basicamente contando com as suas experiências e históricos, com grandes margens de erro, o que mitiga o uso de padronizações, métricas, comparativos, uma mais acurada estimação e, por fim, a competitividade das empresas.

Como a questão de formação de recursos humanos é sempre crítica em TI, um método como este pode ser usado como uma interessante ferramenta de treinamento em estimação para novos gestores de TI.

Teoricamente o método tem o potencial de agilizar o processo de estimação em termos de torná-lo mais rápido. Isso tem no entanto um certo relativismo. Gestores atuais, com empirismo e visão técnica apenas parcial sobre o que se deve considerar tecnicamente numa análise "completa" SOA, podem fazer uma estimação em menos tempo que o método. Já se ele tivesse que considerar todos os aspectos e os calcular manualmente, então muito possivelmente demoraria mais tempo sem o método. Durante a validação houveram funcionalidades que demoraram apenas 15 minutos para serem estimadas; outras, até 45 minutos. Esta variação não é apenas dependente da experiência do gestor de TI, mas das características de cada caso, da necessidade de interação entre pessoas, e da disponibilidade de dados confiáveis para o preenchimento das fórmulas dos fatores. Segundo os gestores pesquisados, é comum se gastar muito mais do que 1 hora para se estimar uma funcionalidade, e mesmo assim sem calcular com o devido rigor e amplitude.

Considera-se que a lista de fatores do método ajuda os gestores não apenas no processo da estimação em si, mas também na sua gestão. Isto porque cada fator tem suas características particulares, atores e

papéis, o que propicia ao gestor antever ações, seus custos e tempos, e nível de complexidade e risco.

A rigor, o método pode abarcar empresas de todo os portes, de *startups* a grandes empresas de software. Com base nos resultados da pesquisa julga-se que o fundamental para o sucesso dos resultados do método é a cultura existente de SOA e de estimação, e não necessariamente o porte da empresa. Por outro lado, fica uma questão relativamente em aberto para futuras averiguações o investigar em que medida este método interferiria na natureza enxuta e mais fluída das metodologias ágeis pelas empresas que as adotam.

Este ponto das metodologias ágeis bem como outras adaptações pode na verdade ser de certa forma resolvido pela flexibilidade do método proposto. Apesar dele ter sido concebido para ser o mais abrangente possível no escopo de projetos SOA, cada empresa pode efetuar adaptações no método. Isto pode ser tanto acrescentando ainda mais fatores ou variáveis às suas fórmulas, como retirando fatores ou variáveis de forma a simplificar o método ou o cálculo de alguns fatores consoante à realidade das empresas. Na verdade, esta última situação já é permitida, bastando o gestor de TI atribuir zero a variáveis (algumas ou todas) dos fatores considerados "desnecessários".

A flexibilidade do método se estende também à questão tecnológica. A análise dos aspectos de implementação dos serviços considera outras tecnologias que não apenas o padrão *web services* (e suas associadas WSDL e SOAP). No fator complexidade ambiental são consideradas outras tecnologias emergentes, como microserviços, REST, uso de middlewares, etc. Este fator pode – e deve – ser naturalmente atualizado à medida que novas tecnologias surjam.

Sobre o ineditismo do método, verificou-se a inexistência na literatura de um similar ao proposto, que envolva os considerados principais fatores de um processo de desenvolvimento SOA, de forma alargada, como um processo sistematizado integrado, cobrindo não apenas os inúmeros aspectos tecnológicos envolvidos em cada fator mas também os organizacionais, culturais e de governança de uma empresa.

Com base nos resultados da avaliação efetuada, pôde-se concluir que, dentro do ambiente controlado utilizado, limitações existentes e procedimentos metodológicos adotados, o objetivo geral da tese foi atingido. Não apenas em termos da concepção do método de estimação em si, mas também quanto a sua proposição de valor.

Em termos dos objetivos específicos, considera-se que eles foram atingidos na medida em que: i) foram determinados quais fatores devem ser analisados durante processo da estimação numa ótica SOA e suas

inter-relações; ii) foram definidos e especificados os elementos de análise a serem computados em cada fator; iii) foi identificado como o processo geral de desenvolvimento SOA difere do de desenvolvimentos não-SOA e como isto deve ser considerado num método; iv) foram definidas as fórmulas matemáticas de cálculo de cada fator envolvido no processo da estimação; v) foi adotado um método multicritério para ranqueamento das funcionalidades; vi) foram sistematizadas as atividades do processo de estimação da forma de um método; e vii) foi implementado um protótipo de software para auxiliar o gestor de TI no uso do método.

Do ponto de vista de contribuição científica, esta tese oferece alguns avanços em relação ao estado da arte.

O primeiro se refere à engenharia de software e SOA. Há toda uma linha de pesquisa de alinhamento de metodologias clássicas de engenharia de software para SOA e também para desenvolvimento "tradicional", onde o desenvolvimento de software não é comparado utilizando um método único. Nesta tese, esses modelos foram expandidos, adaptando-os a um cenário onde o gestor pode estimar um desenvolvimento baseado em modelos diferentes.

O segundo se refere ao levantamento de todos os fatores considerados relevantes de serem avaliados num processo de desenvolvimento SOA, incluindo a adaptação de métodos de estimativas para SOA e o uso de técnicas multicritério para melhor ranquear os serviços candidatos.

8.1 LIMITAÇÕES

Apesar do potencial do método desenvolvido, este não deve ser visto como uma solução final para todo e qualquer projeto de SOA. O método representa apenas uma parte de um *framework* maior que as empresas podem utilizar para tomada de decisão ao se considerar outras dimensões de análise também desejáveis para projetos SOA.

Embora considere parcialmente alguns aspectos de outras dimensões, o método foca essencialmente na dimensão TI, para a qual os principais referencias teóricos na área de estimação são voltados.

Um projeto SOA envolve várias etapas no seu ciclo de vida. O método desenvolvido dá suporte apenas à etapa de *Análise*. Dentro desta etapa, o método assume que uma empresa adota certos processos que, em suma, compreendem as atividades de *serviços candidatos, análise de gap* e *realização*. O método atua basicamente após as duas primeiras

atividades, a de *realização*, para fazer a estimação. Dentro da atividade de realização, o método não automatiza a decisão em si, mas deixa completamente ao cargo do gestor de TI a decisão final; ou seja, o método *sugere*, não *determina*.

O método analisa apenas o horizonte da ação de desenvolvimento de software. Ele não dá estimativa ou não faz qualquer tipo de análise sobre os custos gerais desta decisão em termos de futura manutenção e mantenimento da solução de software desenvolvida e nem de análises sobre modelos de disponibilização do software (por exemplo, se uma dada solução fosse total ou parcialmente ofertada na forma SaaS).

No caso da análise de desenvolvimento composto/agregado, o software implementado não gera e não calcula automaticamente todas as combinações possíveis de composição, também deixando a cargo do gestor de TI a escolha de quais composições poderiam ser mais pertinentes de serem analisadas.

O método desenvolvido não visa propor melhorias em modelos de governança SOA, mas apenas atuar na melhoria de processos de governança relativos à análise e seleção de serviços. Portanto, apenas se assume que as empresas têm algum modelo de governança de TI e SOA.

Do ponto de vista de riscos, não é feita nenhuma análise propriamente dita de riscos e/ou de sua gestão. O método dá suporte à estimação tendo-se como premissa que o gestor de TI terá uma melhor base de avaliação e com isto pode-se diminuir o risco da decisão.

Apesar dos fatores do método, suas inter-relações, suas fórmulas, etc., foram cuidadosamente concebidos após rigorosa revisão da literatura, melhores práticas, avaliações de grupos de usuários e avalições de revisores dos artigos publicados sobre o método por especialistas na área, não se pode dizer que *todos* os fatores efetivamente necessários estão contemplados no método. O método é flexível para ser alterado e evoluir, mas de qualquer forma seria necessária uma pesquisa longitudinal junto a inúmeras empresas para "garantir" que "todos" os fatores estariam presentes. Além disto, só um estudo longitudinal de mais longo prazo assim como uma maciça aplicação deste método e comparativos com as práticas atuais das empresas e com as técnicas/métodos clássicos poderá "garantir" a efetiva confiabilidade do método desenvolvido.

