UNIVERSIDADE FEDERAL DO PARÁ
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RENATO DE PINA FERREIRA

Um Estudo Exploratório dos Fatores de Adoção de Plataformas de *Software* Móveis

Dissertação de Mestrado

Belém
2016

UNIVERSIDADE FEDERAL DO PARÁ
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RENATO DE PINA FERREIRA

Um Estudo Exploratório dos Fatores de Adoção de Plataformas de *Software* Móveis

Dissertação submetida à Banca Examinadora do Programa de Pós-Graduação em Ciência da Computação da UFPA para a obtenção do Grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Cleidson Ronald Botelho de Souza

Belém
2016

Belém, Janeiro de 2016

UNIVERSIDADE FEDERAL DO PARÁ

INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Um Estudo Exploratório dos Fatores de Adoção de Plataformas de *Software* Móveis

AUTOR: RENATO DE PINA FERREIRA

DISSERTAÇÃO DE MESTRADO SUBMETIDA À AVALIAÇÃO DA BANCA EXAMINADORA A SER AVALIADA PELO COLEGIADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO DA UNIVERSIDADE FEDERAL DO PARÁ.

BANCA EXAMINADORA:

_____

Prof. Dr. Cleidson R. B. de Souza

(ORIENTADOR – UFPA – Instituto de Ciências Exatas e Naturais)

_____

Prof. Dr. Fernando Figueira

(UFRN – Departamento de Informática e Matemática Aplicada)

_____

Profa. Dra. Sabrina Marczak

(PUC-RS – Instituto de Computação)

Dedico estre trabalho à minha famílIa

AGRADECIMENTOS

Agradeço primeiramente a Deus, à minha família e a todos os envolvidos no meu processo de formação, todos que me incentivaram e me apoiaram sempre que precisei. Agradeço ao meu pai e à minha mãe, as pessoas mais importantes da minha vida, que sempre me deram amor e bons exemplos, sempre ao meu lado nos momentos bons e ruins, nunca deixando de acreditar em mim. Ao meu irmão por estar sempre comigo me ajudando em tudo que pôde sempre me fornecendo bons conselhos e sempre me inspirando a ser uma pessoa melhor e buscar meus objetivos, além de ser um modelo de pessoa que eu desejo me tornar um dia – obrigado por seu meu ídolo. À minha irmã e madrinha, que sempre me deu muito amor e carinho, que sempre foi meu porto seguro em situações de estresse, que sempre acreditou no meu potencial e nunca se absteve de me dar seus conselhos mais sinceros. Ao meu pequeno sobrinho Luis Felipe, que me alegra com sua inocência e a quem procuro sempre ser um modelo de ser humano. A Luciene, pessoa que ajudou a me criar, sempre amorosa e cuidadosa, que há muito tempo já faz parte da minha família e a quem eu sempre serei grato por ter cuidado de mim desde muito pequeno. À minha namorada, que ilumina meus dias, me dá força para continuar e é a pessoa que me motiva a sempre procurar ser uma pessoa melhor. Ao meu orientador, Prof. Dr. Cleidson R.B. de Souza pela orientação, por ter acreditado naquele rapaz que nunca tinha visto na vida, por sempre me colocar no caminho certo e principalmente pela sua confiança e paciência. A todos os membros do laboratório de computação do Instituto Tecnológico Vale, por sempre me fornecerem incentivo, boas risadas e um ambiente excelente de trabalho, sem falar nas experiências e no aprendizado que eu tenho certeza que eu não teria conseguido em outro lugar. Ao Müller Miranda, meu caro colega de mestrado, que me ajudou em diversas partes deste mestrado, que esteve comigo nos momentos mais complicados desta jornada e sempre me forneceu sua visão metódica dos fatos - sua visão aliada a minha nos ajudou a crescer junto como pessoas e pesquisadores.

RESUMO

Existem várias plataformas de *software* que concorrem umas com as outras para atrair engenheiros de *software*. No entanto, ainda não são bem conhecidos quais fatores os engenheiros de *software* levam em consideração ao decidir adotar uma determinada plataforma. Esta dissertação apresenta um estudo exploratório que visa preencher esta lacuna. Entrevistas semiestruturadas foram realizadas com 18 engenheiros de *software* utilizando a Teoria de Difusão da Inovação de Rogers como *framework* conceitual. Tais entrevistas foram analisadas utilizando técnicas da Teoria Fundamentada em Dados. Os resultados desta dissertação ilustram alguns dos fatores que levam um engenheiro de *software* a adotar ou não uma determinada plataforma. Por exemplo, engenheiros de *software* percebem a plataforma Android como mais acessível e compatível com o seus conhecimentos existentes, mas eles temem sua fragmentação (alto número de dispositivos). Alguns engenheiros de *software* simplesmente escolhem iOS porque as vendas são maiores nessa plataforma. As conclusões apresentadas neste trabalho podem ajudar os engenheiros de *software* a decidir qual plataforma adotar e também podem auxiliar os responsáveis por ecossistemas de *software* a aperfeiçoar suas plataformas para atrair mais engenheiros de *software*.

Palavras-chave: Ecossistemas de *Software*, Adoção, Teoria de Difusão da Inovação, Plataformas.

ABSTRACT

There are several development platforms that compete with one another to attract software engineers. However, it is not well known which factors software engineers take into consideration when deciding to adopt a particular platform. This thesis presentsan exploratory study to fill in this gap. Semi-structured interviews were conducted with 18 software engineers using the Diffusion of Innovation Theory as a conceptual framework. These interviews were analyzed using techniques of Grounded Theory. Results illustrate some of the factors that lead a software engineer to adopt or not a particular platform. For instance, we uncovered that developers perceive the Android platform as more accessible and compatible with their existing knowledge, however they fear its fragmentation (high number of devices). Some developers choose iOS because sales are more lucrative on that platform. The findings presented in this work can help software engineers to decide which platform to adopt and can also help the owners of platforms to improve their platform to attract more software engineers.

Keywords: Software Ecosystem, Adoption, Diffusion of Innovations, Platforms.

Ad astra per aspera.

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

VISÃO GERAL

No mundo moderno em que vivemos, os ecossistemas de *software* estão inseridos em vários aspectos. Vários anos atrás, o esforço para desenvolvimento de *software* era algo, geralmente, centralizado em uma única empresa. Todo o código gerado, todos os artefatos de entrada e saída eram frutos do trabalho de uma única organização. Todo o mérito de lançar algo no mercado era advento de um esforço solitário. Porém, nos dias atuais isso mudou, as empresas colaboram entre si, de forma direta ou indireta, para lançar seus produtos de *software*, muitas vezes recorrendo ao conhecimento – seja ele de qualquer natureza: algoritmos, manuais, dicas, etc – já disseminado por outras empresas de desenvolvimento de *software*. Este é apenas um aspecto em que os ecossistemas de *software* mudaram a forma como o desenvolvimento de *software* é realizado.

Desenvolver hoje em dia de forma isolada é algo que não pode passar mais pela cabeça dos engenheiros de *software*, pois significa uma perda de tempo enorme. Códigos, casos de uso, diagramas de classe, requisitos e documentos de projeto, manuais, ferramentas e tudo que permeia o desenvolvimento de *software* estão aí para serem utilizados, os ecossistemas de *software* permitem isso. Se isolar e fazer tudo sozinho não é mais uma opção viável para os engenheiros que querem ser bem sucedidos. Os ecossistemas de *software* prevalecem no mundo moderno. Todos os engenheiros de *software* estão inseridos em um ecossistema de *software*, quer eles saibam ou não.

Bosch e Petra (2010) definem ecossistema de *software* como um conjunto de soluções de *softwares* e uma comunidade especialista em serviços para a comunidade de usuários, que compõe soluções relevantes para as suas necessidades. Google Android, Apple iOS, Libre Office e Linux são alguns exemplos destes ecossistemas. Tratando-se de ecossistemas de *softwares* para dispositivos móveis pode-se dizer que um dos fatores de sucesso está ligado ao esforço para criação de novas aplicações para o mesmo, pois tais aplicações refletem em soluções para a comunidade de usuários, e este esforço pode estar associado à complexidade de utilização das ferramentas para o rápido desenvolvimento de novos produtos, por exemplo. Entre essas ferramentas pode-se identificar o *Software Development Kit* (SDK), que é o pacote de instalação para que engenheiros de software tenham integração com o ecossistema proposto para o desenvolvimento de aplicações.

Além de ferramentas, existem outros fatores determinantes para a adoção de um ecossistema de *software* por engenheiros de software, por exemplo, os possíveis benefícios financeiros, a documentação das APIs do ecossistema, etc, fatores que serão apresentados com riqueza de detalhes no Capítulo 4, com os resultados.

Assim, com este trabalho pretende-se compreender as diversas características que podem ser determinantes para a adoção ou rejeição de um ecossistema de *software mobile*. Isto é feito a partir da teoria de inovação de Everett Rogers (2003), sob a ótica de engenheiros de software. A teoria de inovação de Rogers foi desenvolvida para estudar o fenômeno da difusão de inovações, e considera ideias, práticas ou objetos que sejam percebidos como novos por quem os adota, portanto é uma teoria geral que pode ser aplicada em diversas situações.

1.1. MOTIVAÇÃO

1.1.1. Motivação Acadêmica

Nos primórdios do desenvolvimento de *software*, um produto de *software* era o resultado de um esforço de um fabricante independente, trabalhando como uma unidade

isolada, sem contar com outros fabricantes para obter sucesso. Porém, com o avanço da globalização dos mercados, a competição ficou mais disputada levando a uma mudança de paradigma em que fabricantes passam a depender fortemente de componentes de *software* e infraestrutura de terceiros para que seus *softwares* possam ser produzidos (Cusumano, 2004). Assim, as empresas vendedoras de *software* evoluíram de unidades independentes para uma rede interdependente de fornecedores de serviços e *software*. Devido a este novo cenário, a integração de produtos de software e serviços vem demandando cada vez mais atenção de técnicos e gestores de empresas que desenvolvem e usam tecnologias (Cusumano, 2004). Portanto, esta nova forma de relacionamento entre empresas de desenvolvimento de *software* e empresas que produzem componentes e infraestrutura, moldou o cenário atual, criando um novo modelo de negócio chamado ecossistemas de *software*.

Por se tratar de um tema relativamente novo, este novo paradigma demanda de novos estudos que ajudem a entender melhor esta nova forma de desenvolvimento. Pesquisas já foram feitas neste sentido, como Goadrich & Rogers (2011) que fazem uma análise comparativa entre dois ecossistemas de software (Android e iOS). Em Joorabchi *et al.* (2013), os autores buscam obter uma compreensão dos principais desafios técnicos enfrentados pelos engenheiros de *software* para o desenvolvimento de aplicativos para diferentes dispositivos móveis em plataformas nativas. Já Linares-Vásquez *et al.* (2013) reporta um estudo realizado em como a propensão à falhas e a propensão à mudanças em APIs afeta o sucesso de aplicativos móveis.

Nos trabalhos relatados acima o foco dos pesquisadores estava em aspectos técnicos de ecossistemas de *software*, como a propensão a falhas e desafios técnicos que são enfrentados pelos engenheiros de *software*. Tendo isto em mente, este trabalho está apoiado não só em analisar aspectos técnicos de ecossistemas de *software*, mas também aspectos sociais que influenciam na dinâmica de adoção ou rejeição de um determinado ecossistema, pois o desenvolvimento de *software* não é uma atividade puramente técnica, é realizada por seres humanos, que são sociais por natureza, e estes aspectos sociais precisam ser analisados de forma ampla (Christakis & Fowler, 2009). Para

analisar estes aspectos sociais e técnicos, é utilizada a Teoria da Difusão da Inovação, elaborada pelo sociólogo Everett Rogers, que é a teoria que usada como de *framework* conceitual para análise dos resultados.

1.1.2. Motivação Industrial

O panorama da telefonia móvel vem crescendo a cada ano, mais especificamente no mercado de *smartphones*, 55% das vendas relacionadas ao mercado móvel foram de *smartphones* só no terceiro trimestre de 2013 [1]. Segundo a *International Data Corporation* (IDC), empresa que realiza pesquisas de mercado, o mercado de *smartphones* superou a marca de um bilhão de aparelhos vendidos em 2013. Tal informação corrobora com o gráfico apresentado pelo instituto Gartner (Tabela 1), empresa que pesquisa tendências tecnológicas, em parceria com a VisionMobile, empresa que realiza análises sobre a economia aplicativos e modelos de negócios móveis.

Fabricante	3º Trimeste/2013 Unidades	3º Trimeste/2013 Market Share (%)	3º Trimeste/2012 Unidades	3º Trimeste/2012 Market Share (%)
Samsung	80,356.8	32.1	55,054.2	32.1
Apple	30,330.0	12.1	24,620.3	14.3
Lenovo	12,882.0	5.1	6,981.0	4.1
LG Eletronics	12,055.4	4.8	6,986.1	4.1
Huawei	11665.7	4.7	7,804.3	4.5
Outras	102941.8	41.1	70206.8	40.9

[1] http://www.gartner.com/newsroom/id/2623415

Total	**250,231.7**	**100.0**	**171,652.7**	**100.0**

Fonte: Gartner (2013)

Tabela 1: Crescimento de venda de *smartphones*.

De acordo com a Associação Brasileira da Indústria Elétrica e Eletrônica (Abinee), os *smarphones* representam 95% das vendas de celulares em 2015. Nos dois primeiros meses de 2015, foram comercializadas 8,5 milhões de unidades de *smartphones*, contra 539 mil aparelhos tradicionais, atingindo um total de 9 milhões de celulares vendidos. A expectativa era de que fossem vendidos 61,061 milhões de *smartphones* em 2014. Assim, os *smartphones* responderam por um total de 61% das vendas do setor de telefonia móvel no ano de 2014. Em 2015 esta porcentagem está em 95%, segundo o levantamento mais recente feito pela IDC e a Abinee. Isso só fortalece mais o crescimento desse mercado.

Percebendo este crescimento, engenheiros de *software*, empresas de tecnologia e comunicação vêm sendo atraídas pelas oportunidades de negócios geradas por esse mercado emergente. O fator que motiva este trabalho é a carência de estudos que busquem compreender o que leva desenvolvedores ou empresas a escolherem para qual plataforma (Android, iOS, Windows Phone, etc.) desenvolver. Também como citado anteriormente, existe essa nova denominação, esta nova forma de se produzir software, de forma distribuída em ecossistemas de *software*. Poucos estudos buscam compreender como se dá essa nova forma de desenvolvimento, como esses ecossistemas funcionam, as dinâmicas dentro dele e o que leva um desenvolvedor a escolher um ecossistema para desenvolver. Então, este também é um fator motivador para este trabalho.

1.2. Justificativa e Contribuição

Empresas vendedoras de produtos de software devem considerar seus papéis estratégicos em um novo modelo de negócio emergente: ecossistema de *software*. Este trabalho pretende revelar informações que podem trazer benefícios às empresas de

softwares de diferentes segmentos ao auxiliarem as mesmas na tomada de decisão quanto à política de adoção e desenvolvimento para ecossistemas de s*oftwares*, portanto, ajudando estas empresas a sobreviverem neste novo modelo de mercado. Além das empresas, este estudo também tem como objetivo entender o motivo que leva um engenheiro de software a adotar ou rejeitar um determinado ecossistema de software móvel.

Conforme discutido por Butle (2011) um estudo que revela a evolução do *market share* entre 2009 a 2010 de várias plataformas móveis, alguns ecossistemas de *software* móveis são de fato mais adotados que outros.