Um ponto importante observado e levantado por empresas é que o método acaba por exigir do gestor de TI um grande conhecimento em SOA e das práticas da sua empresa para que as potencialidades do método possam ser melhor aproveitadas.

8.2 TRABALHOS FUTUROS

As limitações da versão atual deste método desenvolvido, e mesmo suas contribuições e problemas solucionados dentro de um problema geral bastante amplo, e sugestões da banca, abrem caminho para várias melhorias gerais, para novas pesquisas e desenvolvimentos, a seguir listadas na forma de itens para melhor identificar cada ação:

- Aplicar o método em mais empresas visando obter uma avaliação mais alargada e ao longo de talvez alguns anos e com isso detectar necessidades de melhorias.

- Formalizar o método desenvolvido, mais claramente expressando as atividades envolvidas, o papel de cada tipo de ator ao longo do processo, e as entradas e saídas de cada atividade. E mesmo formalizar ou especificar o processo de uso do método em si.

- Ter a possibilidade de guardar históricos das estimações, para utilizar posteriormente em análises comparativas e melhor calibragem de valores e pesos dos fatores a medida que a empresa ganhe maior experiência, tanto em SOA em si, como no uso do método e dos vários aspectos por ele cobertos.

- Analisar a possibilidade ou necessidade de se criar uma versão "enxuta" do método, tanto na perspectiva de ser usado por empresas pouco maduras em SOA, como por empresas cujos desenvolvimentos são fortemente apoiados por metodologias ágeis. Isto porque, possivelmente, algumas empresas não tenham como usufruir em plenitude da abrangência do método pois só têm informações confiáveis apenas sobre alguns dos fatores cobertos ou de alguns elementos dentro de certos fatores. Portanto, isto leva a uma reflexão de pesquisa, a ser mais aprofundada. Neste sentido pode ser utilizado como base da pesquisa as "*instruções de adaptação*" do CMMi e MPS.BR.

- Avaliar a possibilidade e viabilidade de transformar o método numa ferramenta SaaS, acessível por quaisquer empresas do mundo pela Internet e eventualmente pago por uso/acesso (ou seja, por ação de estimação).

- Estender o método na forma de se criar um *framework* onde, além do método, se tenha um "anterior", de análise da viabilidade, riscos e retornos possíveis em se adotar SOA. Como mencionado ao longo da tese, a adoção de SOA traz impactos nas empresas e

exige dela uma série de preparações. Esta tese assumiu que as empresas já são usuárias SOA, ou seja, já tomaram em algum momento a decisão estratégica de adotar esse paradigma. Porém, esta adoção não pode ser feita apenas pelo fato de SOA ser "uma grande tendência" e por estar vindo a ser adotado por um número crescente de empresas, mas sim após uma cuidadosa análise estratégica.

Algumas melhorias foram sugeridas pelas empresas e parecem relevantes de serem introduzidas em futuras versões:

- Expandir o escopo do método para outras estapas do ciclo de vida SOA, como a estimação / impacto do desenvolvimento como serviços nas futuras manutenções e mantenimento (serviços e infraestrutura de suporte), ou seja, depois de já se ter implementado os serviços e estes se tornarem ativos de software.
- Apesar de ser um método cuja entrada não tenha que ser via Use Cases, poder-se-ia integrar a ferramenta desenvolvida com ferramentas UML de forma a "automaticamente" acessar os Use Cases para início da estimação.
- Aprofundar os aspectos de regulamentações e conformidade no método como forma de se expressar ou melhor contemplar legislações de interesse e aspectos do modelo de governança de TI já adotado pela empresa quando de uma estimação.
- Estender o alcance do método para considerar ativos de software (serviços) desenvolvidos e disponibilizados por outras empresas para poderem ser reusados numa dada solução. Dentro da linha de pesquisa em Redes Colaborativas de Organizações alguns trabalhos vêm expandindo o conceito de compartilhamento de recursos ao nível de ativos de software. No contexto de serviços de software tem sido usado o conceito de "federação de serviços", um ambiente que virtualiza todos os serviços que empresas se dispõem a compartilhar (sob variados modelos de acesso, pagamento e governança) e que funciona como que uma "nuvem" sobre os diversos provedores. Portanto, uma estimação nesse ambiente poderia contemplar (ou mesmo simular composições com) tais serviços, maximizando o reuso e minimizando os custos e tempo de desenvolvimento.

9 REFERÊNCIAS BIBLIOGRÁFICAS

ABES. 2016. http://central.abessoftware.com.br/Content/UploadedFiles/Arquivos/Dados%202011/ABES-Publicacao-Mercado-2016.pdf. Acesso em: 4 de novembro de 2014.

ABNT - Associação Brasileira de Normas Técnicas. "NBR ISO 9000:2000 – Sistemas de gestão da qualidade e garantia da qualidade – Fundamentos e Vocabulário". Rio de Janeiro: ABNT, 2001, 29p.

ADAM, S.; RIEGEL, N.; DOERR, J. **Deriving Software Services from Business Processes of Representative Customer Organizations**. In SOCCER '08. International Workshop on Service-Oriented Computing: Consequences for Engineering Requirements, p. 38-45, 2008.

ALBRECHT, A., **Measuring Application Development Productivity, Programming Productivity: Issues for the Eighties**, IEEE Computer Society Press, Washington DC, 1981.

ALBRECHT, A.L., GAFFNEY, J.E. **Software Function, Lines of Code, and Development Effort Prediction: A Software Science Validation**, 1983.Artigo do BFPUG (Brazilian Function Point Users Group): http://fase.bfpug.com.br/Artigos/Albrecht/Albrecht_Gaffney.pdf . Acesso em: 10 de maio 2015.

ALLEN, J. FASE., BARNUM S., ELLISON R.J., MCGRAW G., MEAD N. R., **Software Security Engineering: A Guide for Project Managers**. Addison Wesley Professional, 2008. 368p.

ALMEIDA, A. T. de, **O Conhecimento e o Uso de Métodos Multicritério de Apoio a Decisão**, 2ª. Edição, Editora Universitária, Recife, 2011.

ALMEIDA, A.T. de; MORAIS, D. C.; COSTA, A.P.C.S; ALENCAR, L. FASE; DAHER, S.F.D, **Decisão em Grupo e Negociação: Métodos e Aplicações**, Editora Atlas, 2012.

ALMEIDA, A.T. de, **Processo de Decisão nas Organizações: Construindo Modelos de Decisão Multicritério**, 1a Edição, Editora Atlas, São Paulo, 2013.

ALVES, S. N. L.; SIMÕES, S. A. C.; NEYRA, B. M. C. **Revisão de literatura sobre a aplicação do método ANP ao problema de seleção de fornecedores**. Anais do XL SBPO, Setembro 2-5, João Pessoa, Brasil. 2008.

ARROW, K. J., **Social Choice and Individual Values**, 2d ed. Cowell Commission Monograph no. 12, Wiley, New York, 1963

ARSANJANI, A.; HOLLEY, K. **The Service Integration Maturity Model: Achieving Flexibility in the Transformation to SOA**. Services Computing, 2006. SCC '06. IEEE International Conference on, 2006. p.515-515.

ARSANJANI, A.; GHOSH, S.; ALLAM, A. *et al*. **SOMA: A method for developing service-oriented solutions.** IBM Systems Journal, v. 47, n. 3, 2008, p. 377-396.

AZEVEDO, L., BAIAO, F., SANTORO, F., SOUZA, J., REVOREDO, K., PEREIRA, V., HERLAIN, I. **Identificação de serviços a partir da modelagem de processos de negócio.** In: Simpósio Brasileiro de Sistemas de Informação , Brasília, 2009.

AZEVEDO, L. G. ; BAIAO, F. ; SANTORO, F. M. ; SOUZA, J. F. . **A Business Aware Service Identification and Analysis Approach.** In: Proceedings of the IADIS Information Systems 2011. Espanha: IADIS - International Association for Development of the Information Society, 2011. p. 196-203.

AZEVEDO, L.G.; SANTORO, F. M.; BAIÃO, F. ; DIIRR, T. ; SOUZA, A. ; SOUZA, J. F.; SOUSA, FASE. P. **A Method for Bridging the Gap between Business Process Models and Services**. iSys: Revista Brasileira de Sistemas de Informação, v. 6, p. 62-98, 2013.

BANO, M. e IKRAM, N.,. **Issues and challenges of Requirement Engineering in Service Oriented Software Development**, 5th Int. Conf. on Software Engineering Advances, 2010, p. 64-69.

BASILI, V. R.; CALDIERA, G.; ROMBACH, FASE. D. **The Goal Question Metric Approach**. In: WILEY (Ed.). Encyclopedia of Software Engineering, v.1, 1994.

BERTO, R. M.; NAKANO, D. N. **Metodologia da Pesquisa e a Engenharia de Produção,** Anais ENEGEP, 1998.

BERNSTEIN, P. A.; HAAS, L. M. **Information integration in the enterprise**. Commun. ACM, v. 51, n. 9, 2008, p. 72–79.

BFPUG; **Brazilian Function Point Users Group (2008), Número de CFPS por País**. Disponivel em: www.bfpug.com.br, Acesso em: 29 Janeiro 2014.

BENNETT, Stephen G., **Determining ROI of SOA through Reuse**, Oracle® Practitioner Guide, 2012.

BENTE A., HEGE D., MAGNE J., DAG S., **Estimating Software Development Effort based on Use Cases - Experience from Industry.** In M. Gogolla, C. Kobryn (Eds.): UML 2001 - The Unified Modeling

Language. Springer-Verlag. 4th International Conference, Toronto, Canada, LNCS 218, 2001.

BIANCO, P.; KOTERMANSKI, R.; MERSON, P. **Evaluating a service-oriented architecture**. TECHNICAL REPORT CMU/SEI-2007-TR-015 ESC-TR-2007-015, Software Engineering Institute 2007.

BIEBERSTEIN, N., BOSE S., FIAMMANTE M.,JONES K., SHAH R., **Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap**. Estados Unidos: IBM Press, 2005.

BOEHM, B.W., EGYED A.F., KWAN J., **Developing Multimedia Applications with the WinWin Spiral Model**, ESEC/FSE 97, Springer, 1997, p. 20–39.

BOEHM B., **Software Engineering Economics**, IEEE Transactions on Software Engineering SE, 1984.

BÖRNER R., GOEKEN M., **Service identification in SOA governance literature review and implications for a new method.** The 3rd IEEE international conference on digital ecosystems and technologies (DEST'09). USA, 2009, p. 588–593.

BROWN, W. A.; LAIRD, R. G.; CLIVE GEE, T. M. **SOA Governance: Achieving and Sustaining Business and IT Agility** Pearson, 2009, p. 416.

CANCIAN, M. H.; RABELO, R. J.; WANGENHEIM, C. G. V. **Uma proposta para elaboração de Contrato de Nível de Serviço para Software-as-a-Service (SaaS). 8th International Information and Telecommunication Technologies Symposium, 2009**. The 8th International Information and Telecommunication Technologies Symposium (I2TS'2009), Florianópolis. 2009.

CANCIAN, M. H.; HAUCK, J. C. R.; WANGENHEIM, C. G. V. *et al*. **Discovering Software Process and Product Quality Criteria in Software as a Service**. In: SPRINGER, The 11th International Conference on Product Focused Software Development and Process Improvement (PROFES 2010). Limerick, Ireland. June 21-23, 2010. p.95-103.

CERVO, A. L.; BERVIAN, P. A.; SILVA, R. D. **Metodologia científica**. São Paulo: Pearson editor, 2007.

CHAN, K. S. M. **Formal methods for web services: a taxonomic approach.** The 32nd ACM/IEEE International Conference on Software Engineering - Volume 2. Cape Town, South Africa: ACM: 2010, p. 357-360.

CHUNG L., LEITE J. C. S. P. **On Non-Functional Requirements in Software Engineering.** In: Borgida A., Chaudhri V., Giorgini, P., Yu E. (Org.). Conceptual Modeling: Foundations and Applications. 1 ed. Berlin: Springer-Verlag, 2009.

CIGANEK, A. P.; HAINES, M. N.; HASEMAN, W. **Challenges of Adopting Web Services: Experiences from the Financial Industry**. The 38th Hawaii International Conference on System Sciences, 2005, p. 1-10.

CLELAND J. H., SETTIMI R., ZOU X., **Automated classification of non-functional requirements,** Requirements Engineering Journal,v2(2), 2007.

COSTA, H. G., **Introdução ao método de análise hierárquica: análise multicritério no auxílio à decisão**. Niterói: H.G.C., 2002.

COSTA, S. F.; CAETANO, A.; SANTO, S. C. **The Role of Business Opportunity Prototypes at the Recognition and Decision Stages of the Entrepreneurial Process**. In: 5th European Conference on Innovation and Entrepreneurship, Scotland.2011, p.269-277.

COSMIC. COSMIC **Functional Size Measurement Method**. COSMIC.2007.

CMMI - **Software Engineering Institute (SEI) for Services (CMMI-SVC)** versão 1.3. Carnegie Mellon University / Software Engineering Institute 2010. Disponível em: http://www.sei.cmu.edu, Acesso em: 15 de maio de 2016.

CRAWFORD, C. H.; BATE, G. P.; CHERBAKOV, L.; HOLLEY, K.; Tsocanos, C. **Toward an on demand service-oriented architecture**. IBM Systems Journal, v. 44, n. 1, 2005, p. 81-107

CURBERA, F.; KHALAF, R.; MUKHI, N.; TAI, S.; WEERAWARANA, S. **The Next Step in Web Services.** Communications of the ACM, v. 46, n. 10, 2003, p. 29-34

DIAS, J., OLIVEIRA, J., MEIRA, S., **Estudo Empírico sobre Adoção de SOA: Um Mapeamento Sistemático da Literatura**. Simpósio Brasileiro de Qualidade de Software, Bahia, 2013.

DIETRICH, A. J.; KIRN, S.; SUGUMARAN, V. **A Service-Oriented Architecture for Mass Customization: A Shoe Industry Case Study.** IEEE Transactions on Engineering Management, v. 54, n. 1, 2007, p. 190-204,

DYER, R. F.; FORMAN, E. H. **Group decision support with the Analytic Hierarchy Process, Decision Support Systems**. v.8. 1999, p. 99-124.

EL-MAADDAWY, T. **Enhancing learning of engineering students through self-assessment**. In: 2017 IEEE Global Engineering Education Conference (EDUCON), 2017, p.86-91.

EMAM, K. E. **Benchmarking Kappa: Interrater Agreement in Software Process Assessments**. Empirical Software. Eng., v. 4, n. 2, 1999, p. 113-133.

ENGHOLM JÚNIOR, H. **Engenharia de Software na prática.** São Paulo: Novatec, 2010.

ERL, T. **SOA Princípios de Design de Serviço.** Tradução de Carlos Schafranski e Edson Furmankiewicz. 1. ed. São Paulo: Pearson Prentice Hall, 2009.

ERL, T., **Service-Oriented Architecture: concepts, technology, and Design**, Prentice Hall, Crawfordsville: Indiana, 2005.

FAGG P., GALEGAONKAR S., JAIN P., LESTERHUIS A., NATARAJAN K., O'NEILL M., RULE G., SANTILLO L., SYMONS C., VOGELEZANG F., UNGERER G., **Guideline for Sizing Service-Oriented Architecture Software, The Common Software Measurement International Consortium (COSMIC)**, 2010.

FAREGHZADEH, N. **Service Identification Approach to SOA Development.** World Academy of Science, Engineering and Technology, 2008.

FARRAG, E. A., MOAWAD, R., **Phased effort estimation of legacy systems migration to service oriented architecture**. International Journal of Computer and Information Technology. 2014.

FERGUSON, P., HUSTON, G., **Quality of Service on the Internet: Fact, Fiction, or Compromise?**, John Wiley & Sons, 1998.

FISHBURN, P. C. **Utility theory for decision making**. New York: Wiley, (Operations Research Society of America Publications in operations research), 1970.