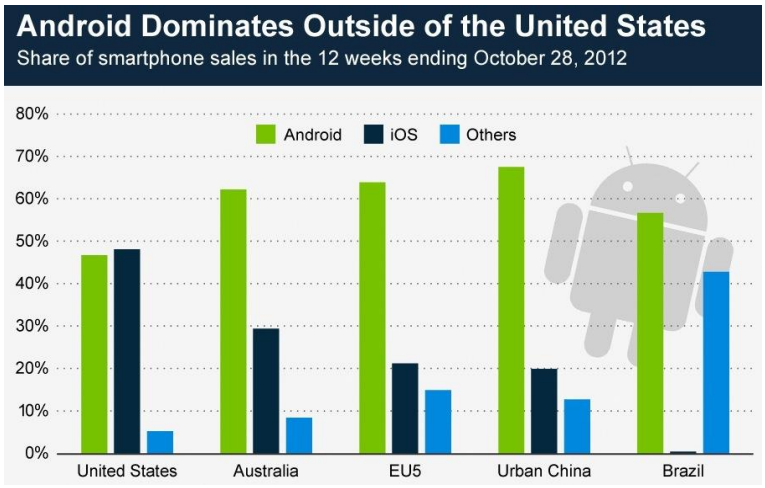

Figura 1: Este gráfico mostra a participação de mercado do sistema operacional para *smartphones*, com base nas vendas no período de 12 semanas entre Agosto e Outubro de 2012.

Dada a grande evolução dos ecossistemas de software, ilustrados pelas vendas dos mesmos em 2012 (Figura 1), se faz necessário um estudo que contribua para o entendimento deste novo modelo de negócio. Isso se faz necessário, principalmente para entender sobre a ótica dos engenheiros de software que desenvolvem para estes ecossistemas, os motivos que fizeram os mesmos a adotar ou rejeitar tais ecossistemas e motivos que os fizeram ingressar neste modelo de negócio.

1.3. Objetivo

Considerando o contexto da pesquisa e o problema apresentado, o objetivo geral desta dissertação é identificar quais os principais fatores que levam a um engenheiro de *software* a adotar ou rejeitar determinado ecossistema de *software* móvel. No caso desta

dissertação o Google Android e o Apple iOS serão os ecossistemas em questão. Em relação aos objetivos específicos, tem-se:

- Realizar uma revisão bibliográfica sobre Ecossistemas de *Software* e apresentar a Teoria de Difusão da Inovação (Rogers, 2003).
- Apresentar a aplicação de entrevistas semiestruturadas com os engenheiros de *software* e a realização da análise de dados utilizando técnicas da Teoria Fundamentada em Dados. Isto é feito a partir de uma pesquisa empírica;
- Discutir os resultados encontrados nesta dissertação com trabalhos semelhantes já disponíveis na literatura e apontar como eles se relacionam, assim contribuindo para a evolução do conhecimento em adoção de ecossistemas de *software*.

1.4. Organização do Trabalho

Esta dissertação está organizada em outros cinco capítulos além deste, que apresenta conceitos importantes acerca do tema abordado e também as motivações deste trabalho, a apresentação do problema que se pretende resolver, assim como os seus objetivos e organização deste documento.

O **Capítulo 2** realiza um levantamento sobre os principais conceitos, definições, estruturas, características e fundamentos teóricos de ecossistemas de software e explica a Teoria de Difusão da Inovação (Rogers, 2003).

O **Capítulo 3** apresenta a metodologia adotada para coleta e análise dos dados, descrevendo os principais conceitos relacionados e como a pesquisa foi conduzida.

O **Capítulo 4** apresenta os resultados encontrados durante a pesquisa, mapeados de acordo com os pilares presentes na Teoria de Difusão da Inovação (Rogers, 2003).

O **Capítulo 5** apresenta a discussão sobre os resultados encontrados nesta dissertação com a literatura existente.

Finalmente, o **Capítulo 6** conclui esta dissertação, apresentados as conclusões, indicando as principais contribuições deste trabalho e os trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. TEORIA DA DIFUSÃO DA INOVAÇÃO

2.1.1. VISÃO GERAL

Este capítulo apresentará alguns conceitos da Teoria da Difusão de Inovações, formulada por Everett Rogers, em 1962, teoria essa que é amplamente utilizada desde sua criação, nas mais diversas áreas do conhecimento. Os conceitos propostos por Rogers foram adotados como um dos principais referenciais teóricos deste trabalho, e será usado na análise do mesmo.

A Teoria da Difusão de Inovação desenvolvida por Everett Rogers busca explicar o processo de difusão de uma inovação, bem como o porquê de uma inovação ser aceita ou rejeitada por indivíduos ou unidades adotantes. Assim, no contexto desta dissertação, a intenção é a analisar os fatores relacionados à aceitação ou rejeição de plataformas móveis por parte de engenheiros de *software*, inseridos em um ecossistema de *software* móvel.

No o intuito de responder a essa questão, Rogers se esforça em mensurar o índice de adoção de inovações a partir da análise de seus atributos. Segundo o autor, o índice de adoção é a velocidade com que uma inovação é adotada pelos membros de um sistema social. Os atributos da inovação, por sua vez, explicam os fatores que podem influenciar na sua aceitação ou rejeição.

2.1.2. PILARES

De acordo com o autor, *"A difusão é o processo pelo qual uma inovação é comunicada através de certos canais ao longo do tempo para os membros de um*

sistema social." Quatro elementos suportar o modelo: a inovação, os canais de comunicação, o tempo e o sistema social.

De acordo com Rogers, a inovação é definida por ideias, práticas ou objetos que sejam percebidos como novo pelo indivíduo, mesmo que para outros em algum outro lugar esta "inovação" já é ultrapassada. Se uma ideia parece nova para o indivíduo, é uma inovação.

Rogers diz que nem todas as inovações são adotadas na mesma velocidade, algumas são adotadas rapidamente, por exemplo, 71% dos americanos adotaram a Internet no período de 1989 a 2002, porém outras inovações podem apresentar um índice menor que 20%. Assim, Rogers levanta o seguinte questionamento: "Quais características da inovação afetam o índice com que elas são adotadas?".

Em relação aos canais de comunicação, Rogers define o meio pelo qual uma inovação é passada de um indivíduo para outro. Ele pode ser o meio de comunicação de massa, que são geralmente mais rápidos e eficientes para mostrar a potenciais adotantes que determinada inovação existe, ou comunicação pessoal, que são mais eficazes em persuadir o indivíduo a aceitar a nova ideia. Além disso, Rogers afirma que a difusão da inovação ocorre mais frequentemente entre indivíduos com objetivos e interesses diferentes. Enquanto a maior adoção ocorre entre indivíduos que compartilham os mesmos interesses, cultura, ou algo em comum que os une.

O tempo é um pilar fundamental na teoria da difusão de Rogers, e consiste de outros três pilares: o processo de tomada de decisão, aonde o indivíduo vai desde a primeira vez que é apresentado a adoção da inovação e uso contínuo, inovação, ou categoria de adotantes, e o tempo em que a inovação é adotada pelos membros do sistema social, determinando, assim, a taxa de adoção.

O processo de decisão é composto por cinco fases: conhecimento, persuasão, decisão, implementação e confirmação, como mostrado na Figura 2.

Figura 2: O processo de decisão de acordo com Rogers (2003).

O conhecimento é a fase em que o indivíduo é exposto pela primeira vez à inovação e não é motivado a buscar informações sobre essa inovação, porém começa a entender como ela funciona. Persuasão é a fase em que o indivíduo forma uma opinião favorável ou não à frente da inovação e começa a ser informar sobre a mesma. A decisão é a fase em que o indivíduo já sabe o potencial da inovação e executa um processo mental, no qual ele pesa os prós e os contras da inovação, e decide se quer ou não adotá-la. Rogers afirma que esta é a fase mais difícil da para coletar evidências empíricas por estar lidando com algo pessoal. Na fase de implementação o indivíduo está confiante na utilização de tal inovação e começa a implementá-la em sua rotina diária, certificando-se de que esta inovação é útil ou não, e muitas vezes ainda verifica se há novas informações sobre a mesma. Finalmente, no estágio da confirmação a pessoa toma a decisão final de continuar usando a inovação.

A capacidade de inovação, ou categoria de adotantes está relacionada com os diferentes momentos em que uma inovação é adotada por indivíduos que constituem o sistema social, o que permite classificá-las em cinco grupos: inovadores, adotantes iniciais, maioria inicial, maioria tardia e retardatários, aonde os inovadores são os primeiros e os retardatários são os últimos. A taxa de adoção é a velocidade relativa com que uma inovação é adotada por membros de um sistema social, por isso, está intimamente ligada à capacidade de inovação, ou categoria de adotantes. A maioria das curvas de adoção de inovações apresenta-se em forma de S *(Bell curve)* e é configurada com poucos inovadores , seguido por um aumento no número de adeptos (adotantes inicias) até que o valor da adoção pode chegar ao topo na fase madura (maioria inicial), logo após ocorre o processo de redução (maioria tardia) e, finalmente, a fase de maior declínio (retardatários), ilustrada na Figura 3.

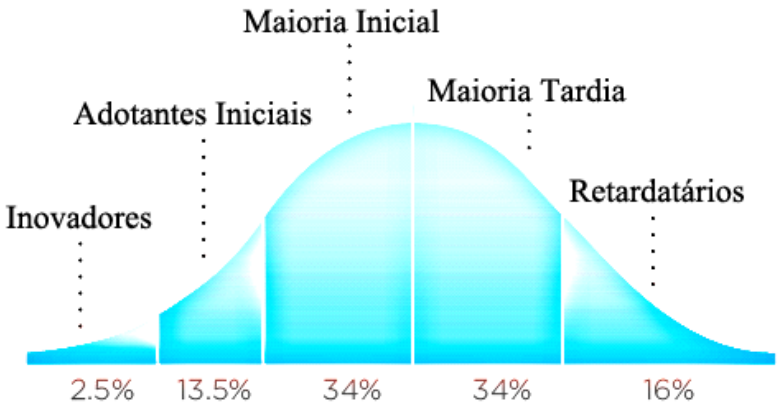

Figura 3: Velocidade com que uma inovação é adotada. Rogers (2003)

E, finalmente, o sistema social é definido como um grupo de indivíduos, organizações ou subsistemas, em outras palavras, unidades adotantes, dentro do qual ocorre a comunicação da inovação (difusão) e estão unidos para resolver problemas comuns a todos.

2.1.3. ATRIBUTOS DA INOVAÇÃO

O modelo proposto por Rogers define a inovação a partir de cinco atributos: vantagem relativa, compatibilidade, complexidade, testabilidade e observabilidade. Tais atributos são utilizados para explicar o processo de adoção, e segundo Rogers, os atributos da inovação nos ajudam a explicar a variação da taxa de adoção em até 87%, segundo vários estudos.

- A vantagem relativa é o grau com que a inovação é percebida como melhor que a ideia antecedente. O grau de vantagem relativa pode ser medido em relação a vários fatores, por exemplo, prestígio social, economia de tempo ou dinheiro, etc. Assim, a vantagem relativa de uma inovação está fortemente relacionada ao seu índice de adoção.

- A compatibilidade é o grau com que a inovação é percebida como compatível com as atividades diárias do indivíduo, experiências passadas ou suas necessidades. Segundo Rogers, uma inovação deve ser cuidadosamente implantada sem ir de encontro aos valores e crenças, ou

seja, sem modificar valores culturais e éticos. Mesmo que a inovação seja percebida como positiva, a cultura da organização pode se mostrar mais forte, podendo levar à resistência em adotá-la. Assim, a compatibilidade de uma inovação está fortemente relacionada ao seu índice de adoção.

- A complexidade ou facilidade de uso é o grau que uma inovação é percebida como difícil de entender ou ser utilizada, portanto, quanto mais simples, mais fácil à inovação tende a ser adotada. Este atributo está relacionado negativamente ao seu índice de adoção, quanto mais difícil, mais resistência a inovação vai encontrar para ser utilizada.

- A testabilidade é o grau com que a inovação pode ser experimentada antes de adotá-la ou rejeitá-la, por exemplo, amostra grátis, *test-drive*, versão demo de um *software*. Rogers afirma que, inovações que podem ser facilmente experimentadas são geralmente adotadas mais rapidamente. Isto pode ajudar a minimizar possíveis dúvidas sobre a inovação e permitir que a inovação seja adotada mais rapidamente. Logo, a testabilidade está fortemente relacionada ao seu índice de adoção.

- Observabilidade é o grau com que os benefícios de uma inovação são visíveis a outros indivíduos. Este atributo está fortemente relacionado ao seu índice de adoção, pois quanto mais visíveis forem os resultados de uma inovação, mais rápido se dará o processo de sua adoção.

Resumindo, as inovações que são percebidas com maior grau de vantagem relativa, compatibilidade, testabilidade e observação e menor complexidade serão aprovadas mais rapidamente do que outras inovações. Com o intuito de reforçar a relevância do que está sendo dito, abaixo são apresentados dois trabalhos que utilizam o conceito de inovação de Rogers no contexto computacional. O primeiro trata-se de uma análise sobre a adoção de sistemas de informação na área da saúde. O segundo é um

estudo que reflete a disseminação de inovações quanto ao desenvolvimento de software no *website* Stack Overflow[2].

Perez e colegas (2010) buscam compreender quais os fatores principais percebidos pelos usuários que contribuem para a adoção de sistemas de informações na área da saúde. A pesquisa procurou características do contexto social interno que mais contribuem para a adoção de inovações baseadas em tecnologia da informação no setor da saúde e como os diversos grupos internos atuam durante o processo de adoção de inovações. Para este fim, realizaram uma pesquisa de natureza qualitativa exploratória efetuada por meio de entrevistas e questionários com os principais gestores e usuários-chave que influenciam a adoção da inovação. No total, oito entrevistas foram realizadas em duas instituições de saúde. Assim, identificaram fatores que contribuem para a adoção de uma inovação tecnológica na área da saúde e como os vários grupos internos atuam na adoção (e uso) ou rejeição de uma inovação tecnológica.

Outro trabalho recentemente publicado, que também utiliza o conceito de inovação de Rogers é apresentado por Gómez e colegas (2013). Nesta pesquisa, os autores investigaram o compartilhamento de links no *website* Stack Overflow - que é um *website* focado em perguntas e respostas para engenheiros de *software* - para determinar se os engenheiros de software disseminam inovações quanto ao desenvolvimento de software com outros desenvolvedores por meio do uso de *URLs (Uniform Resource Locator*). Assim, investigaram quais tipos de *links* engenheiros de software compartilham e se estão relacionados com desenvolvimento de *software* (documentações oficiais, *posts* em blogs, links para wikis, *posts* em fóruns de discussão, etc). Além disso, qual a frequência com que isto ocorre, e quais *websites* (ex. mysql.com, mozila.com, wikipedia.com, jquery.com, google.com, stackoverflow.com, etc) são mais referenciados nos *links* do Stack Overflow. Como conclusão, observaram que o compartilhamento de *links* é uma atividade importante para o ecossistema em torno do Stack Overflow. O estudo mostra que uma proporção significativa de links compartilhados trata-se do desenvolvimento de software, por exemplo, novas bibliotecas e *frameworks*. Além disso, o Stack Overflow apareceu em primeiro lugar

[2] http://stackoverflow.com/

quanto aos *websites* mais referenciados, talvez pelo compartilhamento de soluções já existentes na comunidade. Os resultados também mostram que Stack Overflow é apenas uma parte de um ecossistema muito maior de compartilhamento de recursos *online*. Assim, novas pesquisas na área são de grande importância para a compreensão deste cenário.