FREITAS, A.L.P. **Uma abordagem Multicritério para a classificação de hotéis**. RAUSP. Revista de Administração da USP. v. 42, n. 3, 2007, p. 338-348.

FREITAS, A.L.P., MORAIS, A.S.C. & BRITO, M.M. **Emprego de Métodos de AMD na Avaliação e Ordenação de Estabelecimento de Hospedagem Via Internet**, 2008.

FUGITA, H. S.; HIRAMA, K. **SOA Modelagem, Análise e Design**. Elsevier, 2012.

FUJITA, H. **Big data-based clouds health-care and risk predictions based on ensemble classifiers and subjective projection**. In: 2016 IEEE 17th International Symposium on Computational Intelligence and Informatics (CINTI), 2016. p.11-12.

GARTNER. **New Realities of IT. Stamford,** U.S.A.2010. Disponível em: http://www.gartner.com/technology/home.jsp, Acesso em: 01 Abril 2015.

GARCIA, D. TOLEDO, B. **Semantics-enriched QoS Policies for Web Service Interac-tions.** WebMedia'06. Natal, Brasil. 2006.

GHODDOSI, N. ; RABELO, R. J., **A Method for Evaluating the Feasibility of SOA Projects**. Anais 12th IEEE International Conference on Service Systems and Service Management, Guangzhou, China, 2015. p. 1-8.

GHODDOSI, N. ; RABELO, R. J., **A systematized analysis process for SOA services estimation**. Anais 11th International Conference on Computer Engineering & Systems (ICCES), Cairo, Egito, 2016. p. 384-389.

GIL, A. C. **Como elaborar Projetos de Pesquisa**. São Paulo: Editora Atlas S.A., 2010.

GITMAN, LAWRENCE J., Principles of Managerial Finance, Pearson Education,2010

GLIEM, J. A.; GLIEM, R. R. **Calculating, Interpreting, and Reporting Cronbacha's Alpha Reliability Coefficient for Likert-Type Scales**. In: Midwest Research to Practice Conference in Adult, Continuing, and Community Education, The Ohio State University, Columbus, OH .2003.

GOMES Y., PEREIRA M. , **Functional Size, Effort and Cost of the SOA Projects with Function Points**, Service Technology Magazine, 2012.

GROSSMAN, R. L. **The Case for Cloud Computing**. IT Professional, v. 11, n. 2, 2009, p. 23-27.

GU, Q. e LAGO, P., **Exploring service-oriented system engineering challenges: a systematic literature review, Service Oriented Comp. and Appl**., v.3, 2009, pages: 171-188.

GUPTA D. **Service point estimation model for SOA based projects**. Service Technology Magazine 2013. Issue LXXVIII.

GUPTA, M.; FERNANDEZ, J. **How Globally Distributed Software Teams Can Improve Their Collaboration Effectiveness?** , Global

Software Engineering (ICGSE), 2011 6th IEEE International Conference on. 2011. p.185-189.

HAFEEZ, K.; ZHANG, Y.B.; MALAK, N. Determining Key Capabilities of a Firm Using Analytic Hierarchy Process, Int'l J. Production Economics, vol. 76, no. 1, pp. 39-51, 2002.

HONGQI, L.; ZHUANG, W. **Research on Distributed Architecture Based on SOA**. Communication Software and Networks, 2009. ICCSN '09. International Conference on, 2009. p.670-674.

HOWARD, R.; KERSCHBERG, L. **A Framework for Dynamic Semantic Web Services Management**. International Journal of Cooperative Information Systems, v. 13, n. 4, 2004, p. 441-485.

HUHNS, M. N.; SINGH, M. P. **Service-Oriented Computing: Key Concepts and Principles**. IEEE Internet Computing, v. 9, n. 1, 2005, p. 75-81.

HUGHES B. e COTERELL M. **Software Project Management**, 2nd Ed. McGraw-Hill, Boston, 1999.

IFPUG, **Function Point Counting Practices Manual**, CPM v4.3.1, 2010. Disponível em: http://www.ifpug.org Acesso em: 10 maio 2014.

ISO/IEC. **International Organization for Standardization and International Electrotechnical Commission**, ISO/IEC 15504-2: Performing an assessment Genebra. 2002.

ISO/IEC. **Software engineering COSMIC: a functional size measurement method**. ISO/IEC 19761:2011, p. 1-14.

_____. **International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC)**. Standart 9126-1: Software engineering - Product quality - Part 1: Quality model 2003b.

_____. **International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC)**. Standart 9126: Software engineering - Product quality. 2004.

ISO/IEC 15504-5, **International Organization for Standardization and International Electrotechnical Commission, ISO/IEC 15504-5**: Information Technology - Process Assessment. Genebra. 2008a.

_____. **International Organization for Standardization and International Electrotechnical Commission, ISO/IEC 12207: Systems and software engineering — Software life cycle processes**. New York, p.138. 2008b.

_____. **International Organization for Standardization and International Electrotechnical Commission**, ISO/IEC 15504-7: Assessment of Organizational Maturity. Genebra. 2008c.

JAMSHIDI, P.; SHARIFI, M.; MANSOUR, S. **To Establish Enterprise Service Model from Enterprise Business Model**. IEEE International Conference on Services Computing, 2008, p. 93-100.

JANIESCH C., MATZNER M., MULLER O., VOLLMER R., BECKER J.. **Slipstream: Architecture options for real-time process analytics.** In, Proceedings of the ACM Symposium on Applied Computing (SAC'11), TaiChung, Taiwan, 2011.

JÄRVINEN, P. **Action Research is Similar to Design Science**. Quality & Quantity, n. 41, Springer. 2007, p.37-54.

JÄRVINEN, P. **On Research Methods. Tampere**, Finlândia: Opinpajan Kirja 2004.

JENKINS, A. M. **Research Methodologies and MIS Research**. Research Methods in Information Systems, 1985. Amsterdam, Holland. Science Publishers B.V. 1985, p.103-117.

JOSUTTIS, N., M. **SOA na prática: A arte da modelagem de sistemas distribuídos**. n. 9788576081845, 2008.

JONES, C. **A Guide to Selecting Software Measures and Metrics**, CRC Press, 2017.

JONES, S. Enterprise SOA Adoption Strategies. Using SOA to Deliver IT to the Business. C4Media, 2006.

KUMAR, S.; DAKSHINAMOORTHY, V.; KRISHNAN, M. S. **Does SOA Improve the Supply Chain? An Empirical Analysis of the Impact of SOA Adoption on Electronic Supply Chain Performance.** Proceedings of the 40th Hawaii International Conference on System Sciences, 2007, p. 1-10

KAJKO M., M., LEWIS, G. A., SMITH D. B., **A Framework for Roles for Development, Evolution and Maintenance of SOA-Based Systems**. SDSOA '07 Proceedings of the International Workshop on Systems Development in SOA Environments ,2007, p. 7.

KAPUR, P. K.; SINGH, G.; SACHDEVA, N.; TICKOO, A. **Measuring software testing efficiency using two-way assessment technique. In: Reliability**, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions), 2014 3rd International Conference on, 2014. p.1-6.

KARATZAS, K.; DIOUDI, E.; MOUSSIOPOULOS, N. **Identification of major components for integrated urban air quality management and information systems via user requirements prioritization**. Environmental Modelling & Software. v. 18, 2003, p. 173-178.

KARNER, G., **Metrics for Objectory.** Diploma thesis, University of Linköping, Suécia. 1993.

KARLSSON, J. **Software Requirements Prioritizing.** In: ICRE '96, 1996. Proceedings of International Conference on Requirements Engineering, 1996, p. 110-116.

KARLSSON, J.; WOHLIN, C.; REGNELL, B. **An evaluation of methods for prioritizing software requirements. Information and Software Technology**, v. 39, 1998, p. 939-947.

KEENEY, R. L.; RAIFFA, H. **Decisions with Multiple Objectives: preferences and value tradeoffs**. New York: John Willey & Sons, 1976. 569 p.

KIRIKOVA, M., GRUNDSPENKIS, J.; WOJTKOWSKI, W.; WRYCZA, S.; ZUPANCIC, J. **Information Systems Development: Advances in Methodologies, Components, and Management**, Springer Science & Business Media, 2012.

KITCHENHAM, B. **Guidelines for performing Systematic Literature Reviews**, in Software Engineering, 2007, p.87.

KITCHENHAM, B.; PFLEEGER, S. L. Principles of survey research part 6: data analysis. **SIGSOFT Softw. Eng. Notes,** v. 28, n. 2,2003, p. 24-27.

KITCHENHAM, B. A.; PFLEEGER, S. L. Principles of survey research part 2: designing a survey. **SIGSOFT Softw. Eng. Notes,** v. 27, n. 1, 2002, p. 18-20.

KOKKO Timo, ANTIKAINEN Jari, SYSTA Tarja. **Adopting SOA – Experiences from nine Finnish organizations**, IEEE -2009 European Conference on Software Maintenance and Reengineering, 2009

KONSTA, K.; PLOMARITOU, E. **Key Performance Indicators (KPIs) and Shipping Companies Performance Evaluation: The Case of Greek Tanker Shipping Companies**. International Journal of Business and Management v. 7, n. 10, 2012.

KHALUF, L., GERTH, C. and ENGELS, G., **Pattern-Based Modeling and Formalizing of Business Process Quality Constraints**, Int. Conf. on Advanced Information Systems Engineering (CAiSE), 2011.

KUPPURAJU, S.; KUMAR, A. **Enabling SOA Governance for Production Deployed Services**. In: Services - Part I, 2008. IEEE Congress on, 2008. p.109-110.

KOHLBORN, T.; KORTHAUS, A.; CHAN, T.; ROSEMMAN, M. **Identification and Analysis of Business and Software Services – A Consolidated Approach**, IEEE Transactions on Service Computing, Vol 2, No 1, 2009.

KRAFZIG, D.; BANKE, K.; SLAMA, D. **Enterprise SOA: Service-Oriented Architecture: Best Pratices**. Estados Unidos: Prentice Hall, 2004. 408p.

KRAMER, W. J.; JENKINS, B.; KATZ, R. S. **The role of the information and communications technology sector in expanding economic opportunity**. Economic Opportunity Series, Harvard, 2007.

KRISHNAN, S.; CLEMENTI, L.; JINGYUAN, R., **Design and Evaluation of Opal2: A Toolkit for Scientific Software as a Service**. Services - I, 2009 World Conference on, 2009. p.709-716.

KROGDAHL, P.; LUEF, G.; STEINDL, C. **Service-oriented agility: Methods for successful Service-Oriented Architecture (SOA) development.** 2010. http://www.ibm.com/developerworks/webservices/library/ws-agile1/ Acesso em: 05/10/2010.

KROLL, P., KRUCHTEN, P. **The Rational Unified Process Made Easy: A Practitioner's Guide to the Rup**. Addison-Wesley, 2003, 416p.

KULKARNI, Naveen; DWIVEDI, Vishal. **The Role of Service Granularity in A Successful SOA Realization – A Case Study**, IEEE Congress on Services, 2008.

LAIRD, L. e BRENNAN, C., **Software Measurement and Estimation - A Practical Aproach** (1a ed.). New York: Wiley and Sons, 2006.

LAMSWEERDE, A. Van, . **Requirements Engineering in the Year 00: A Research Perspective**, In Proc. of 22nd Int. Conf. on Software Engineering, 2000, p. 5-19.

LANE, S.; RICHARDSON, I. **Process models for service-based applications: A systematic literature review. Information and Software Technology**, v 53 (5), p. 424-439, 2011.

LANE, S.; BUCCHIARONE, A.; RICHARDSON, I. SOAdapt: **A process reference model for developing adaptable service-based applications**. Information and Software Technology, v 54 (3), 2012, p. 299-316.

LEITE, J.; YU, Y.; LIU, L. *et al*. **Quality-Based Software Reuse**. In: ENGINEERING, A. I. S. (Ed.): Springer Berlin / Heidelberg, v.3520, 2005. p.535-550.

LEE, K.; JOSHI, K.; BAE, M. **Using analytical hierarchy Process (AHP) to identify the relative importance of the features needed for web-based systems development**. Information Resources Management Journal, v. 21, n. 3, 2008, p. 88-100.

LEUSSE, P. DE; DIMITRAKOS, T.; BROSSARD, D. **A Governance Model for SOA**. In: Web Services, 2009. ICWS 2009. IEEE International Conference on, 2009. p.1020-1027.

LEWIS G. A., SMITH D. B., KONTOGIANNIS K., **A Research Agenda for Service-Oriented Architecture (SOA): Maintenance and Evolution of Service-Oriented Systems**, Software Engineering Institute, 2010.

LEWIS G. A., MORRIS E., SMITH D. B., **Analyzing the Reuse Potential of Migrating Legacy Components to a Service-Oriented Architecture**, Software Maintenance and Reengineering (CSMR'06), 2006.

LI, S.; LUO, Y. **Offshore Outsourcing Bridge: Designing and Development A Model Applied Management Platform for Chinese Offshore Outsourcing. Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing**, 2006. SNPD 2006. Seventh ACIS International Conference on, 2006. p.177-183.

LI, Z., KEUNG, J., **Software cost estimation framework for service-oriented architecture systems using divide-and-conquer approach**. Proceedings of the 2010 Fifth IEEE International Symposium on Service Oriented System Engineering, 2010, pp. 47-54.

LUTHRIA, H., RABHI, F., **Service Oriented Computing in Practice: An Agenda for Research into the Factors Influencing the Organizational Adoption of Service Oriented Architectures**, Journal of theoretical and applied electronic commerce research, 2009.

MAJLESI, S.; MEHRPOUR, A.; MOHSENZADEH, M.; **An approach for agile SOA development using agile principals**, International Journal of Computer Science & Information Technology (IJCSIT) Vol 4, No 1, Feb 2012

MAHMOOD, K., ILAHI, M. M., AHMAD, B., AHMAD, S.. **Empirical analysis of function points in service oriented architecture (SOA) applications**. Industrial Engineering Letters. 2012.

MAMANI, N. A., **Integrando requisitos não funcionais aos requisitos baseados em ações concretas**. Dissertação de mestrado, Departamento de Informática – PUC, RIO de Janeiro, 1999.

MANES, A. T,. **Registry Services: The Foundation for SOA Governance**. The Burton Group, 2007.

MARKS, E. A.; BELL, M., **Service-Oriented Architecture: a planning and implementation guide for business and techonology**. John Willey & Sons Inc, 2006.

MARTENSEN, A.; DAHLGAARD, J. J. **Strategy and planning for innovation management - supported by creative and learning organisations**. International Journal of Quality & Reliability Management, v. 16, n. 9, 2005, p. 878-891.

MARZULLO, Fabio Perez. **SOA na prática: inovando seu negócio por meio de soluções orientadas a serviços**. São Paulo: Novatec Editora, 2009.

MCBRIDE, G., (2007) **The role of SOA quality management in SOA service lifecycle management**. Developer Works, Rational SOA Go to Market Manager, IBM. Disponível em: ftp://ftp.software.ibm.com/software/rational/web/articles/soa_quality.pdf . Acessado em: 15 de agosto 2015.

MCCONNELL, S.,. **Software Estimation, Demystifying the Black Art**. Microsoft Press. 2006.

MEZO, B. M.; CHAPARRO, T. S.; HERAS, A. D., **Características de las empresas que utilizan Arquitectura Orientada a Servicios y de su contexto de operación**. Journal of Information Systems and Technology Management, v. 5, n. 2, 2008, p. 269-304

MEULEN, R., & PETTEY, C. (2009). **Gartner Survey Shows 40 Per Cent of SOA Users Don't Measure Time to Achieve Return on Investment**. Disponível em http://www.gartner.com/newsroom/id/978712, Acesso em: 15 julho de 2015.

MIN, L.; LIANG-JIE, Z.; FENGYUN, L. **An Insuanrance Model for Guaranteeing Service Assurance, Integrity and QoS in Cloud Computing**. Web Services (ICWS), 2010 IEEE International Conference on. 2010, p.584-591.