2.2. ECOSSISTEMAS DE SOFTWARE

2.2.1. VISÃO GERAL

Nos primórdios do desenvolvimento de *software*, um produto de *software* era o resultado de um esforço de um fabricante independente, trabalhando como uma unidade isolada, sem contar com outros fabricantes para obter sucesso, porém, com o avanço da globalização dos mercados, a competição ficou mais disputada levando a uma mudança de paradigma, em que fabricantes passam a depender fortemente de componentes de *software* e infraestrutura de terceiros para que seus *softwares* possam ser produzidos (Cusumano, 2004). Assim, as empresas vendedoras de *software* evoluíram de unidades independentes para uma rede interdependente de fornecedores de serviços e *software*. Devido a este cenário, a integração de produtos de software e serviços vem demandando cada vez mais atenção de técnicos e gestores de empresas que desenvolvem e usam tecnologias. Portanto, esta nova forma de relacionamento entre empresas de desenvolvimento de *software* e empresas que produzem componentes e infraestrutura, moldou o cenário atual, criando o novo modelo de negócio chamado ecossistemas de *software.*

2.2.2. ORIGEM

O termo ecossistemas de *software* é baseado no termo de ecossistemas de negócio que se baseia no termo ecossistemas biológicos. A definição de ecossistemas de *software* é estabelecida e melhor compreendida ao se conhecer as suas origens.

Os ecossistemas biológicos, de onde tudo surgiu, oferecem metáforas úteis para ajudar a entender o que são ecossistemas de *software*. Iansiti e Levien (2004) utilizam o exemplo do jaguar ou, como conhecemos no Brasil, a onça-pintada. Este animal é conhecido por se alimentar de quase 85 espécies de animais, ajudando no controle de populações dentro do ecossistema em que ele está inserido, mas ele apenas é uma pequena parte que forma esse todo. Essa metáfora pode ser utilizada para entender o papel de uma governante de plataforma, como a Google, dona do Android, que apesar de ser uma gigante do mercado tecnológico, é apenas uma parte do ecossistema de *software* móveis. Dhungana *et al.* (2010) ainda ressaltam que existem outras metáforas que podem ser estabelecidas entre ecossistemas de *software* e ecossistemas biológicos, como: ambos os ecossistemas tem uma quantidade finita de recursos e em ambos, a colaboração e a competição são elementos fundamentais para que os mesmos prosperem.

As comparações podem ser feitas, mas até um determinado ponto, visto que o estudo desses dois ecossistemas é influenciado por fatores diferentes. O estudo de ecossistemas biológicos está focado em determinar fatores externos que influenciem em o ecossistema e suas dinâmicas, enquanto que ecossistemas de *software* têm como principais estudos, quais fatores de sucesso e crescimento influenciam em sua dinâmica. E o maior fator que diferenciam ambos, é que em ecossistemas de *software* existe o fator humano, e por se tratar de seres humanos, os seus participantes podem conscientemente decidir se vão continuar naquele ecossistema ou se vão sair, e podem até mesmo, destruí-lo.

Saindo da visão natural de ecossistemas, têm-se os ecossistemas de negócio, que são definidos por Moore (1993) como uma comunidade econômica apoiada por uma fundação: organizações e indivíduos interagindo como organismos do mundo de negócios. Os membros deste mundo incluem: fornecedores, produtores, competidores, entre outros, esses papéis são semelhantes aos papéis presentes em ecossistemas de *software*, que serão expostos e explicados mais a frente. Moore (1993) também diz que durante o tempo esses membros aprendem a evoluir em conjunto e tendem a alinhar-se

em torno de uma ou mais empresas centrais, que no caso de ecossistemas de *software* são chamadas são chamadas *platform leaders* ou governantes (Gawer e Cusumano, 2002).

Ecossistema de *software* trata-se de um termo utilizado para a crescente colaboração entre unidades, por exemplo, empresas, engenheiros que desenvolvem soluções de sistemas de informação, clientes, entidades governamentais, entre outros que fazem parte deste contexto. Segundo Jansen (2009), "ecossistema de *software* é como um conjunto de negócios, empresas ou entidades que funcionam como uma unidade e interagem com um mercado compartilhado para fornecer *software* e serviços, levando em consideração o relacionamento entre eles". Outra definição é apresentada por Bosch e Petra (2010), "ecossistema de *software* é um conjunto de soluções de *softwares* e uma comunidade especialista no domínio em serviços para a comunidade de usuários, que compõe soluções relevantes para as suas necessidades". Ecossistemas de *software* geralmente são governados e dirigidos por uma ou mais partes que lucram quando o ecossistema prospera. Geralmente esses governantes controlam a tecnologia que serve como base para o ecossistema.

Têm-se a Apple, como exemplo, de uma empresa que possui um ecossistema, ou seja, uma empresa que se pode chamar de governante. O seu ecossistema gira em torno de seu sistemas operacionais, o iOS e o OS X, que possui uma loja de aplicativos, chamada App Store, com ela, a Apple oferece um meio para que os fornecedores de *software* para sua plataforma possam vender seus produtos de *software* aos usuários e, também, fornece suporte aos seus usuários, que usufruem da estrutura provida pela mesma e podem comprar soluções e, assim, fomentar o mercado de vendas de *softwares* dentro do ecossistema.

Os governantes de um ecossistema de *software* precisam focar em três diferentes perspectivas para coordenar e governar: o nível de ecossistema, o nível da cadeia de abastecimento e o nível de fabricantes de *software*. No nível de ecossistema, escolhas estratégicas devem ser feitas pelo governante para guiar como o fabricante de *software* deve se comportar dentro do ecossistema para maximizar seus lucros e compartilhar

informações. Governantes de ecossistemas detêm o controle de como esse ecossistema deve se comportar, podendo desenvolver estratégias para auxiliar esses fabricantes a atingir a meta de lucrar. No nível de cadeia de abastecimento, governantes determinam seu público alvo e quais serão seus fornecedores de tecnologia, além de também definir e estabelecer parcerias de cooperação. No nível de fabricantes de *software*, os governantes estabelecem quais os efeitos que aquele ecossistema deverá ter em seus produtos de *software*. Jansen (2009) sugere que neste nível, diretrizes quanto a reuso tecnologia e quanto a desenvolver do zero ou comprar tecnologias têm que ser estabelecidas, para que os fabricantes possam usufruir de todo o potencial do ecossistema em que estão inseridos e usar o conhecimento que já existe no mesmo para uso interno em sua companhia.

Este novo modelo de mercado, que envolve desenvolvimento de *software* e relações entre diversos fabricantes, introduz algumas problemáticas novas que precisam ser estudadas: dado que os fabricantes de software funcionavam como unidades separadas e agora precisam de outros fabricantes para desenvolver seu produto de *software*, um grande desafio é até onde se deve abrir a interface de seus produtos para que terceiros possam usar, ou abrir, até mesmo, o seu *software*. Esses fabricantes devem pensar o quão aberto será seu produto, o quanto do seu código pode ser visto por terceiros e o quanto de informação será trocada com outros membros do ecossistema.

Outra problemática está presente na hora de modelar um ecossistema de *software*, as suas características formais devem ser estabelecidas. Tem-se como exemplos de características formais: presença de padrões, diferentes papéis que os fabricantes de *software* podem assumir dentro do ecossistema, como esse ecossistema vai se manter e prosperar. Esses desafios ainda precisam ser melhor estudados, visto que ainda não foram encontrados na literatura formalismos que possam resolver este problema (Brinkkemper, 2007).

Outro desafio que surge com este novo modelo: determinar uma estratégia para lucrar e prosperar em um ecossistema de *software*. Várias dúvidas surgem sobre qual a melhor estratégia a seguir, por exemplo, o melhor modelo de negócio é *open source* ou

código fechado? Qual o público alvo? É melhor reutilizar código ou construir código do zero? Os fabricantes precisam escolher um modelo de negócio que forneça soluções para estas diversas dúvidas (Jansen, 2007).

2.2.3. PAPÉIS DENTRO DE UM ECOSSISTEMA

Utilizando as definições de Jansen (2009) e Bosch e Petra (2010), que foram previamente explanadas, um ecossistema de *software* é formado por um conjunto de atores, que desempenham papéis dentro do mesmo. Na Figura 4 é ilustrado um exemplo de ecossistema de *software* real, nesse caso mostra-se o exemplo de um ecossistema móvel. Na referida figura tem-se a Apple e os demais atores que atuam em seu ecossistema móvel, o iOS. Existem cinco papéis chave nesse ecossistema, a Apple, as operadoras de telefonia móvel, os desenvolvedores de aplicações, os usuários e os fabricantes de *hardware*.

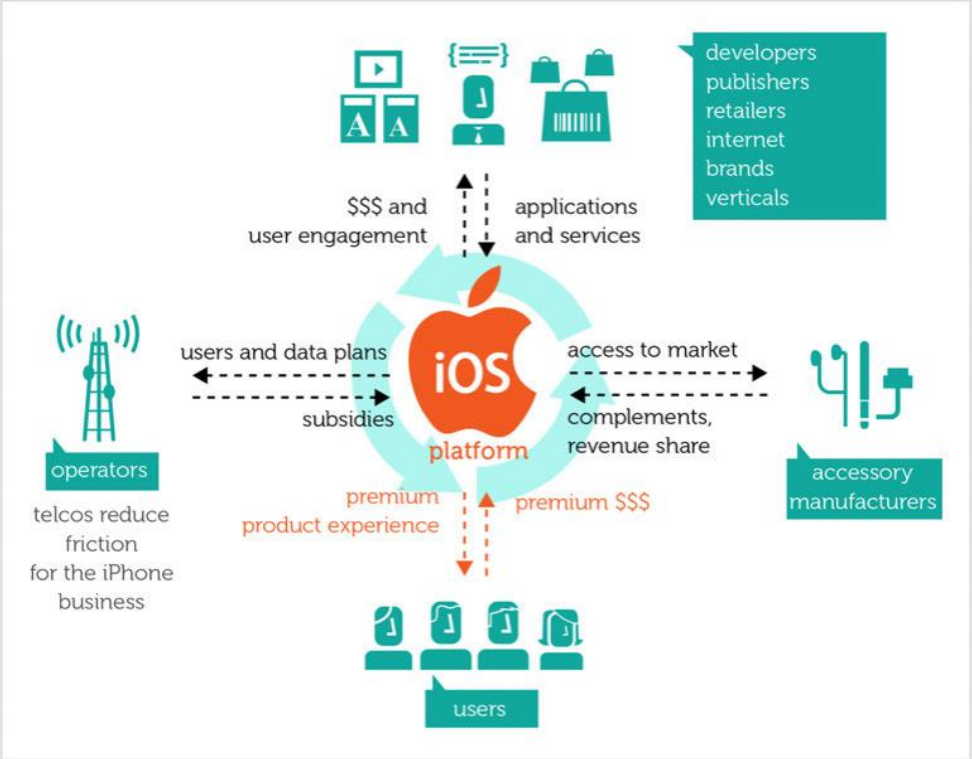

Figura 4: Atores e seus papéis em um ecossistema de *software* móvel, neste caso o iOS. (VisionMobile, 2013)

Com mais detalhes, irão ser apresentado agora todos os atores e quais as relações que ocorrem entre eles.

- **Apple**: é a dona da plataforma ou, como é denominada neste trabalho, a governante. Ela desenvolve é a dona da plataforma iOS, que é um sistema operacional para *smartphones*. É em torno desta plataforma que o ecossistema de *software* gira. Todos os atores se relacionam de alguma maneira com a plataforma iOS e, consequentemente, com a Apple.

- **Usuários**: são os usuários finais da plataforma, as pessoas que utilizam o *smartphone* com o iOS instalado, e também fazem uso das aplicações que são desenvolvidas pelo desenvolvedores de *software*, além de possuírem um contrato com as empresas de telefonia móvel. As aplicações desenvolvidas estão presentes em uma loja virtual, chamada App Store. Os usuários pagam para usufruir desta plataforma, seja apenas comprando o *smartphone*, seja também comprando produtos de *software* presentes na loja virtual, além de estarem pagando por *hardware* já embutido no *smartphone* comprado e pagando por planos de dados junto às operadoras de telefonia móvel.

- **Desenvolvedores**: são os engenheiros de *software* que desenvolvem aplicações/serviços para a plataforma. Eles criam produtos de *software* que são utilizados pelos usuários da plataforma, e podem disponibilizá-los de forma gratuita ou paga. O seu lucro é oriundo de usuários que pagam pelas aplicações que são desenvolvidas por eles. Os mesmos recebem da Apple documentações e tecnologia para que possam desenvolver suas aplicações e pagam uma taxa para a Apple para estarem aptos a publicar seus produtos em sua loja virtual.

- **Fabricantes de *hardware*:** fornecem componentes físicos que serão acoplados ao *smartphone* da Apple. São fabricantes que vão desde produtores de *chips* que são utilizados diretamente na fabricação e montagem dos *smartphones*, até fabricantes de acessórios, como fones de ouvido, capas protetoras, etc, que podem ser comprados separadamente. Eles prosperam quando o *smartphone*, que possui o sistema operacional iOS, é utilizado por muitos usuários, pois aumenta a demanda por seus produtos.

- **Operadoras de telefonia móvel**: este é um ator exclusivo deste tipo de ecossistema de *software*. Uma vez que esses *smartphones* são evoluções dos telefones celulares comuns, ou *dumbphones*, o usuário precisa estar atrelado a uma operadora de telefonia móvel. Ele precisa de um plano de dados para utilizar todo o potencial de seu *smartphone*, que inclui acesso à internet. Então as operadoras tem uma nova fonte de receita, planos de dados específicos para *smartphones*, assim os usuários podem acessar a loja virtual de aplicações, podem comprar os produtos de *software* e, com isso, gerar receita para os desenvolvedores e para a empresa governante.

Com a explicação de cada papel de cada ator dentro de um ecossistema de *software* móvel, é possível notar que todos estão interligados de alguma forma e precisam cooperar para que o ecossistema de *software* prospere, caracterizando o cenário de ecossistema de *software* e exemplificando o que os conceitos que Jansen (2009) e Bosch e Petra (2010) definem muito bem.

Uma vez que foi apresentado um exemplo de ecossistema de *software* e seus atores, nota-se como eles realmente funcionam como uma comunidade de atores que precisam se relacionar e trocar conhecimento e informações para que todos possam prosperar em torno da plataforma central do ecossistema.

2.2.4. ECOSSISTEMAS ABERTOS E FECHADOS

Os termos aberto ou fechado são percebidos como um conceito binário: verdadeiro ou falso. Esses termos são comumente utilizados pelos fabricantes de *software* para categorizar seus produtos e/ou sua organização. Organizações podem se utilizar do termo aberto para se categorizar buscando o interesse de um determinado segmento de desenvolvedores de *software*. Molder e colegas (2011) buscam em seu trabalho responder a uma pergunta: existem ecossistemas totalmente fechados ou totalmente abertos? A resposta direta é: não. Nenhum ecossistema de *software* é verdadeiramente aberto ou fechado, eles possuem um grau que determina qual seu nível de abertura. Até a data do referido estudo, não existia quase nenhuma discussão na literatura que esclarecesse o que é realmente um ecossistema de *software* fechado/aberto (Hylén, 2006).

Sistemas fechados e abertos são estudados desde os anos 1950 nas ciências físicas, mais precisamente na área da termodinâmica. Em sistemas termodinâmicos, algo aberto é caracterizado quando ocorre troca de calor e trabalho com o ambiente, já um sistema fechado é o exato oposto (von Bertalanffy, 1950). Boulding (1956), em seu estudo sobre a teoria geral dos sistemas, enfatiza que organizações são sistemas abertos, pois sofrem influências de ambientes externos em suas dinâmicas internas e vice-versa. Na teoria apresentada por Boulding, um sistema fechado também é o exato oposto de um sistema aberto, pois considera que apenas sofre influências de fatores internos da organização. A Figura 5 ilustra um sistema aberto e um sistema fechado, de acordo com Boulding.