MONSALVE M., **Managing the QoS of E-Government: Metrics for Large Scale SOA,** XXVI International Conference of the Chilean Computer Science Society. 2007

MPS.BR **Associação para Promoção da Excelência do Software Brasileiro, MPS.BR - Melhoria de Processo de Software e Serviços Brasília 2013**. Disponível em: http://www.softex.br/mpsbr /_guias/guias/MPS.BR_Guia_Geral_Software_2013.pdf, Acesso em: 25 de junho de 2015.

O'BRIEN, L., **A framework for scope, cost and effort estimation for service oriented architecture (SOA) projects**, Proceedings of the 20th Australian Software Engineering Conference (ASWEC '09), IEEE Computer Society, Gold Coast, Australia, 2009, p.101–110.

O'BRIEN, L., MERSON P., BASS L., **Quality attributes for service oriented architectures**, International Workshop on Systems Development in SOA Environments. IEEE Publisher, 2007.

OFFICE, U. M. UK IT Association. 2017. http://www.ukita.co.uk/what-is-ukita/

PAPAZOGLOU, M. P. **Web Services & SOA, Principles and Technology**. Pearson Education, 2012.

PAPAZOGLOU, M. P.; VAN DEN HEUVEL, W. J. **Service-Oriented Design and Development Methodology**. International Journal of Web Engineering and Technology (IJWET), v. 2, no. 4, 2006, p. 412-442.

PEREPLETCHIKOV M., RYAN C., and TARI Z., **An analytical framework for evaluating service-oriented software development methodologies,** Journal of Software, V8 (7), 2013, pp. 1642-1659.

PERIN A. S.; RABELO, R. J. **An Approach for a More Agile BPM-SOA Integration Supported by Dynamic Services Discovery**. In Proc. of the EDOC 14th IEEE, 2010.

POULIN, J., HIMLER, A. (2006). **The ROI of SOA based on traditional component reuse**. Disponível em: http://semanticommunity.info/ @api/deki/files/2729/=ROI_of_SOA.pdf, Acesso em: 18 de julho 2015.

PMI – **Project Management Institute. PMBOK Guide – um guia para o conjunto de conhecimentos em Gerenciamento de Projetos**; versão oficial em inglês. 5ª ed. Filadélfia: PMI, 2013.

PRESSMAN S. Roger, **Engenharia de Software. Uma Abordagem Profissional**, Porto Alegre, Editora AMGH, 2016

PRIKLADNICKI, R.; WILLI, R.; MILANI, F., **Métodos Ágeis para Desenvolvimento de Software**. Bookman Editora. 2014

PUTNAM, Lawrence H.. **A General Empirical Solution to the Macro Software Sizing and Estimating Problem**. Putnam Estimates IEEE Transactions Software Engineering, V.4 N.4, 1978. p 345 – 361.

PUSCHMANN, T.; ALT, R. (2001) **Enterprise Application Integration - The Case of the Robert Bosch Group**. In: Proc. 34th Hawaii International Conference on System Sciences, 2001, p. 1-10

RABELO, Ricardo J.; NORAN, Ovidiu; BERNUS, Peter. **Towards the Next Generation Service Oriented Enterprise Architecture**, Service-oriented Enterprise Architecture for Enterprise Engineering-SoEA4EE,Adelaide, Australia, 2015.

RABELO, R. **Advanced Collaborative Business Ict Infrastructures**. In: SPRINGER (Ed.). Methods and Tools for Collaborative Networked Organizations, 2008. p.337-370.

RABELO, R.; GUSMEROLI, S.; NAGELLEN, T. *et al*. **An Evolving Plug and Play Business Infrastructure for Networked Organizations**. International Journal on Information Technology and Management, 2008.

REN, Z.; ANUMBA, C. J.; HASSAN, T. M. *et al*. **Collaborative project planning: A case study of seismic risk analysis using an e-engineering hub. Computers in Industry**, v. 57, n. 3, 2006, p. 218-230.

RICHARDSON, I. **SPI Models: What Characteristics are Required for Small Software Development Companies**. Software Quality Journal, 2002, p. 101-114.

ROSEN M., LUBLINSKY B., SMITH K. T., BALCER M. J., **SOA service-oriented architecture ans design strategies**, Wiley 2008

ROMERO, D.; RABELO, R. J.; MOLINA, A. **On the management of virtual enterprise's inheritance between virtual manufacturing & service enterprises: Supporting "Dynamic" product-service business ecosystems**. Engineering, Technology and Innovation (ICE), 2012 18th International ICE Conference on, 2012. June 2012. p.1-11.

ROSTAMPOUR A., KAZEMI A., SHAMS F., JAMSHIDI P., AZIZKANDI A. N., **Measures of structural complexity and service autonomy**, 2011, p. 1462-1467

ROY, B. **Multicriteria Methodology for Decision Aiding**. Netherlands: Kluwer Academic Publishers, 1996.

SAATY, T. L. **The Analytic Hierarquic Process.** Pittsburg: RWS Publications, 1980.

SAMPAIO, C., **SOA e Web Services em Java**. Brasport, 2006

SANTANNA, J. F.; RABELO, R. J.; BERNUS, P.; KLEN, A.P. **Improving the sustainability of SOA providers' networks via a collaborative process innovation model**. Anais APMS'2016 - IFIP International Conference on Advances in Production Management Systems, p. 1-9. Springer, 2016.

SATTY, T. L., **Decision Making for Leaders**, Pittsburgh: RWS Publications, 2001

SCHEPERS, T. G. J., IACOB, M. E., VAN ECK, P. A. T. **A lifecycle approach to SOA governance**. Proceedings of the 2008 ACM symposium on Applied computing, Brasil, 2008, p. 1055-1061.

SERMAN, D. V. **Estratégias de TI para a integração eletrônica da informação: um estudo sobre o estado da arte e da prática**. 2007. 119 p. Dissertação (Mestrado em Administração)-Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007.

SETH, A., AGRAWAL, H., SINGLA, A. R., **Techniques for evaluating service oriented systems: A Comparative Study**. Journal of Industrial and Intelligent Information, 2014.

SHUJA, A. K. , KREBS J., **IBM Rational Unified Process Reference and Certification Guide: Solution Designer,** IBM Press book, 2007.

SILVA, E. L. D.; MENEZES, E. M. **Metodologia da Pesquisa e Elaboração de Dissertação**. Editora da UFSC, 2005.

SOMMERVILLE, I. **Engenharia de Software Orienda a Serviços**. In: HALL, P. P. (Ed.). Engenharia de Software: 5ª edição 2006.

SOMMERVILLE, I. **Engenharia de software**, São Paulo: Editora Pearson, 2011.

SORDI, J., MARINHO, B., NAGY, M., **Benefícios da Arquitetura de Software Orientada a Serviços para as empresas: análise da experiência do ABN AMRO Brasil**. Journal of Information Systems and Technology Management, v3 (1), 2006, p. 19-34.

SOUZA K., FANTINATO M., **WS&i*-RGPS: An Approach to Service-Oriented Requirements Engineering Based on RGPS Metamodels**, IEEE 8th International Symposium on Service Oriented System Engineering (SOSE), 2014, pp. 76-81.

STAL, M., **Web Services: Beyond Component-Based Computing**. Communications of the ACM, v. 45, n. 10, 2002, p. 71-78

STRNADL, C. F., **Bridging architectural boundaries design and implementation of a semantic BPM and SOA Governance Tool**, in

ICSOC ’07: Proceedings of the 5th international conference on Service-Oriented Computing, pages 518–529, Berlin, Heidelberg, 2007, Springer.

SVELTO, C.; MATTEUCCI, M.; RESMINI, R.; PNIOV, A.; PEDOTTI, L.; GIORDANO, F. **Semi-and-automatic wheal measurement system for prick test digital imaging and analysis**. In: 2016 IEEE International Conference on Imaging Systems and Techniques (IST), 2016. 4-6 Oct. 2016. p.482-486.

TANSEY, B., STROULIA, E., **Valuating software service development: Integrating COCOMO II and real options theory**. Proceedings of the First International Workshop on The Economics of Software and Computation. 2007.

TSAI, W. T., JIN, Z., WANG, P., WU, B., **Requirement Engineering in Service Oriented System Engineering**, In Proc. of Int. Conf. on e-Business Engineering, 2007, p. 661 – 668.

TUDEVDAGVA, U.; LKHAGVASUREN, B. E. **Application of the structure oriented evaluation model for faculty members self-assessment**. In: 2016 11th International Forum on Strategic Technology (IFOST), 2016. p.448-451.

UKSMA. (1998). **MK II Function Point Analysis: Counting Practices Manual**, Version 1.3.1. United Kingdom Software Metrics Association (UKSMA). Disponível em: http://www.gifpa.co.uk/library/Resources/MkIIr131.pdf. Acesso em: 21 de maio de 2015.

VAN DEN HONERT, R. C. **Decisional Power in Group Decision Making: A Note on the Allocation of Group Member’s Weight in the Multiplicative AHP and SMART**. Group Decision and Negociation, n.10. 2001, p. 275-286.

VINCKE, P., **Multicriteria decision aid**. Bruxelles: John Wiley & Sons, Inc. 1992.

WANG, L.; VON LASZEWSKI, G.; YOUNGE, A. et al. **Cloud Computing: a Perspective Study**. New Generation Computing, v. 28, n. 2, p. 137-146, 2009. Disponível em: http://dx.doi.org/10.1007/s00354-008-0081-5, Acesso em: 21 de Abril 2014.

WANGENHEIM, C. G. V.; HAUCK, J. C. R.; ZOUCAS, A. *et al*. **Creating Software Process Capability/Maturity Models**. Software, IEEE, v. 27, n. 4, 2010, p. 92-94.

WEBMETHODS, S**OA Governance - Enabling Sustainable Success with SOA**, white paper, 2006, p.20.

WESTPHAL, I.; THOBEN, K. D.; SEIFERT, M. **Managing collaboration performance to govern virtual organizations**. Journal of Intelligent Manufacturing, 21-3, 2010, p. 311-320.

WITTERN, E., FISCHER R., **A Life-Cycle Model for Software Service Engineering, European Conference on Service-Oriented and Cloud Computing**, ESOCC 2013: Service-Oriented and Cloud Computing, 2013, p. 164-171.

WOOLF, B. **Introduction to SOA governance. Governance: The Official IBM Definition, and Why You Need It**, IBM, http://www.ibm.com/developerworks/library/ar-servgov/. 2006

ZAYARAZ, G.; THAMBIDURAI, P. **Quantitative Measurement of Software Architectural Qualities through COSMIC FFP**. In: 2006 Annual IEEE India Conference, 2006. p.1-4.

ZELKOWITZ, M. V. **Techniques for Empirical validation**. In: BASILI, V. R. et al. (Eds.). Empirical Software Engineering Issues. Critical Assessment and Future Directions. NY: Springer, 2007. p. 4-9.

ZIMMERMANN, O. et al. **Analysis and Design Techniques for Service-Oriented Development and Integration**. Stuttgart, Alemanha: IBM Press, 2005.

ZIMMERMANN O., KROGDAHL P., GEE C., **Elements of Service-Oriented Analysis and Design**, Developer Works, IBM Corporation, 2004.

APÊNDICE A– DETALHAMENTO DO SLR

String principal de busca:

(“SOA implementation methodology” OR “SOA model” OR “model for SOA” OR “SOA framework” OR “SOA guideline” OR “SOA implementation guide”) AND
(“software process” OR ”software processes” OR ”BPM”) AND
(SaaS OR "Software-as-a-Service" OR "Software as a Service" OR Services OR SOA OR "Service Oriented Architecture" OR "Service-oriented architecture" OR "Service Oriented Integration") AND
(“cost of” AND “SOA”) AND
(“effort of” AND “SOA”) AND
(“risk of” AND “SOA”) AND
(“metodologia da implantação SOA” OR “modelo SOA” OR “modelo aplicado em SOA” OR “Framework SOA” OR “Guia da implementação SOA”) AND
(“Processo de software” OR ”processo de implementação” OR ”BPM”) AND
(SOA AND "Arquitetura orientada a Serviços") AND
(“custo” AND “SOA”) AND
(“esforço” And “SOA”) AND
(“risco” And “SOA”)

Para a elaboração desta *string*, foram considerados os termos principais desta proposta de tese, considerando as palavras similares e as abreviaturas.

Esta *string* é formada por doze trechos de condições separadas por AND, ou seja, os doze trechos devem ter alguma *string* selecionada. Dentro de cada trecho, *string*s separadas por OR, ou seja, qualquer uma dessas *string*s deve aparecer no resultado.

Além de uma *string* ser elaborada, alinhada com conteúdo da proposta de teste, por vezes a busca retorna trabalhos que não são o foco do SLR. Para isso, foram elaborados os seguintes critérios de exclusão, ou seja, critérios que fazem com que trabalhos retornados na busca sejam eliminados:

- CE1: foco em processos específicos (ex. segurança, requisitos, ESB, repositório de ativos);

- CE2: foco em questões técnicas de desenvolvimento (linguagens de programação) ou infraestrutura, como plataforma de hardware;
- CE3: trabalhos que relatam o desenvolvimento de um sistema utilizando SOA, sem considerar os aspectos de adoção SOA nas organizações;
- CE4: short papers com menos de três páginas.

A *string* foi adaptada conforme cada base e sistema de busca. Cada base possui uma busca avançada que aceita *string*s como argumento para o sistema de busca, além de filtros para possibilitar melhor seleção dos resultados, mas a estrutura dessa *string* varia de uma base para outra.

As *string*s de cada base são mostradas na tabela a seguir:

Base da Busca	***String***
IEEExplore	(“SOA implementation methodology” OR “SOA model” OR “model for SOA” OR “SOA framework” OR “SOA guideline” OR “SOA implementation guide”) AND (“software process” OR ”software processes” OR ”BPM”) AND (SaaS OR "Software-as-a-Service" OR "Software as a Service" OR "Services" OR SOA OR "Service Oriented Architecture" OR "Service-oriented architecture" OR "Service Oriented Integration") AND (“cost of” and “SOA”) AND (“effort of” and “SOA”) AND(“risk of” and “SOA”)
ACM Digital Library	*(((Abstract:"implementation methodology") OR (Abstract:"SOA model") OR (Abstract:"* model for SOA*") OR (Abstract:"*SOA framework*") OR (Abstract:"*SOA framework*") OR (Abstract:" SOA guideline") OR (Abstract:" SOA implementation guide”)) AND ((Abstract: “software process”) OR (Abstract:”software processes”) OR (Abstract:”BPM”))) AND ((Abstract:SaaS) OR (Abstract:"Software-as-a-Service") OR (Abstract:"Software as a Service") OR (Abstract:Services) OR (Abstract:SOA) OR (Abstract:"Service Oriented Architecture") OR (Abstract:"Service-oriented architecture")) OR ((Abstract:" Service Oriented Integration")) AND*

	((Abstract: "cost of" AND "SOA") AND (Abstract: "effort of" And "SOA") AND (Abstract: "risk of" And "SOA"))))
ScienceDirect	*(("SOA implementation methodology" OR "SOA model" OR "model for SOA" OR "SOA framework" OR "SOA guideline" OR "SOA implementation guide") AND ("software process" OR "software processes" OR "BPM") AND (SaaS OR "Software-as-a-Service" OR "Software as a Service" OR Services OR SOA OR "Service Oriented Architecture" OR "Service-oriented architecture" OR "Service Oriented Integration") AND ("cost of" AND "SOA") AND ("effort of" And "SOA") AND ("risk of" And "SOA"))*
UFRGS – Tese de doutorado	*("SOA implementation methodology" OR "SOA model" OR "model for SOA" OR "SOA framework" OR "SOA guideline" OR "SOA implementation guide") AND ("software process" OR "software processes" OR "BPM") AND (SaaS OR "Software-as-a-Service" OR "Software as a Service" OR Services OR SOA OR "Service Oriented Architecture" OR "Service-oriented architecture" OR "Service Oriented Integration") AND ("cost of" AND "SOA") AND ("effort of" And "SOA") AND ("risk of" And "SOA") AND ("metodologia da implantação SOA" OR "modelo SOA" OR "modelo aplicado em SOA" OR "Framework SOA" OR "Guia da implementação SOA") AND ("Processo de software" OR "processo de implmentação" OR "BPM") AND (SOA AND "Arquitetura orientada a Serviços") AND ("custo" AND "SOA") AND ("esforço" And "SOA") AND ("risco" And "SOA")*
UFPE – Tese de doutorado	*("SOA implementation methodology" OR "SOA model" OR "model for SOA" OR "SOA framework" OR "SOA guideline" OR "SOA implementation guide") AND ("software process" OR "software processes" OR "BPM") AND (SaaS OR "Software-as-a-Service" OR "Software as a Service" OR Services OR SOA OR "Service Oriented Architecture" OR "Service-oriented architecture" OR "Service Oriented Integration") AND ("cost of" AND "SOA") AND ("effort of" And "SOA") AND ("risk of" And "SOA") AND ("metodologia da*