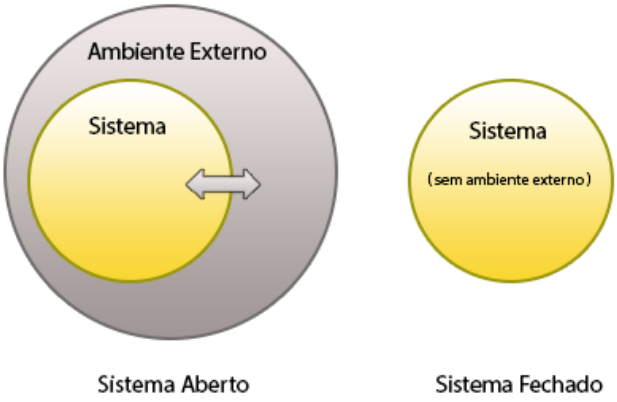

Figura 5: Ilustração de um sistema aberto e um fechado.

Molder cita o trabalho de Maxwell (2006), que sugere que algo não é simplesmente aberto ou fechado, mas sim vai de aberto para fechado, através de níveis. Existe uma escala, aonde aberto e fechado são os pontos extremos. Algo pode ser colocado nessa escala, mas nunca pode ser o ponto extremo (completamente aberto ou completamente fechado).

Sistemas abertos são muito populares na comunidade de código-fonte aberto, mas não pode ser algo que se deva acreditar cegamente. Quando se trata de código aberto, o próprio nome diz: é um sistema que possui o seu código-fonte aberto e apenas isso, não quer dizer que a organização por trás do sistema também possui seus processos organizacionais abertos e transparentes. Código-fonte é apenas uma parte que compõe o todo. Então usando um exemplo real: o sistema operacional móvel da Google, o Android é de código-fonte aberto, já o iOS da Apple é fechado. Isso quer dizer apenas que o iOS tem um alto nível de isolamento quando se refere à código-fonte, mas nada além disso, uma vez que outros aspectos do sistema operacional podem ser abertos e o inverso também pode ser verdadeiro na Google, aonde ela abre o código-fonte de seu sistema operacional, mas não torna transparente seus processos, práticas e estratégias organizacionais.

Através de seu levantamento da literatura e a sua análise, Molder cria um modelo chamado *Clopenness Assessment Model* (CAM), que serve para avaliar o grau de abertura de um sistema. Ele realizou quatro estudos de caso para avaliar o seu modelo. Ele chega à conclusão que seu modelo não é guia definitivo para avaliar o nível de abertura de um sistema, pois alguns fatores externos, mais relacionados ao âmbito social, que devido a sua natureza não podem ser controlados, como: confiança, franqueza e honestidade, podem influenciar no resultado final.

Sistemas, *softwares* ou organizações completamente fechadas não existem. A discussão que leva em consideração os exemplos previamente citados como totalmente fechados ou abertos são falsas dicotomias, que é a divisão de um elemento em duas partes exatamente opostas.

2.2.5. AS TRÊS PERSPECTIVAS

Ecossistemas de *software* podem ser analisados utilizando três dimensões: técnica, negócio e social. Em um estudo realizado por Barbosa e colegas (2011) essas três dimensões são propostas e analisadas, através de uma revisão sistemática da literatura existente sobre ecossistemas de *software* até o momento da pesquisa. Barbosa justifica a utilização dessa abordagem através do argumento de que é importante entender como os ecossistemas podem ser criados e evoluídos e, dividir a análise em três dimensões facilita o entendimento.

A dimensão técnica é focada na plataforma na qual o ecossistema gira em torno, em aspectos como: o mercado, a tecnologia, infraestrutura, entre outros. Nesta dimensão, o foco está em selecionar uma plataforma para analisar, para entender como foi projetada e desenvolvida, como foi o seu processo de abertura, considerando a sua arquitetura, seu nível de componentização e transparência durante sua evolução e manutenção (Santos e Werner, 2011a). Esta dimensão também examina aspectos inerentes ao processo de tomada de decisão e propriedades da manutenção da arquitetura da plataforma.

Existe também outro aspecto importante que é analisado nesta dimensão, e está relacionado ao processo de abertura da arquitetura da plataforma. Inicialmente, níveis devem ser identificados para que fique mais fácil separar os componentes e módulos da plataforma, explorando os benefícios de uma arquitetura baseada em camadas, tornando todo este processo de abertura mais simples, pois pode ser dividido em tarefas e subtarefas designados a grupos de trabalho.

A dimensão de negócio está focada em analisar o contexto da plataforma, definir seu escopo, quais os papéis e suas características. Isso pode ser feito utilizando algum tipo de métrica, como a Goal-Question-Metric (GQM), proposta por Basili e colegas (1999), como por exemplo:

1. Selecionar os objetivos do ecossistema.
2. Elaborar perguntas para melhor entender estes objetivos.
3. Definir, coletar e analisar métricas, como por exemplo, números de desenvolvedores, países e usuários que fazem parte do ecossistema.

Essa abordagem permite coletar, manipular e apresentar indicadores de sustentabilidade e diversidade e transformar em informações que indiquem o quão saudável é certp ecossistema (Dhungana *et al.*, 2010). A partir destes indicadores um *framework* pra monitoração e gestão do ecossistema pode ser modelado e utilizado (Santos e Werner, 2011b).

A dimensão de negócio requer contextualização com outros tipos de ecossistemas (biológicos e de negócio), pois precisa mapear conceitos, atributos e comportamentos, que estão presentes de forma mais madura nesses outros tipos de ecossistemas. Este cenário impacta diretamente no processo de tomada de decisão por parte dos membros da indústria de *software*, pois não precisa apenas de entendimento de investimentos e custos, mas também de benefícios, riscos e oportunidades, que compõe uma evolução coletiva, inovadora e baseada em valores (Biffl e colegas, 2006).

Têm-se outros aspectos importantes para serem analisados através da dimensão de negócios: sustentabilidade e diversidade do ecossistema. Considerando a sustentabilidade, o elemento chave é a empresa governante. Ela precisa sempre estar atenta a sua base de desenvolvedores e usuários, sempre checando se essas bases podem ser mantidas ou aumentadas levando em consideração longos períodos de tempo, aonde mudanças irão ocorrer, como novas tecnologias emergentes ou novos produtos, procurando analisar como essas mudanças podem afetar seus usuários e desenvolvedores, se sua plataforma pode sobreviver a essas mudanças e até mesmo sobreviver a ataques ou sabotagens oriundos de outras plataformas. Pensando no aspecto de diversidade, a empresa governante e os membros influentes no ecossistema, precisam procurar trazer inovações e elementos que incentivem o ecossistema a mudar, se adaptar e não estagnar. Ele precisa continuar interessante. Novas linguagens de programação ou dispositivos de *hardware* podem ser introduzidos no ecossistema, e tendo sua utilização incentivada por membros influentes do ecossistema, afim de que o objetivo de manter o interesse seja alcançado.

Barbosa também apresenta a dimensão social, a terceira e última analisada por ele em seu estudo. Ele relata que uma das maiores implicações para o seu trabalho é analisar esta dimensão. Esta dimensão possui uma grande importância quando se olha para fatores que levam ao sucesso de um determinado ecossistema. Campbell e Ahmed (2010), em um estudo semelhante ao de Barbosa, apontam que empresas governantes que se comprometem mais com o fator social, tendem a possuir um ecossistema de sucesso.

Messerschmitt e Szyperki (2003) dizem que o próprio desenvolvimento de *software* é uma atividade social, uma vez que *software* precisam satisfazer diversos requisitos de usuários. Então, identificar e analisar requisitos faz com que desenvolvedores pensem mais como sociólogos e psicólogos (cursos das ciências humanas) do que como profissionais formadas em ciências exatas. E mais, neste mesmo estudo, os autores levantam outro ponto: o conceito de aplicações sócio-técnicas. Este tipo de aplicação combina grupos de pessoas trabalhando de forma coletiva, onde a tecnologia da informação é apenas uma parte que compõe o todo, que

são organizações inteiras. Com isso, a tecnologia da informação não pode ser interpretada como um aspecto isolado, mas sim como parte de um ambiente sócio-técnico, em que aspectos sociais devem ser levados em consideração antes de usar, contratar ou criar um novo produto de *software*.

Os conceitos e exemplos apresentados nestas três dimensões de ecossistemas de *software* serão utilizados quando os resultados deste trabalho forem expostos, explicados e discutidos mais a frente em capítulos desta dissertação.

Neste capítulo foram apresentados os principais conceitos sobre o novo modelo de desenvolvimento e negócio: ecossistemas de *software*. Como foi dito, este é um conceito relativamente novo, que se originou de conceitos oriundos de outros tipos de ecossistemas (biológicos e de negócio). Ainda existem outros tipos de desafios que os atores presentes dentro de um ecossistema de *software* precisam enfrentar, mas ainda pouco se sabe até o momento sobre quais seriam esses desafios, então se optou por citar os mais relevantes.

3. METODOLOGIA DE PESQUISA

3.1. Visão Geral

Parte deste trabalho foi desenvolvido em parceria com outro aluno de mestrado, Müller Miranda. Os resultados, que serão apresentados no Capítulo 4, foram publicados em Miranda *et al.* (2014) e em Ferreira *et al.* (2015).

A pesquisa realizada neste trabalho é caracterizada como exploratória, pois não se tinha conhecimento prévio sobre a problemática de pesquisa. Portanto, para que o objetivo deste trabalho - entender os fatores que influenciam um engenheiro de *software* a escolher um determinado ecossistema de *software* (ver Seção 1.3) - pudesse ser alcançado, foi escolhida uma abordagem qualitativa, pois a mesma baseia-se nas observações das pessoas que realizam a atividade que é o objeto de estudo (Strauss e Corbin, 2008), e optou-se por utilizar entrevistas semiestruturadas como método para de coleta de dados.

O método qualitativo é justificado por ser mais eficaz em casos onde o estudo é exploratório, sem teorias definidas, teorias estas que poderiam ser testadas em um experimento controlado. Entrevistas semiestruturadas (Singer *et al.*, 2008) foi o método escolhido para a coleta de dados, porque com elas é possível uma maior liberdade no momento da entrevista, sem a necessidade de seguir exclusivamente um roteiro, podendo explorar outras perguntas/respostas, da forma que os entrevistadores acharem melhor.

Após a realização da coleta de dados, neste caso, as entrevistas – que foram realizadas em 2 etapas (Junho de 2013 e Agosto de 2014) - foram feitas as transcrições das mesmas a fim de serem analisadas utilizando técnicas da Teoria Fundamentada em Dados (Strauss e Corbin, 2008). Além disso, a realização do preenchimento do

debriefing (Fanning e Gaba, 2007) para facilitar a discussão do grupo para a construção conjunta do conhecimento.

Técnicas da Teoria Fundamentada em Dados (Strauss e Corbin, 2008) foram utilizadas, pois no final do processo, que consiste em alternar etapas de coleta e análise de dados para que os dados da etapa anterior guiem o pesquisador para uma nova etapa de coleta e análise de dados, uma teoria baseada nos dados coletados e analisados, pode ser formada e descrita. Cada uma destas técnicas – pesquisa qualitativa, entrevistas, técnicas da teoria fundamentada em dados e o documento de *debriefing* - serão expostas na seção 3.5.

3.2. Pesquisa Qualitativa

Diferente dos estudos quantitativos, a pesquisa qualitativa não segue a rigor um plano previamente estabelecido, mas sim é direcionada ao longo do seu desenvolvimento. Além disso, não é seu objetivo enumerar ou medir eventos, seu foco de interesse é amplo e parte de uma perspectiva diferenciada da adotada pelos métodos quantitativos.

Nas pesquisas qualitativas, é comum que o pesquisador procure entender os fenômenos, a partir da visão que tem os participantes da situação estudada, e com isto, situe sua interpretação destes fenômenos. Seaman (2008) diz que a principal vantagem de se realizar uma pesquisa qualitativa reside no fato de que métodos qualitativos obrigam os pesquisadores a mergulhar nos fatos observados, sem abstraí-los. Tendo isso em mente, os resultados tendem a ser mais ricos em detalhes e informativos (Ferreira *et al.*, 2015).

Segundo Strauss e Corbin (2008), há três componentes principais nesse tipo de pesquisa: dados, procedimentos e relatórios. Os *dados* são oriundos de entrevistas, gravações de áudio ou vídeo, documentos, relatórios, etc, ou seja, a partir dos procedimentos de coleta de dados. Portanto, o dado é a matéria prima que será estudada a partir de *procedimentos* que possuem o objetivo de reduzir esses dados de forma a

relacionar conceitos existentes entre eles. Finalmente, os *relatórios* para que seja possível apresentar os resultados (Miranda *et al.*, 2014).

Algumas características de pesquisa qualitativa incluem: (1) o ambiente natural como fonte direta de dados e o pesquisador como instrumento fundamental da pesquisa, (2) o caráter descritivo, (3) o significado que as pessoas dão às coisas e à sua vida como preocupação do investigador e por fim (4) o enfoque indutivo (Neves, 1996).

3.3. Métodos de coleta de dados

Um método para coleta de dados é o ato de pesquisar, organizar e juntar documentos, provas e gravações; é buscar informações sobre o tema em estudo e agrupá-las de forma a facilitar uma posterior análise. Existem vários tipos de métodos de coleta de dados, tais como entrevistas, questionários, formulários, observações etc. Para este trabalho entrevistas semiestruturadas foram conduzidas. Estas são entrevistas que não possuem um roteiro que deve ser estritamente seguido, dando liberdade assim ao entrevistador para se aprofundar em tópicos da entrevista conforme ele achar necessário - como método de coleta de dados. O método possui algumas desvantagens: a falta de compreensão do entrevistado quanto o significado das perguntas; o fornecimento de respostas falsas; a influência da presença do pesquisador; etc. De forma a mitigar tais desvantagens, buscou-se conhecer previamente os entrevistados e manter-se o mais imparcial possível, procurando não emitir opiniões, insegurança ou admiração. Além disso, o planejamento prévio da entrevista - no caso deste trabalho, este planejamento foi feito estudando a Teoria de Difusão da Inovação (Rogers, 2003) e elaborando esboços de entrevistas, até chegar o guia final – ajuda a manter o foco no objetivo da mesma.

Os tipos de entrevistas vão de estruturadas a não estruturadas. O que varia entre os tipos de entrevistas é o nível de controle por parte do pesquisador (Seaman, 2008). No caso de entrevistas estruturadas, o pesquisador tem o foco bem estruturado e definido, deixando suas perguntas bem específicas, pois ele sabe exatamente o que está procurando. As entrevistas não estruturadas são exatamente o

oposto disso, elas são amplas e procuram coletar o máximo de informações possíveis para o tópico de estudo, pois o pesquisador conhece pouco sobre o assunto de estudo.

Para este estudo optou-se pelo método de entrevistas semiestruturadas (Apêndice 1), pois elas são a interseção entre os dois tipos de entrevistas citados anteriormente. Entrevistas semiestruturadas possuem uma variedade de perguntas, específicas ou abertas, permitindo assim que o pesquisador não só colete informações sobre pontos já conhecidos, mas também tenha a oportunidade de levantar informações sobre aspectos inesperados, ou seja, informações que não poderiam ser coletadas seguindo um roteiro restrito (Dewalt e Dewalt, 2009). No fundo, uma entrevista semiestruturada se parece muito com uma conversação, mas se diferencia por ter o objetivo básico de coletar dados. Além disso, uma entrevista semiestruturada considera a existência de um guia que consiste de uma lista de perguntas para garantir que todos os tópicos de interesse serão abordados. Este roteiro de entrevistas foi elaborado com base na Teoria da Difusão da Inovação, pois é o *framework* conceitual deste trabalho. O guia de entrevistas deste estudo está no Apêndice 1. Para elaboração deste guia, o autor deste estudo teve o auxílio de seu orientador e de outro estudante de mestrado.

3.4. Método de análise de dados

Um método de análise de dados é a atividade de transformar um conjunto de dados coletado por meio de entrevistas, questionários, formulários, em uma análise racional.

Em um cenário de pesquisa qualitativa, como neste trabalho, os dados coletados são interpretados e não mensurados. Portanto, métodos análise de dados qualitativos devem ser utilizados, ou seja, a análise não deve ser baseada em análise de dados numéricos, mas deve ter como objetivo descobrir conceitos e relações nos dados brutos obtidos nas entrevistas e/ou questionários, organizando tais conceitos e relações em um esquema explanatório teórico (Strauss e Corbin, 2008).

Geralmente, o volume de dados gerados por pesquisas qualitativas é grande. Neste caso, os dados devem ser analisados através de um processo de redução, onde um

novo volume de dados mais compreensível é formado, utilizando para tanto o processo denominado de codificação (Singer *et al*., 2008). Este processo de redução está atrelado aos objetivos do estudo, objetivando a criação de um roteiro para categorização destes dados. Uma vez que estes dados estão categorizados, a análise torna-se possível, pois estes dados estão agora codificados em uma linguagem mais próxima dos objetivos do estudo. As técnicas de codificação utilizadas neste trabalho são baseadas na teoria fundamentada em dados (Strauss e Corbin, 2008), assunto a ser abordado na próxima seção.

3.5 A Teoria Fundamentada em Dados

A Teoria Fundamentada em Dados é um método de pesquisa qualitativa, focado no desenvolvimento de estruturas teóricas construídas a partir da análise de dados qualitativos. A ideia principal da teoria fundamentada em dados, é que o resultado final, a teoria em si, surja através dos dados recolhidos pelo pesquisador (Charmaz, 2006). O aspecto central da teoria fundamentada em dados está presente nas codificações, que é o processo de categorizar os dados obtidos nas entrevistas ou questionários. Esta categorização é importante para o pesquisador no que diz respeito ao fato de que ele minimiza o número de elementos que ele precisa considerar, agrupando conceitos em categorias.

A teoria fundamentada em dados ocorre através de fases: a codificação aberta, a codificação axial e a codificação seletiva. Na codificação aberta, os dados são analisados minuciosamente, linha por linha, para que as categorias possam ser criadas. A Figura 6, retirada do trabalho de Banks *et al.* (2000) - que analisa o uso de cartões postais, é um exemplo de codificação aberta. Nela é possível observar as (i) categorias numeradas e (ii) exemplos de trechos retirados dos cartões postais relacionados a estas categorias, seguidos do (iii) número de vezes que cada categoria foi utilizada mostra um exemplo de codificação aberta utilizada na análise do uso de cartões postais.

2. Description of especially enjoyable experience; great fun	"He continually pushes himself to the limit (and us) to have as much fun as he possibly can, a true '**fun hog**.' " [#50]	63
3. Reference to extraordinary work effort or work demands	"Was a working dynamo — couldn't keep enough jobs ahead of him!" [#120]	3
4. Relates connections to celebrity or drops name of celebrity	"Alas, Patty did not get mentioned in humorist Dave Barry's newspaper column this year …" [#98]	2
5. Mention of experience in cultural arts — fine arts museums, opera, ballet, symphony, etc.	"She is also showing quite an interest in ballet." [#51]	19

Figura 6: Exemplo de codificação aberta (Banks *et al.*, 2000)

Na etapa da codificação axial, ocorre um processo de criação de subcategorias, aonde as categorias principais criadas na codificação aberta são divididas. Categorias representam fenômenos alinhados com os objetivos do estudo, e as subcategorias são criadas para responder perguntas sobre estes fenômenos tais como: por que, quando, onde, como, quem e com quais consequências (Strauss e Corbin, 2008). A Figura 7, também retirada do trabalho de Banks *et al.* (2000) mostra o exemplo de subcategorias utilizadas na codificação axial. "*Positive Experience/Adventure*" é a categoria, e tudo que segue embaixo são as subcategorias encontradas e a frequência com que apareceram na análise, neste caso, por exemplo, a subcategoria "*exciting adventure*" apareceu 34 vezes. Somando todas as suas subcategorias, a categoria principal teve 278 aparições.

1 ***POSITIVE EXPERIENCE/ADVENTURE*** (278 tokens):
- 2) fun experience/great time/I-we enjoyed (63)
- 4) celebrity experience/name drop (2)
- 5) cultural activities (19)
- 8) exciting adventure (34)
- 9) you'd be amazed at what I experienced (3)
- 61) writer/we traveled (91)
- 62) family member traveled (37)
- 78) vacation (31)

Figura 7: Exemplo de categorias resultantes da codificação axial (Banks *et al.*, 2000)

Na última etapa, a codificação seletiva, é eleita a categoria de maior relevância, e a partir dela a teoria final é escrita. Esta categoria pode ser uma já existente, ou até mesmo uma nova categoria pode ser criada, pois o processo de análise dos dados

obtidos não para, e novas categorias podem ser criadas para satisfazer necessidades da pesquisa.

Neste estudo os códigos foram mapeados seguindo os pilares da Teoria da Difusão da Inovação: inovação, canais de comunicação, sistema social e tempo. Isto foi feito desta forma para seguir fielmente a Teoria de Difusão da Inovação, visto que foi através dela que o guia de entrevista foi elaborado, e, de maneira análoga os resultados serão apresentados no capítulo 4. Portanto, para este estudo, foram 4 códigos principais escolhidos: Inovação, Sistema Social, Canais de Comunicação e Tempo.

3.6. Debriefing

O *debriefing* (Fanning e Gaba, 2007) tem como intuito analisar os eventos e os comportamentos dos participantes, bem como identificar pontos onde as entrevistas podem ser melhoradas ou relatar fatos interessantes relacionados à pesquisa. Assim, esta abordagem foi utilizada após cada entrevista realizada, através de um formulário que era preenchido pelo autor deste estudo. O *debriefing* proporciona aos pesquisadores a possibilidade de expressar o que aconteceu durante o levantamento de dados, neste caso durante as entrevistas com os desenvolvedores de sistemas, facilitando assim, a discussão para a construção conjunta do conhecimento. O modelo de *debriefing* utilizado neste estudo está no Apêndice 2.

3.7. O Contexto da Pesquisa

Este estudo descreve um estudo que visa compreender como os desenvolvedores de software escolhem para qual ecossistema de software móvel irão desenvolver, ou seja, adotar, conforme mencionado na Seção 1.3. Assim, 18 desenvolvedores de diferentes partes do Brasil foram entrevistados. Estes engenheiros de *software* foram

escolhidos por já desenvolverem aplicativos para dispositivos móveis. Os dados foram obtidos através de entrevistas semiestruturadas conduzidas em duas fases: uma em de Junho de 2013 e a segunda em Agosto de 2014. Na primeira fase foram entrevistados apenas engenheiros de *software* do estado do Pará. Após esta primeira fase, as entrevistas foram transcritas, codificadas e analisadas de forma preliminar. Na segunda fase, foram entrevistados não só engenheiros de *software* do estado do Pará, mas também de outros estados brasileiros. Após a segunda fase, as entrevistas também foram transcritas e integradas ao conjunto de códigos já criados na primeira fase. Esta integração contou com a ajuda do orientador deste estudo, de outro aluno de mestrado e de um professor de outra instituição de ensino. As entrevistas duraram em média 25 minutos e geraram 67 páginas de transcrições. Os códigos foram estritamente mapeados para os 4 pilares da Teoria de Difusão da Inovação (Rogers, 2003).

A seguir, estas 18 pessoas são brevemente descritas com fins apenas de caracterização dos desenvolvedores. Todos os dados são anônimos, por questões de confidencialidade.

Pessoa 1 (P1) – 24 anos de idade, graduando em Licenciatura da Computação, com 6 meses de experiência em desenvolvimento para Android.

Pessoa 2 (P2) – 24 anos de idade, mestre em Engenharia Elétrica com ênfase em Computação Aplicada, com 2 anos de experiência em desenvolvimento para iOS.

Pessoa 3 (P3) – 25 anos de idade, Bacharel em Ciência da Computação e Engenharia de Controle de Automação, com 2 anos e meio de experiência em desenvolvimento iOs.

Pessoa 4 (P4) – 22 anos de idade, Bacharel em Ciência da Computação, com 2 anos e meio de experiência em desenvolvimento iOs.

Pessoa 5 (P5) – 26 anos de idade, Mestre em Ciência da Computação, com 2 anos de experiência em desenvolvimento para Android e 4 meses em desenvolvimento para iOS.

Pessoa 6 (P6) – 33 anos de idade, Bacharel em Engenharia da Computação, com 2 anos de experiência em desenvolvimento para Android.

Pessoa 7 (P7) – 27 anos de idade, Bacharel em Ciência da Computação com MBA em Engenharia da Computação Avançada, com experiência em desenvolvimento para Android e iOS.

Pessoa 8 (P8) – 24 anos de idade, Bacharel em Engenharia da Computação, com 2 anos de experiência em desenvolvimento para Android.

Pessoa 9 (P9) – 29 anos de idade, formado em Filosofia, com 1 ano e meio de experiência em desenvolvimento Android.

Pessoa 10 (P10) – 28 anos de idade, formado em Tecnologia, Análise e Desenvolvimento de Sistemas, com experiência em desenvolvimento Android.

Pessoa 11 (P11) – 23 anos de idade, Bacharel em Ciência da Computação, com 1 ano de experiência em desenvolvimento Android.

Pessoa 12 (P12) – 20 anos de idade, graduando em Engenharia da Computação, com 1 ano de experiência em desenvolvimento Android.

Pessoa 13 (P13) – 24 anos de idade, formado em Sistemas para Internet, especialista em Engenharia de Software Aplicada, com 2 anos de experiência em desenvolvimento Android.

Pessoa 14 (P14) – 24 anos de idade, Bacharel em Ciência da Computação, com 7 meses de experiência em desenvolvimento Android.

Pessoa 15 (P15) – 28 anos de idade, Bacharel em Ciência da Computação, com 6 anos de experiência em desenvolvimento Android.

Pessoa 16 (P16) – 23 anos de idade, Bacharel em Sistemas de Informação, com 1 ano e meio de experiência em desenvolvimento Android.

Pessoa 17 (P17) – 25 anos de idade, bacharel em Ciência da Computação, com 2 anos de experiência em desenvolvimento para Android e iOS.

Pessoa 18 (P18) – 36 anos de idade, formado em Publicidade e em Ciência da Computação, com 1 ano de experiência em desenvolvimento para Android.

Como dito anteriormente, os dados foram analisados utilizando técnicas da Teoria Fundamentada em Dados e coletados através de entrevistas semiestruturadas. As técnicas da Teoria Fundamentada em Dados forneceram uma perspectiva na qual foi possível analisar os dados coletados das entrevistas e compreender quais fatores podem levar um desenvolvedor a escolher um ecossistema de software móvel. O próximo capítulo apresentará os resultados obtidos da coleta de dados e a análise realizada nesta pesquisa.

4. RESULTADOS

Conforme dito anteriormente, essa pesquisa busca melhor compreender melhor o cenário de ecossistemas de *software* móveis tendo como perspectiva os engenheiros de *software*. Assim, foram realizadas entrevistas semiestruturadas com desenvolvedores, tendo como base a Teoria de Difusão da Inovação de Everett Rogers. Com os resultados das entrevistas, foi possível perceber, por exemplo, os motivos que levam a plataforma Android ser mais popular que a plataforma iOS na visão dos entrevistados ou que a experiência de um adotante de determinada tecnologia poderá influenciar sua percepção quanto a uma inovação que surja na sequencia, especialmente se elas dizem respeito a tecnologias ou ideias semelhantes.

Nas seções seguintes, serão apresentados os resultados obtidos após a análise dos dados, classificando os mesmos de acordo com o modelo proposto da Teoria de Difusão da Inovação (os pilares descritos na seção 2.1.2) para a melhor compreensão do mapeamento da Teoria de Difusão da Inovação com os resultados identificados.

4.1. Quanto a Inovação

De acordo com Rogers, a inovação é definida por ideias, práticas ou objetos que sejam percebidos como novo pelo indivíduo, mesmo que para outros em algum outro lugar esta “inovação” já é ultrapassada. Se uma ideia parece nova para o indivíduo, é uma inovação. Nas seções seguintes os resultados mapeados para os atributos da inovação serão expostos.

4.1.1. Quanto a Testabilidade

A testabilidade é definida por Rogers (2003) como sendo o grau com que a inovação pode ser experimentada antes de ser adquirida, isto consiste em efetivamente colocar as mãos na inovação antes de realmente adotar o mesmo. Neste sentido, a

existência de um maior número de usuários que utilizam determinada inovação, ou a facilidade de se poder testar, neste caso o desenvolvimento de aplicações mobile, pode representar um maior grau de testabilidade para a plataforma. Foi identificado que a plataforma Android possui maior grau de testabilidade, se comparada ao iOS.

Alguns trechos retirados das entrevistas que afirmam os parágrafos acima:

> "*...o Android é o sistema operacional, que se você for olhar no mundo, deve ter mais de 50% das pessoas utilizando, até porque o Android tá desde o LG ao Samsung, então tem vários modelos, vários preços acessíveis ...* " – (P1)
>
> *"...depois veio o boom do Android,* [que] *teve diversos dispositivos muito baratos e ainda tem ... não é a toa que, disparadamente, ele vai ser a maior plataforma usada, não tem como combater isso."* (P3*)*
>
> "*a quantidade de pessoas que tem smartphone Android, de umas 10 pessoas, pelo menos umas 9 têm.*". - (P1)

Os trechos acima sugerem que o fator testabilidade está mais inclinado para a plataforma Android que iOS, ou seja, é mais fácil testar a plataforma Android justificando, possivelmente, inclusive o fato da plataforma Android ser mais difundida que a sua concorrente, iOS. E como relatado, este maior grau de testabilidade é provavelmente em decorrência dos preços acessíveis, no Brasil, para se comprar um *smartphone* que possua o Android.

Esta foi realizada com um número maior de desenvolvedores da plataforma Android, então os resultados podem estar parciais. A fatia de desenvolvedores Android é grande no mercado mundial[3], o que talvez justifique o fato de ter sido mais comum encontrar desenvolvedores Android. Portanto, isto explica o grau de testabilidade sendo identificado como mais alto para esta plataforma, dado ao grande número não só de desenvolvedores, como também o grande número de usuários.

[3] http://venturebeat.com/2013/07/17/6000-mobile-developers-android-most-popular-ios-most-profitable-windows-phone-most-next/

Em relação ao conceito de testabilidade também foram encontrados relatos de usuários que elogiaram bastante outros fatores da plataforma Android. Os temas principais foram: a facilidade de distribuição do aplicativo, a facilidade de encontrar equipamentos baratos (*smartphones*, computadores, etc.) e a facilidade de distribuir os aplicativos para os usuários os aplicativos, ou seja, a facilidade de efetivamente instalar o aplicativo no *smartphone* do usuário.

Levando isso de encontro com o iOS, os desenvolvedores da plataforma que foram entrevistados, relatam os mesmos fatores acima, mas como um aspecto de dificuldade para testar: os equipamentos necessários para iOS possuem um alto preço e a distribuição de aplicativos é mais complicada.

Quanto à facilidade de distribuição de aplicativos, o entrevistado P9 relata:

> "*Na Apple Store você precisa dos IDs dos aparelhos que vão testar, no Google só precisa do e-mail e uma conta. Enfim, vejo que tem vantagens pro desenvolvedor que quer fazer com que a iteração do desenvolvimento seja mais rápida no Android que no iOS."*.