	implantação SOA" OR "modelo SOA" OR "modelo aplicado em SOA" OR "Framework SOA" OR "Guia da implementação SOA") AND ("Processo de software" OR "processo de implmentação" OR "BPM") AND (SOA AND "Arquitetura orientada a Serviços") AND ("custo" AND "SOA") AND ("esforço" And "SOA") AND ("risco" And "SOA")
USP – Tese de doutorado	*("SOA implementation methodology" OR "SOA model" OR "model for SOA" OR "SOA framework" OR "SOA guideline" OR "SOA implementation guide") AND ("software process" OR "software processes" OR "BPM") AND (SaaS OR "Software-as-a-Service" OR "Software as a Service" OR Services OR SOA OR "Service Oriented Architecture" OR "Service-oriented architecture" OR "Service Oriented Integration") AND ("cost of" AND "SOA") AND ("effort of" And "SOA") AND ("risk of" And "SOA") AND ("metodologia da implantação SOA" OR "modelo SOA" OR "modelo aplicado em SOA" OR "Framework SOA" OR "Guia da implementação SOA") AND ("Processo de software" OR "processo de implmentação" OR "BPM") AND (SOA AND "Arquitetura orientada a Serviços") AND ("custo" AND "SOA") AND ("esforço" And "SOA") AND ("risco" And "SOA")*
UFSC/ Ciências da Computação– Tese doutorado	*("SOA implementation methodology" OR "SOA model" OR "model for SOA" OR "SOA framework" OR "SOA guideline" OR "SOA implementation guide") AND ("software process" OR "software processes" OR "BPM") AND (SaaS OR "Software-as-a-Service" OR "Software as a Service" OR Services OR SOA OR "Service Oriented Architecture" OR "Service-oriented architecture" OR "Service Oriented Integration") AND ("cost of" AND "SOA") AND ("effort of" And "SOA") AND ("risk of" And "SOA") AND ("metodologia da implantação SOA" OR "modelo SOA" OR "modelo aplicado em SOA" OR "Framework SOA" OR "Guia da implementação SOA") AND ("Processo de software" OR "processo de implmentação" OR "BPM") AND (SOA AND "Arquitetura orientada a Serviços") AND ("custo" AND "SOA") AND*

	(“esforço” And “SOA”) AND (“risco” And “SOA”)
UFSC/ Engenharia de Automação e sistemas – Tese doutorado	*(“SOA implementation methodology” OR “SOA model” OR “model for SOA” OR “SOA framework” OR “SOA guideline” OR “SOA implementation guide”) AND (“software process” OR ”software processes” OR ”BPM”) AND (SaaS OR "Software-as-a-Service" OR "Software as a Service" OR Services OR SOA OR "Service Oriented Architecture" OR "Service-oriented architecture" OR "Service Oriented Integration") AND (“cost of” AND “SOA”) AND (“effort of” And “SOA”) AND (“risk of” And “SOA”) AND (“metodologia da implantação SOA” OR “modelo SOA” OR “modelo aplicado em SOA” OR “Framework SOA” OR “Guia da implementação SOA”) AND (“Processo de software” OR ”processo de implmentação” OR ”BPM”) AND (SOA AND "Arquitetura orientada a Serviços") AND (“custo” AND “SOA”) AND (“esforço” And “SOA”) AND (“risco” And “SOA”)*

APÊNDICE B– QUESTIONÁRIO APLICADO NA AVALIAÇÃO

Nome:
Email:
Empresa:

Observações:
- A identificação das pessoas e do nome das empresas serão mantidas em sigilo e não serão divulgadas em nenhuma publicação.
- O uso das informações deste questionário servirá exclusivamente à avaliação do modelo de processos de inovação proposto.

Sobre o avaliador:

1. Seu conhecimento sobre processo de estimação de software em geral:

Pouco

1	2	3	4	5

Muito

2. Seu conhecimento em SOA ou em sistemas / modelos de negócios baseados em serviços de software:

Pouco

1	2	3	4	5

Muito

Sobre a empresa:

3. A sua empresa utiliza algum método formal ou sistematizado de apoio aos gestores de TI na estimação do esforço, custo, tempo de desenvolvimento e/ou de alinhamento ao negócio de projetos SOA?

	O método faz alguma comparação com desenvolvimento tradicional?
Sim	

Não	Como é então basicamente estimado o custo, esforço, tempo de desenvolvimento e/ou alinhamento ao negócio ?

Sobre o Método e avaliação:

4. Você considera que a sistematização provida pelo método no processo de estimação contribui para uma padronização deste na empresa e assim pode se tornar uma prática dentro de um processo maior de governança de SOA & TI?

Concordo totalmente	Concordo Parcialmente	Indiferente	Discordo Parcialmente	Discordo totalmente
Observação:				

5. Você considera que o método desenvolvido estima com o devido rigor técnico / corretude um projeto de software baseado em serviços em uma perspectiva SOA?

Concordo totalmente	Concordo Parcialmente	Não consigo avaliar	Discordo Parcialmente	Discordo totalmente
Observação:				

6. Você considera que o método desenvolvido estima com a devida abrangência os vários aspectos de um projeto de software baseado em serviços em uma perspectiva SOA?

Concordo totalmente	Concordo Parcialmente	Não consigo avaliar	Discordo Parcialmente	Discordo totalmente
Observação:				

7. Você considera que o método desenvolvido diminui a subjetividade na estimação e assim dá maior confiança na decisão?

Concordo totalmente	Concordo Parcialmente	Indiferente	Discordo Parcialmente	Discordo totalmente
Observação:				

8. Você considera que o método desenvolvido ajuda no refinamento e/ou definição da granularidade dos serviços e decisões de reuso ?

Concordo totalmente	Concordo Parcialmente	Indiferente	Discordo Parcialmente	Discordo totalmente
Observação:				

9. Você acha que faz sentido poder dar pesos de importância diferentes aos vários fatores do método para cada projeto SOA em específico?

Concordo totalmente	Concordo Parcialmente	Indiferente	Discordo Parcialmente	Discordo totalmente
Observação:				

10. Avaliando o método proposto e considerando de forma *genérica* um dado projeto SOA, como você classificaria o grau de importância dos fatores cobertos pelo método no processo de estimação?

Fator / Avaliação	Essencial	Importante	Pouco relevante	Não necessário	Incorreto
Requisitos Funcionais					
Complexidade Integração					
Experiência Profissional					
Condição Ambiental					
Complexidade Ambiental					
Requisitos Não Funcionais					
Complexidade Funcional					
Tamanho					
Reuso					
Esforço					
Custo					
Tempo de desenvolvimento					
Alinhamento Negócio					

11. Você acha que haveria outros fatores necessários de serem mensurados para estimação de projetos SOA? Se sim, quais?

12. Você acha que o método tem o potencial de proporcionar as seguintes melhorias?

Melhoria / Avaliação	Concordo totalmente	Concordo Parcialmente	Não sei	Discordo Parcialmente	Discordo totalmente
Maior rapidez no processo de estimação					
Melhor base para gestão de projetos					
Ajuda como referencial de custos e tempos numa terceirização ou subcontratação					
Se tornar uma ferramenta de apoio a treinamento em estimação					

Aspectos gerais do Método:

13. Quais são as principais dificuldades (de qualquer tipo) que você vislumbra na implementação do método proposto na sua empresa ou nas empresas de software em geral?

14. Se desejar fique livre para comentar sobre o método de estimação proposto.