P14 afirma quanto aos equipamentos: "*Cheguei a ver na universidade Android e iOS, mas para o Android precisa-se de pouca coisa para desenvolver*." Vale ressaltar que para desenvolver para iOS é necessário computadores Apple, e P14 também complementa os comentários anteriores no que diz respeito ao teste com usuários: "*A gente testa a app por amostra, passa o aplicativo para alguns amigos/familiares e a gente aguarda o feedback. A gente passa por e-mail, por pen drive.*". Já no iOS o processo é diferente, como relata P6: "*Para o iOS é mais burocrático, temos que liberar o aparelho para testes. Usando o ambiente da Apple, e eles recebem o aplicativo, instalam e recebemos o feedback por e-mail.*". Em resumo, estas entrevistas corroboram o que já foi dito anteriormente, a facilidade em testar aplicativos, seja no seu ambiente de trabalho, seja com usuários reais, é mais simples para os desenvolvedores Android, que não enfrentam tantas barreiras (financeiras ou técnicas) como os desenvolvedores iOS, que, como citado por P6 tem o seu processo mais controlado.

4.1.2. Quanto a Complexidade

Conforme visto na Seção 2.1.3, a complexidade é definida por Rogers (2003) como sendo o grau que uma inovação é percebida como difícil de entender ou ser utilizada, portanto, quanto mais simples de ser utilizada, mais facilmente a inovação tende a ser adotada. Desta forma, foi identificado que ambas as plataforma iOS e Android são fáceis de serem aprendidas, ambas possuem interface intuitiva, linguagem simples, possuem uma documentação extensa e com diversos exemplos, etc. Porém a plataforma iOS foi considerada mais difícil de ser utilizada, pela necessidade de se ter um Mac Book, que é um notebook exclusivo da Apple, e outras ferramentas necessárias para o desenvolvimento, confome P2 relata: *"Cara, ela precisa de um Mac... essa é a coisa mais chata! Ela obrigatoriamente, ela tem que ter um Mac."*

O desenvolvedor P3, que desenvolve aplicações para o iOS, afirma:

> *"A documentação é sensacional, isso é uma das melhores coisas que tem neles, a documentação é muito boa, quando tu tem o Xcode, tu consegue direto ver a documentação dentro dele..."*.

P13 tem uma declaração semelhante quanto a plataforma que ele adotou:

> "*A documentação do próprio Android é muito boa. Ele tem uma série de tutoriais muito intuitivos de treinamentos básicos. Foi a minha primeira experiência de como funciona o universo Android.*".

Falando agora em aspectos técnicos da complexidade, que se trata de aprender a programar para determinada plataforma, seja ela iOS ou Android, aqui podem ser encontrados desenvolvedores que acham mais fácil programar para a respectiva plataforma em que já está inserido. Por exemplo, P4, um desenvolvedor iOS relata o seguinte: "*Eu já estudava programação na faculdade e aprender Objetive-C é bem mais simples do que Java, então foi bem mais fácil para mim. Essa questão de barreiras eu não tive*." Por outro lado, P17, desenvolvedor Android apresenta uma opinião contrária:

"*O Android foi mais simples pois é Java, na minha graduação estudei java e por isso nao tive tantos problemas.*" Em outras palavras, o que se pode concluir de acordo com o que foi coletado dos entrevistados é que o aspecto técnico é subjetivo, isto é, depende de cada pessoa: uma pessoa pode achar que por aprender Java na universidade influencia bastante o desenvolvedor a escolher o Android como sua plataforma, mas como visto no exemplo de P4 e de outros entrevistados, isso nem sempre é verdade. Portanto, o fator complexidade da Teoria de Rogers é entendido aqui como algo relativo. Não é possível dizer que uma plataforma possui vantagem sobre a outra a partir dos relatos das entrevistas, pois o que é complexo para um, não necessariamente é complexo para outro.

4,1.3. Quanto a Observabilidade

A observabilidade é definida por Rogers (2003) como sendo o grau com que os benefícios de uma inovação são visíveis a outros indivíduos. Portanto, pode-se assumir que quanto mais indivíduos utilizam determinada tecnologia, maior será o grau da observabilidade da mesma. Assim, de acordo com os dados deste estudo, a plataforma Android apresenta-se como a de maior grau em relação à plataforma iOS. Por exemplo, no contexto das entrevistas realizadas para este estudo, a maior parte dos entrevistados eram desenvolvedores da plataforma Android. P3 deixa clara esta conclusão quando afirma:

> *"Lá na Federal* (Universidade Federal do Pará) *tem curso de Android, então se tu for lá pra Federal* (UFPA) *é muita gente com Android, e fato de você precisar de um Mac para desenvolver para iOS, e os notebooks não são baratos, então isso facilite com que o cara vá trabalhar com Android*.".

P1 também afirma o seguinte a este respeito:

> *"É muito difícil tu encontrar alguém que tenha um iPhone e também para desenvolver para iPhone é meio complicado, pelo menos pra mim, porque eu não sei direito o que precisa, parece que tem que ter um Mac, já o Android não,*

basta você ter o Eclipse e qualquer sistema operacional e também se tu fizer uma pesquisa com a quantidade de pessoas que tem smartphone Android, de umas 10 pessoas, pelo menos umas 9 têm.".

Tendo isto dito, o alto número de usuários Android, possivelmente devido à fatores como: hardware barato, *software* livre e alto número de modelos de *smartphones* com o Android instalado, é um fator que aumenta a observabilidade, levando a mais desenvolvedores a notarem que é uma boa oportunidade desenvolver para Android, do que para iOS, que possui dispositivos mais elitizados e caros. P12 relata este efeito:

"*É muito difundido (Android), muitos smartphones mais baratos possuem, eu não tenho um poder de compra muito alto... você tem ali um smartphone e você sabe que dá pra fazer muito com ele e você não tem muito dinheiro, então você ali e descobre rápido... é open source...",*

P13 também confirma esta mesma observação:

"*eu escolhi Android pela abrangência do mercado pela facilidade de desenvolver as ferramentas são free, é fácil você se estabelecer com o ambiente de desenvolvimento que não gera custo diferentemente do iOS que você tem um custo inicial e anual.*"

4.1.4. Quanto a Compatibilidade

A compatibilidade é o grau com que a inovação é percebida como compatível com as atividades diárias do indivíduo, experiências passadas ou suas necessidades. Devido a linguagem Java ser mais adotada que Objective-C, tanto em universidades quanto em empresas privadas[4], pode-se identificar que possivelmente existe uma maior compatibilidade à plataforma Android, por utilizar a linguagem Java, que a plataforma iOS que utiliza a linguagem Objetive-C.

[4] http://spectrum.ieee.org/computing/software/top-10-programming-languages

A compatibilidade, nesta pesquisa em específico, está de forma intrínseca ligada com o fator de observabilidade. Ao notar a tecnologia, de acordo com Rogers (2003), vem o processo de avaliação da mesma, que se dá com pesquisas, contatos pessoais e outras formas de busca tem como objetivo conhecer a inovação e ver se a mesma possui compatibilidade com as atividades diárias ou se é compatível com uma necessidade.

Portanto, alguns dos resultados encontrados na seção de Observabilidade (ver Seção 4.1.3) podem ser instanciados nesta seção. Por exemplo, o comentário de P12, que ao observar as facilidades que o Android apresentava, achou compatível com sua situação financeira e com a sua vontade de desenvolver para dispositivos móveis. P13 também é outro exemplo, que ao notar a abrangência do mercado do ecossistema Android e a sua falta de custos fixos, decidiu adotar o Android, pois era o mais compatível com suas atividades diárias.

O grau de Compatibilidade também está ligado com o grau de Complexidade, no que diz sentido que se uma pessoa percebe que algo é muito complexo, difícil de ser aprendido, por mais que seja compatível com sua necessidade atual, ela pode ir procurar outra solução mais simples, que faça a mesma coisa. Um exemplo deste caso é mencionado por P4 que mencionou na seção de Complexidade (ver Seção 4.1.2): "*Eu já estudava programação na faculdade e aprender Objetive-C é bem mais simples do que Java, então foi bem mais fácil para mim. Essa questão de barreiras eu não tive*." Em outras palavras, por mais que P4 já estudasse Java na universidade, ele preferiu recorrer ao Objective-C da Apple, por achá-lo mais fácil que o Java que já era ensinado em seu curso.

Sendo assim, a partir dos resultados encontrados nas entrevistas, pode-se concluir que o grau de Compatibilidade é influenciado pelo grau de Complexidade e pelo grau de Observabilidade, portanto, os seus resultados deste aspecto são herdados dos outros aspectos já mencionados nas seções anteriores. Além disso, como se trata de uma análise subjetiva, não é possível dizer que, em relação a este aspecto, exista uma plataforma com vantagem sobre a outra.

4.1.5. Quanto a Vantagem Relativa

A vantagem relativa é o grau com que a inovação é percebida como melhor que a ideia antecedente. O grau de vantagem relativa pode ser medido em relação a vários fatores, como por exemplo, prestígio social, economia de tempo ou dinheiro, etc. Assim, foi percebida uma vantagem relativa do ecossistema da Apple, o iOS, em relação ao Android: ele gera mais renda aos desenvolvedores de acordo com os entrevistados. Os entrevistados relataram que as pessoas que possuem aparelhos Apple, tem um maior poder aquisitivo que as pessoas que possuem aparelhos Android. Por exemplo, P4 diz:

> "*Eu acho que a cultura das pessoas que possuem iPhone é diferenciada na postura de compra de aplicativos ... elas não tem pena em gastar em um aplicativo, são mais consumistas, diferente do Android, que já é mais popular ... as pessoas ainda relutam em comprar aplicativos, tanto que o maior número de aplicativos que ela tem é de graça.*"

O Android está segmentado entre diversos modelos de smartphones e as pessoas que adquirem bens Apple, já conhecem a marca e sabem dos custos para compra de um aparelho, e por este motivo, estão mais inclinadas a comprar aplicativos na loja virtual. P3 comenta sobre este aspecto:

> "*A mentalidade de quem usa Android é muito diferente, a pessoa ela... por exemplo, a maior dificuldade é quando a gente faz um aplicativo pago... Então, se ele é pago e é de iOS, as pessoas compram, se ele é pago e é de Android as pessoas não compram. É ridículo isso... por causa da mentalidade das pessoas que usam.*"

Entretanto, existem pontos positivos também levantados por desenvolvedores Android, que neste caso, torna o seu ecossistema mais vantajoso que o ecossistema do iOS. Por exemplo, existe a questão do baixo custo para se desenvolver para Android, se

comparado ao iOS: este ponto já foi levantado em seções anteriores, mas é algo que é relacionado também ao grau de Vantagem Relativa. Neste caso, P11 diz:

> "*Primeiro que você não precisa pagar nada, é de graça, é livre... é muito acessível...*", P10 também fala algo semelhante: "*Primeiro que* (o Android) *é gratuito, o iOS é um pouco caro, tinha que ter o Macintosh, muito bloqueio, e o Android tinha mais facilidade pelo fato dele ser opensource, eu escolhi por causa disso, não que eu não goste de iOS, mas o Android era mais acessível.*"

P15, por outro lado, traz outra visão, a da abrangência do Android: "*A comunidade é uma das maiores de desenvolvimento. Justamente pelo fato dela ser free*." Por fim, o informante P8 exalta em seu relato o papel da Google em lançar constantes atualizações para o Android, como um fator motivador para perseverar no desenvolvimento para Android:

> "*O sistema operacional está melhorando cada vez mais ... acabaram de lançar o KIT KAT, então da pra perceber que não querem deixar o Android morrer, sabe? Então o respaldo de quem está distribuindo o sistema operacional junto com a comunidade, que é forte, torna muito forte* [a plataforma] *e me influenciou.*"

Como o próprio nome já diz, Vantagem Relativa, este aspecto também tende a ser subjetivo, pois a visão de vantagem é a de cada entrevistado. Como neste estudo foram entrevistados mais desenvolvedores Android, foram obtidos mais dados sobre esta plataforma, o que pode influenciar no resultado final desta seção, por isso não irá ser afirmado que uma plataforma possui mais vantagens relativas que a outra, apenas ocorre a sugestão de que o Android possui mais vantagens que o iOS.

Este atributo conclui a análise dos aspectos de inovação encontrados nas entrevistas. Na próxima seção serão analisados os pilares de Sistema Social e Canal de Comunicação.

4.2. Canais de Comunicação e Sistema Social

Esta seção apresentará os resultados coletados das entrevistas, mapeados para os dois pilares sociais da Teoria de Difusão da Inovação. A escolha de unir estes pilares se dá pelo fato de que, conforme apresentado anteriormente na seção 2.1.3, ambos possuem cunho social. Assim, as respostas das entrevistas acabam todas convergindo para a unificação dos mesmos.

Nas perguntas relacionadas à como eles se relacionam com membros da comunidade – o seu grau de atuação, engajamento e contribuição – foi observado que os entrevistados, em sua maioria, não são membros muito atuantes de comunidades e fóruns de discussão, mas exaltam a importância da comunidade na ajuda para resolver problemas, como P13 afirma:

> *"É muito importante. Dificilmente você consegue fazer sozinho. Então participar é um dos melhores caminhos para solucionar problemas. Você participando de um nicho de pessoas em comum te ajuda a motivar, estudar mais, testar algo novo, resolver teu problema. Hoje em dia não tem como resolver as coisas sozinho. Muitos já passaram pelo mesmo problema então isso ajuda bastante. É um enorme repositório de conhecimento onde as pessoas podem ajudar. É como uma prestação de serviço gratuita*."

Todos os entrevistados relataram apenas se relacionar com as pessoas, principalmente através de redes sociais, fóruns de discussão e listas – canais de comunicação de massa. Neste caso, o Facebook foi citado como a principal rede social utilizada, enquanto que o Stack Overflow foi bastante citado como o principal fórum para sanar dúvidas. O informante P12 corrobora isso: "*O Stack Overflow eu uso muito. Atualmente, é um dos melhores que tem, uma dúvida rápida você, com certeza, acaba encontrando ali. Eles explicam bem o problema na maioria das vezes."* Essa afirmação deixa clara a importância dos canais de comunicações para que o usuário possa continuar no desenvolvimento para a plataforma escolhida. Para que exista sucesso, também é possível a utilização de canais de comunicação menores, ou o "boca-a-boca",

ao invés de apenas canais de comunicação de massa, como o Stack Overflow. O contato direto com desenvolvedores é relatado como um aspecto importante para sanar dúvidas e até para o nivelamento de um grupo de desenvolvedores, P15 informa: "*A gente troca conhecimento em DOJOs,* [nos] *reunimos para realizar desafios. Minha equipe é composta de desenvolvedores plenos, iniciante e sênior. A gente tem que nivelar. Eu gosto de ensinar, minha equipe é muita comum nessas práticas*". O desenvolvedor P17 relata algo semelhante: "*Tento primeiramente entrar em contato com pessoas que já passaram por esses problemas. Mas se não, vou para internet. Eu pesquiso no Google...".*

A frequência com que os entrevistados participam é muito relativa, com pouca contribuição para o crescimento do conhecimento acerca do ecossistema em que eles fazem parte. P3 evidencia este fato:

> "N*não, cara, eu não participo de nenhuma comunidade, no máximo eu tenho uns dois grupos no facebook, aonde os caras postam as dúvidas (...) a frequência varia muito, tem vezes que eu posto umas cinco ou seis vezes no mês e tem vezes que passa um mês e eu não posto nada, varia muito de acordo com a pergunta que a pessoa fez lá, com a dúvida que eu tenh*o".

Por outro lado, existem informantes que costumam participar de forma mais ativa, como é o caso de P13: "*Já contribui. Já ajudei em tópicos de fóruns, já fui ajudado também. Ultimamente não ando muito ativo, mas já participei, publiquei problemas que eu tive, no Stack Overflow.*" também foram entrevistadas pessoas que inclusive realizam treinamentos, como é o caso de P15: "*Eu costumo responder bastante, faço treinamentos também, dentro e fora da minha cidade".*

Apesar da maioria dos entrevistados não contribuir com frequência para as comunidades que participam, eles sempre as consultam para se atualizar sobre como anda o desenvolvimento dos seus colegas de ecossistema, se estão progredindo ou não. Eles também relatam a importância que essas comunidades possuem para as pessoas

que estão começando a desenvolver para um determinado ecossistema, como por exemplo o relato de P3:

> *"Eu acho importante, porque tem muita gente que entra no grupo e tá aprendendo, daí posta uma dúvida e pessoal ajuda, nunca vi ninguém ser grosseiro, então acho que ajuda bastante a aprender iOS*", e complementa: "*eu acho super importante o pessoal ter um grupo para participar, para tirar dúvidas, ter pessoas que tu possa contar*".

Alguns entrevistados relatam que foram fortemente influenciados por amigos que já desenvolviam para as plataformas em questão, ou seja, o sistema social em que eles estavam inseridos foi um fator determinante para a decisão dos mesmos em adotar um determinado ecossistema. Isto pode ser evidenciado por P1: *"tem um colega da UFRA* (Universidade Rural da Amazônia), *que desenvolve, eu fui vendo e gostando*." O informante P3, cita que dois amigos o influenciaram, H. e J.: "*O J. desenvolvia pra Android e o H. desenvolvia pra iOS, e aí vendo o desenvolvimento deles eu fiquei interessado e comecei a estudar também*". Entretanto, alguns entrevistados não citam diretamente algum amigo, mas que relatam que ao verem pessoas utilizando uma determinada plataforma, acabaram se interessando pela mesma, como P11 exemplifica: "*foi algo natural, a descoberta, no dia a dia as pessoas já estavam falando de Android, de desenvolvimento*", ou seja, mesmo que não exista uma pessoa específica que influencie diretamente, o sistema social é muito importante como fator motivador para adoção de um determinado ecossistema de software.

Neste estudo, os participantes que aprenderam Java anteriormente tinham uma maior compatibilidade com o Android, o que tornou mais fácil para eles a adoção desta plataforma, conforme ilustrado abaixo por P4:

> *"iOS foi mais complicado, devido à sua linguagem* [de programação], *eu fiz algumas coisas, mas realmente não entendia como funcionava. Android era mais simples porque é Java, que eu estudei no meu curso de graduação."*

Em outras palavras, a citação anterior ilustra como o contexto geral, ou o seu sistema social, no qual um programador está inserido influencia a sua decisão. Neste caso, isso está relacionado com a universidade[s] que ele/ela estudou, o conhecimento anterior que possuía. Este caso é diferente das citações anteriores, de entrevistados que foram influenciados por seus amigos; aqui o ambiente, a universidade teve um papel fundamental na escolha.

A adoção de um ecossistema por parte de um engenheiro de *software* também é influenciada pela sua identificação pessoal com as decisões de governança feitas pelas empresas que governam o ecossistema. Por exemplo, alguns participantes ficaram entusiasmados com o desenvolvimento para uma plataforma de código aberto e, portanto, rejeitaram o iOS por considerá-lo uma plataforma fechada. Eles gostaram de ter a possibilidade de modificar o código fonte do Android e moldá-lo às suas necessidades. P12 comprova esta afirmação: *"Eu participo de algumas comunidades de código aberto. Então isso foi algo que me influenciou para escolher Android, porque é mais aberto do que a plataforma iOS. "*

A decisão de adotar um ecossistema também é influenciada pela abrangência que os canais de comunicação e o sistema social possuem. Um maior número de usuários inseridos em um ecossistema pode indicar maiores chances de seu aplicativo ser utilizado. Um estudo da Kantar Worldpanel ComTech[5], indica que a plataforma Android estava presente em 89% dos *smartphones* brasileiros até o final do segundo trimestre de 2014. O entrevistado P5 corrobora esta afirmação: *"Eu diria que depende da finalidade, se se quer chegar a um maior número de usuários, tem de se desenvolver para o Android, porque há poucas pessoas no Brasil que possuem um dispositivo iOS."*

Neste estudo, foi descoberto que desenvolvedores externos valorizam a participação e engajamento dos colaboradores internos (empregados das empresas que governam o ecossistema) em sites de redes sociais e meios de comunicação social. O informante P16 apoia esta afirmação:

[5] http://www.kantarworldpanel.com/global/smartphone-os-market-share/

"Eu não sei quem são os caras que desenvolvem o iOS, eu não sei como eles se comportam em redes sociais, enquanto os desenvolvedores do Android deixam claro como você pode entrar em contato com eles ".

O informante P12 também confirma que esta transparência existe é um fator motivador: *"A comunidade Android se comunica no Google Groups ou Google+. [...] Há mesmo a participação de membros da equipe do Google, o que nos motiva muito"*.

Foram apresentados nesta seção os pilares de caráter social da Teoria de Difusão da Inovação, uniu-se Sistema Social e Canais de Comunicação em um único tópico para facilitar a análise dos resultados.

4.3. TEMPO

O pilar de tempo, como explicado na Seção 2.1.2 é o pilar aonde ocorre todo o processo de adoção de inovação. Nesta pesquisa este pilar não foi explorado, o que caracteriza isto como uma limitação do trabalho. Não foi encontrada uma forma de cobrir este pilar neste contexto de pesquisa, então para que não fossem gerados dados equivocados, este pilar foi suprimido.

Este capítulo encerra a análise dos resultados obtidos neste estudo. No próximo capítulo será feita uma discussão entre os resultados neste estudo com outros estudos semelhantes.

5. DISCUSSÃO

Este capítulo tem como objetivo apresentar uma discussão entre os resultados desta pesquisa e compará-los com outros estudos semelhantes. Para melhor apresentação, a discussão será dividida em três tópicos: a Teoria de Difusão da Inovação, explicada na Seção 2.1 deste trabalho, aplicada à Engenharia de Software, TAM aplicado à Engenharia de Software e Ecossistemas de *Software* enquanto comunidades *online*.

Como em qualquer estudo empírico, este trabalho possui diversas limitações. Por exemplo, a maior parte dos entrevistados desenvolviam aplicativos para as plataformas Android ou iOS. Assim, os resultados encontrados neste trabalho podem não ser aplicáveis às plataformas Windows Phone, Blackberry, ou outras. De maneira similar, os resultados não foram classificados de acordo com as experiências dos entrevistados, idades ou regiões dos mesmos. Sendo que estes fatores podem influenciar nos resultados obtidos. Desta forma, dependendo do contexto do informante este resultado ou opinião pode ser diferente. Este trabalho também focou apenas em engenheiros de *software* que residem no Brasil, se tal pesquisa fosse estendida para outros países, os resultados também poderiam ser diferentes. Por fim, o pilar Tempo, não foi explorado neste estudo (ver seção 4.3).

Conforme mencionado anteriormente, o estudo apresentado nesta dissertação tinha como foco entender o que leva engenheiros de *software* a adotar a plataforma ou plataformas para qual/quais desenvolvem. Com este intuito, foi utilizado um guia para entrevistas semiestruturado com engenheiros de *software* residentes no Brasil. O

Capítulo 4 apresentou os resultados, que foram estruturados de acordo com os pilares presentes na Teoria de Difusão da Inovação. Neste capítulo alguns deles serão discutidos com a literatura disponível.

5.1. Teoria da Difusão da Inovação aplicada à Engenharia de Software.

Esta seção analisa os artigos[6] encontrados na literatura em que a Teoria de Difusão da Inovação é aplicada a Engenharia de *Software*. Por se tratar de uma teoria bastante difundida, foi possível encontrar estudos datando de 1988, como o de Bayer e Melone, até 2013, como o de Gómez *et al*.

O primeiro estudo que foi identificado como utilizando a Teoria de Difusão da Inovação aplicada à Engenharia de *Software* data de 1988 com autoria de Bayer e Melone (1988). No mesmo é fornecida uma visão geral da Teoria de Difusão da Inovação e sugere a aplicabilidade desta teoria para predição da adoção de inovações tecnológicas, incluindo aquelas relacionadas à Engenharia de Software. O estudo criticamente avalia a teoria de Rogers apresentando elementos desta teoria que devem ser estendidos antes de ser aplicado à transição de tecnologias em geral e a Engenharia de *Software* em particular. Para ser mais específico, segundo os autores, a Teoria de Difusão da Inovação peca em alguns aspectos como: ela não fornece conceitos precisos sobre o que é adoção; não considera interação entre vários sistemas sociais e não integra na teoria a descontinuidade do uso da inovação. O resultado do estudo é uma proposta de modificações na Teoria de Difusão da Inovação, para melhor compreender e prever adoção no contexto de Engenharia de Software.

[6] A busca por esses artigos se deu através de consultas às bibliotecas digitais da ACM (*Association for Computing Machinery*), e da IEEE (*Institute of Electrical and Electronics Engineers*). Tais consultas foram realizadas utilizando palavras-chave como: *software engineering, diffussion of innovations theory*, Everett Rogers, dentre outras.

Já o estudo de Larsen e Kautz (1997) investiga a difusão e a adoção de processos de melhoria de desenvolvimento de *software* nas organizações de desenvolvimento de tecnologia na Noruega. O estudo apresenta o perfil dos usuários de tais processos no país e também apresenta meios para melhorar a difusão e adoção de processos de desenvolvimento de *software* no país em questão.

Este estudo era parte de um grande esforço capitaneado pelo projeto ESPITI (*European Software Process Improvement Training Initiative*) que buscava criar consciência e espalhar conhecimento sobre atividades de melhoria de *software* na Europa. O método de coleta de dados foi um questionário com 33 questões e várias subquestões, gerando um total de 72 de atributos para a análise. O questionário foi baseado em um *survey* realizado pelo projeto ESPITI, que buscou gerar uma visão geral da atividade de desenvolvimento de *software* na Europa. Após a sua elaboração, o questionário foi enviado para 1.250 organizações na Noruega. No total, 171 questionários foram recebidos como resposta, porém cinco não foram utilizados, totalizando 166 questionários para análise.

As empresas analisadas no estudo foram divididas em três grupos: o primeiro consiste em empresas que possuem certificações (como a ISO ou o TickIT) e que usam processos de controle de qualidade; o segundo grupo são empresas que não utilizam qualquer tipo de processo de garantia de qualidade; e o terceiro grupo, que são empresas que utilizam algum processo de controle de qualidade, mas que não possuem certificação. Em suas conclusões, Larsen e Kautz (1997) dizem que as empresas norueguesas ainda precisam melhorar bastante no que diz respeito à adoção de práticas de garantia de qualidade e utilização de processos de desenvolvimento de *software*. Esta adoção está diretamente ligada a espalhar o conhecimento de que estas melhorias são fundamentais para que as organizações norueguesas possam melhorar seus processos. No estudo apresentado nesta dissertação, pode-se notar que disseminar informações também é um fator determinante para que um ecossistema prospere: empresas governantes de um ecossistema que mantém maior transparência e maior contato são elogiadas por entrevistados.

Por outro lado, o trabalho de Larsen e Kautz (1997) se diferencia do realizado nesta dissertação por se utilizar de um meio de coleta de dados impessoal, enquanto que neste optou-se por entrevistas semiestruturadas para que conforme novos tópicos fossem surgindo, o entrevistador tivesse condições de aprofundar-se e coletar mais informações valiosas para a pesquisa.

Em Woo *et al.* (2006) a Teoria de Difusão da Inovação é aplicada para analisar a extensão com que a adoção da programação orientada a objetos é utilizada por desenvolvedores e as razões dela ser, ou não, adotada. Baseada no estudo conduzido, um instrumento é proposto para que as organizações possam saber se elas estão prontas para adotar este paradigma de programação. O propósito do estudo é ajudar a tomada de decisão por parte das empresas fornecendo o máximo de informações possíveis.

O estudo envolveu três fases distintas. Na primeira fase uma lista de perguntas abertas foi elaborada sobre a adoção de metodologias de desenvolvimento de *software*. Para validar estas perguntas foi feito um teste piloto, que consistiu de entrevistas com dez engenheiros de *software*, cada um com 15 anos de experiência. Usando os resultados destas entrevistas, foi elaborado um *guia* para as entrevistas. Para validá-lo também foram chamados membros de um comitê de tecnologia da informação ligados a uma universidade canadense. Por fim, todos os dados obtidos nas fases anteriores foram organizados e um questionário foi distribuído para 1.109 organizações de desenvolvimento de tecnologia no Canadá. Foram obtidas 82 respostas.

Em seus resultados Woo *et al.* (2006) relatam que adoção do paradigma de programação orientada a objeto, na época, era um grande desafio para as organizações canadenses. Das 82 organizações que responderam ao questionário, 35 adotavam o paradigma de alguma forma. A partir das respostas, um *framework* para auxiliar estas empresas foi criado e um estudo de caso foi realizado para validação do mesmo. Este estudo de caso foi realizado em uma única organização. Os resultados mostram que a empresa ainda não estava preparada para adotar o paradigma de orientação a objeto, mas com a ajuda do *framework* a empresa pode planejar seus próximos passos para enfim adotar o paradigma.

A metodologia utilizada por este estudo se assemelha com a apresentada nesta dissertação, onde perguntas foram inicialmente elaboradas, porém não foram validadas como no estudo de Larsen e Kautz (1997). Com o tempo elas foram atualizadas, conforme o entendimento sobre a Teoria de Difusão da Inovação aumentou, gerando o guia utilizado nas entrevistas que compõe esta dissertação.

Em Gómez *et al.* (2013) é realizado um estudo com o intuito de descobrir como os engenheiros de *software* descobrem e compartilham informação. Para isto, foi investigado o compartilhamento de *links* no *site* Stack Overflow[7], para determinar se os engenheiros de *software* disseminam informação através do compartilhamento de URLs com outros engenheiros de *software*. Três perguntas foram formuladas para guiar a pesquisa: (1) Que tipos de *links* os engenheiros de *software* compartilham no Stack Overflow? Esses *links* são para inovações relacionadas à desenvolvimento de *software*?; (2) Que tipos de *links* são compartilhados com mais frequência?; (3) Quais *websites* são mais referenciados em *links* no Stack Overflow? Para responder a estas perguntas foram analisados *posts* de Agosto de 2012. Foi realizada uma mineração de texto para buscar os *links* compartilhados no Stack Overflow. Após isto, foi gerada uma lista de *links* únicos e o número de vezes que cada *link* foi citado. Com este processo, 1,999,026 *links* únicos e um total de 4,197,085 citações foram identificadas.

Como resultado os pesquisadores concluem que a atividade de compartilhamento de *links* no Stack Overflow é algo fundamental para a comunidade. Uma parte significante dos *links* compartilhados no *site* se refere às inovações em desenvolvimento de *software*, como *links* para bibliotecas e ferramentas.

O Stack Overflow também foi relatado no estudo mostrado nesta dissertação como uma das principais fontes de busca de conhecimento pelos entrevistados. *Sites*

[7] O Stack Overflow é um site gratuito de perguntas e respostas sobre desenvolvimento de *software*. Perguntas e respostas feitas pelos usuários podem ganhar votos positivos e negativos, o que recompensa ou pune seus donos com pontos de reputação. Pode ser acessado através do endereço: http://pt.stackoverflow.com/

como este são fundamentais para engenheiros novatos, assim como para veteranos, pois aproximam os membros do sistema social e unem os participantes do ecossistema.

5.2. Technology Acceptance Method (TAM) aplicado a Engenharia de Software.

Esta seção analisa alguns estudos[8] do modelo *Technology Acceptance Method* (TAM), aplicado à Engenharia de Software. Não foram encontrados muitos artigos na literatura que aplicassem o modelo TAM ao contexto da Engenharia de Software.

Para estudar a aceitação de tecnologias, os motivos que leva a esta aceitação e outros fatores relacionados, também se pode usar o modelo proposto por F. D. Davis em 1989, chamado de TAM (*Technology Acceptance Model*). Davis, um pesquisador da área de Sistemas de Informação, propôs um modelo que fosse útil na previsão do uso de um determinado sistema, modelo este que avalia como a tecnologia é aceita.

O modelo se originou através de um trabalho conjunto da IBM com o MIT (*Massachusetts Institute of Technology*), no início da década de 1980, e tinha como intenção avaliar o potencial de produtos da IBM que estavam sendo lançados no mercado e auxiliar a mesma a melhor entender quais os fatores determinantes que levavam a utilização de computadores (Davis e colegas, 1989).

Para Davis e colegas (1989), as pessoas tendem a usar ou não uma tecnologia nova com o intuito de melhorar seu desempenho no trabalho, que em seu modelo ele chama de *utilidade percebida*. Porém, mesmo que essa pessoa entenda que uma determinada tecnologia é útil, sua utilização poderá ser prejudicada se o uso for muito

[8] A busca por esses artigos se deu através de consultas às bibliotecas digitais da ACM (*Association for Computing Machinery*), e da IEEE (*Institute of Electrical and Electronics Engineers*). Tais consultas foram realizadas utilizando palavras-chave como: *software engineering, Technology Acceptance Method*, TAM, dentre outras.

complicado, de modo que o esforço não compense o uso, o que ele denomina de *facilidade percebida*.

O TAM normalmente é utilizado para entender o porquê de o usuário aceitar ou rejeitar uma determinada tecnologia de informação e sugerir como melhorar a aceitação, oferecendo, desta forma, um suporte para prever e explicar a aceitação ou rejeição. A Figura 3, adaptada diretamente do modelo de Davis, sugere que os usuários utilizarão a tecnologia se perceberem que ela lhe trará benefícios, tendo como foco os dois pilares do modelo, a facilidade percebida e a utilidade percebida.

O modelo TAM é suportado por dois pilares fundamentais, ou crenças: facilidade percebida de uso e utilidade percebida. O primeiro se trata de aquela tecnologia nova parece ser fácil de ser utilizada e adotada, o segundo trata da real utilidade daquela tecnologia, se trará benefícios para sua rotina de trabalho e afazeres diários. Esses pilares se complementam, o usuário vai utilizar a tecnologia se perceber facilidade e utilidade. Uma vez isto percebido, ele irá decidir se vai utilizar a tecnologia e adotá-la em seu dia a dia, o que na Figura 3, segue como a intenção comportamental, que é a avaliação propriamente dita, seguida pelo comportamento, que é a adoção.

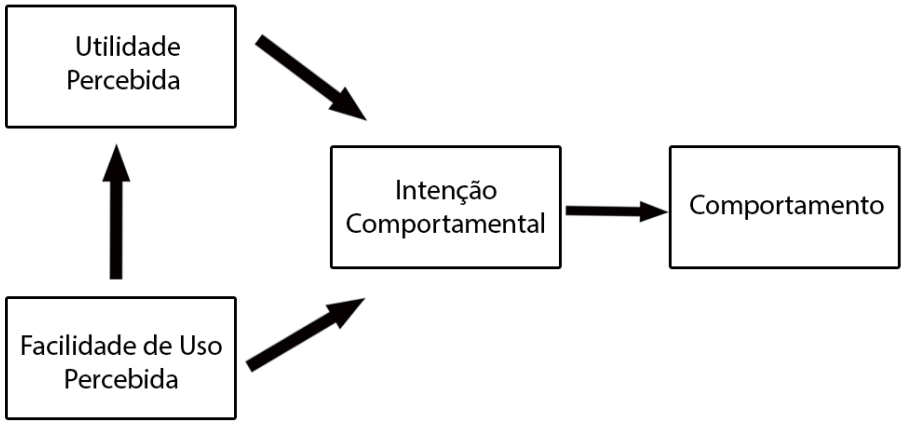

Figura 8: *Technology Acceptance Model* (TAM). Davis (1989)

No estudo de Chitungo e Munongo (2013), o TAM é estendido, sendo adicionados novos pilares como: vantagem relativa oriundos da Teoria de Difusão da Inovação (Rogers 2003), normas sociais, entre outros. Este estudo foi aplicado para investigar a aplicabilidade do modelo e sua eficácia em um ambiente de comunidades rurais do Zimbábue. Os pesquisadores conduziram um estudo pretendendo entender qual a predisposição que aquelas comunidades teriam para adotar serviços bancários móveis, sendo que a região não possui qualquer tipo de banco.

Um questionário foi elaborado, utilizando esta extensão do TAM como modelo conceitual e distribuído em quatro comunidades rurais do interior do país. Os resultados obtidos, com uma taxa de 69% de questionários respondidos, revelou com que o TAM pode prever intenções de consumidores em utilizar serviços bancários móveis. Sendo mais específico: a utilidade percebida, a facilidade de uso, vantagens relativas e normais sociais, são os atributos que mais tem mais efeitos significativos na percepção do usuário e na sua atitude, assim, influenciando a utilização de serviços bancários móveis.

Em Keung *et al.* (2004) a principal motivação para o estudo é investigar o impacto da introdução de uma nova tecnologia de estimativa de custos de *software* em uma pequena empresa de desenvolvimento de *software*. O estudo descreve uma tentativa de entender a razão da falha de se utilizar um novo método de estimativa de custos em uma organização de desenvolvimento de software.

O método de pesquisa empregado consistiu em um estudo de caso no contexto de uma pequena empresa de desenvolvimento de *software*. Foram realizadas entrevistas estruturadas com os engenheiros de *software* da organização estudada. O *Technology Acceptance Method* (TAM) foi utilizado no início e no final do estudo. Primeiramente, ele foi utilizado para medir o comportamento dos engenheiros de *software* quando um novo método de estimativa foi introduzido na organização, e depois foi utilizado para montar um questionário de validação dos resultados iniciais. No final da pesquisa, alguns fatores que prejudicaram na adoção do novo método de estimativa de custos foram identificados, como por exemplo: o modelo só fornecia uma solução parcial para as necessidades da organização; a usabilidade das ferramentas de apoio ao método de

estimativa era falha; houve falta de conexão entre o treinamento para utilizar o método e seu uso proposto, entre outros. A pesquisa realizada nesta dissertação também aponta alguns fatores que dificultam a adoção de uma inovação, neste caso a entrada de engenheiros de *software* em alguns ecossistemas. Problemas com emuladores, equipamentos caros e desconhecimento da linguagem de programação são alguns dos problemas relatados nos resultados apresentados no Capítulo 4.

Vieira *et al.* (2012) descrevem a definição e evolução da técnica REMO (*Requirements Elicitation oriented by business process MOdeling*), além de apresentar a condução de um novo estudo executado para avaliar a viabilidade de uso da segunda versão da técnica. Este artigo mostra as abordagens que foram identificadas na literatura por meio de um mapeamento sistemático e apresenta como o segundo estudo experimental foi planejado, conduzido e os resultados que foram obtidos. Além da análise quantitativa, os resultados foram também analisados qualitativamente seguindo os procedimentos de dois diferentes métodos: o (TAM) e o método da Teoria Fundamentada em Dados (Strauss e Corbin, 2008).

O primeiro estudo experimental foi realizado com o objetivo de avaliar a viabilidade do uso da técnica REMO para identificação dos requisitos a partir dos diagramas de processos de negócios. O segundo estudo foi elaborado segundo os princípios do paradigma GQM (*Goal Question Metric*). Para a análise qualitativa das causas que influenciaram na aceitação da técnica, utilizou-se procedimentos do método da Teoria Fundamentada em Dados – alguns procedimentos (etapas de codificação) da Teoria Fundamentada em Dados também foram utilizados no estudo apresentado nesta dissertação. Os resultados obtidos com os procedimentos de codificação permitiram visualizar quais as principais dificuldades que impactaram no uso da técnica durante a realização do estudo. Os códigos relacionados às dificuldades estão relacionados aos dados obtidos através do questionário do modelo TAM.

Este estudo experimental apontou que a técnica REMO contribui para a identificação de requisitos adequados ao contexto dos processos de negócios. Deste

modo, a técnica torna-se relevante para o desenvolvimento de software, evitando a aplicação de esforços dos analistas de sistemas em encontrar requisitos inadequados

O trabalho de Ciolkowski *et al.* (2007) teve como objetivo ilustrar a aplicação de um instrumento de avaliação desenvolvido para o que é chamado no trabalho de "centros de controle de *software*", que nada mais é que uma sala de controle geral. Um centro de controle de *software* tem que ser customizado de acordo com o *software* que está sendo desenvolvido no momento. O estudo combinou abordagem quantitativa e qualitativa. O instrumento de avaliação foi aplicado em seis estudos de caso. O modelo TAM foi utilizado para a criação deste instrumento de avaliação, principalmente no que tange a facilidade de uso e a utilidade, atributos básicos do modelo.

5.3. Ecossistemas *software* como comunidades *online*

Em 1995, Sproull e Faraj introduziram uma visão da internet como "uma tecnologia social que permite que pessoas com interesses comuns para encontrar um ao outro, reunir e manter conexões ao longo do tempo.". Seguindo essa interpretação, comunidades on-line - definidas como "pessoas com interesses comuns ou metas para quem a comunicação eletrônica é uma forma primária de interação." (Dennis *et al.* 1998) - foram desenvolvidos para diferentes temas. Nessas comunidades, a troca de informações é o principal objetivo (Ridings e Gefen, 2004).

A peça central da maioria das comunidades online é um único site para que os membros devam contribuir. Por exemplo, a comunidade online que alimenta a Wikipedia tem o objetivo de melhorar adição de conteúdo a enciclopédia on-line. Os aspectos sociais são muitas vezes percebidos em tais comunidades online como um efeito colateral. Por exemplo, Forte *et al.* (2012) constataram que o WikiProjects - grupos de contribuintes que querem trabalhar juntos como uma equipe para melhorar a Wikipedia - não só ajudam a produzir artigos, mas também proporcionam um lugar para encontrar colaboradores e facilitar a socialização e *networking*. Uma vez que estas comunidades online precisam de usuários para criar conteúdo para seu *website* da

comunidade, o seu sucesso é determinado por meio de sociabilidade e usabilidade (Preece, 2001).

Enquanto ecossistemas de *software* são semelhantes às comunidades on-line, seu principal objetivo não é a criação de conteúdo para o site da comunidade. Em vez disso, o objetivo é o desenvolvimento de aplicações. Como ficou evidente nas entrevistas, os desenvolvedores de *software* usam vários meios de comunicação e troca de conhecimentos quando participam de um ecossistema, que vão desde o *website* Stack Overflow e outros canais de mídia social para amigos e colegas. Além disso, eles usam várias ferramentas e ambientes de desenvolvimento para projetar, desenvolver e lançar seus aplicativos.

Em um estudo sobre a comunidade de escrita colaborativa *Everything2*, Lampe *et al*. (2010) constataram que os fatores sociais e cognitivas pareciam ser mais importante na previsão de contribuições para o site do que questões de usabilidade. Da mesma forma, Ridings e Gefen (2004) encontraram aspectos sociais como uma das razões pelas quais os indivíduos se unem a comunidades online. Portanto, à primeira vista, comunidades online e os ecossistemas de *software* parecem ser influenciados pelos mesmos aspectos: aspectos sociais.

Neste capítulo foram discutidos os resultados encontrados neste trabalho com estudos semelhantes presentes na literatura. No próximo capítulo serão apresentadas as conclusões finais do trabalho e também serão apresentadas sugestões de trabalhos futuros.

6. CONCLUSÕES E TRABALHOS FUTUROS

6.1. Conclusões

Este trabalho descreveu um estudo qualitativo que visou entender fatores que levam um engenheiro de *software* a escolher determinado ecossistema de *software* móvel. Foram analisados os principais fatores que estão relacionados a adoção, rejeição e até mesmo a permanência em determinado ecossistema de *software* móvel. Por exemplo, identificou-se que um dos principais desafios a ser tratado vem a ser a grande diversidade de aparelhos *smartphones*, que vão desde os modelos mais simples baratos até os modelos mais caros e com diversas funcionalidades. Os fatores identificados foram comparandos com outros estudos presentes na literatura. O que foi possível perceber é que a abordagem e a metodologia utilizadas neste estudo se assemelham muito com outros estudos presentes na literatura, além de que a escolha da Teoria Fundamentada em Dados foi acertada, visto que vários estudos que aliam esta teoria e a Engenharia de *Software* foram encontrados é usada desde 1988 no estudo de Bayer e Melone. Este último capítulo traz primeiramente as considerações finais sobre este trabalho, descrevendo depois as contribuições desta pesquisa e os trabalhos futuros. Especificamente, este trabalho apresentou resultados de cunho técnico e social. Ambas as conclusões finais sobre os mesmos sãoapresentadas a seguir.

6.1.1. Conclusões Técnicas

Pode-se chegar a algumas conclusões com cunho técnico ou tecnológico neste trabalho. Primeiro, a fragmentação dos ecossistemas de *software* é um fator que incomoda os engenheiros de *software*, pois com tantos dispositivos no mercado e diferentes versões de sistemas operacionais, estes engenheiros os mesmos precisam se preocupar com uma ampla gama de dispositivos na hora do desenvolvimento de aplicativos e isso é algo que pode ser resolvido através de atualizações periódicas dos sistemas operacionais, mantendo o mesmo sistema para os dispositivos do mesmo ecossistema de *software*. Outro ponto que incomoda os desenvolvedores são os emuladores utilizados para testar os aplicativos, como relatado nos resultados, os mesmos ainda são atrasados e lentos, o que impede o desenvolvimento mais ágil de aplicativos. A melhora destes emuladores poderia trazer mais agilidade para o trabalho dos engenheiros de *software*.

Estes problemas podem influenciar os engenheiros a não adotar um determinado ecossistema de *software* se o mesmo possuir algum desses problemas ou vice-versa, se o ecossistema não tiver estes problemas, estes podem ser fatores que influenciem um engenheiro a adotar um determinado ecossistema.

6.1.2. Conclusões Sociais

O ser humano é uma criatura social, portanto precisa de bons meios de comunicação para desempenhar seus afazeres diários. Nesta pesquisa, nota-se que engenheiros que têm algum tipo de contato com profissionais dentro das organizações que são donas dos ecossistemas, se sentem mais motivados a não só adotar um ecossistema, como também permanecer no mesmo. Esta transparência e acessibilidade se mostraram fatores importante para a adoção de um ecossistema.

Também se notou neste estudo que os engenheiros gostam de participar de grupos de discussão sobre os ecossistemas em que estão inseridos, eles se sentem satisfeitos em notar que existem pessoas que podem ajuda-los em seus problemas e também se sentem satisfeitos em poder ajudar pessoas com suas dúvidas. Logo, as redes sociais são os meios de comunicação mais utilizados pelos engenheiros para se comunicar, expandindo as fronteiras dos engenheiros de *software*.

6.2. Contribuições do Trabalho

Esta dissertação teve como foco analisar as motivações e critérios de decisão que influenciam engenheiros de *software* a adotar um ecossistema de *software* móvel, para que este objetivo fosse alcançado, foram entrevistados engenheiros de *software* novatos e com experiência. O trabalho contribui dois âmbitos diferentes: academia e mercado.

A Seção 1.1.1 apresenta a motivação acadêmica para a condução deste estudo, então ele se torna útil para a academia devido ao fato de que não só fatores técnicos foram apresentados e explorados neste estudo, mas diversos fatores sociais também foram apresentados e discutidos, contribuindo assim para o crescimento do conhecimento sobre ecossistemas de *software* e possíveis fatores que fazem um ecossistema ser bem sucedido ou mal sucedido, além de obviamente ajudar a entender as motivações que levam a um engenheiro de *software* adotar ou rejeitar determinado ecossistema de *software*.

No que diz respeito ao mercado, a seção 1.1.2 mostra a motivação industrial para a condução deste estudo. Com o grande crescimento do mercado de *smartphones* as empresas precisam fortalecer seus ecossistemas de *software*, e para tanto, é necessário entender os motivos que levam um engenheiro de *software* adotar seu ecossistema para desenvolver aplicativos. Além de tudo, tais empresas devem pensar na forma de fidelizar tais engenheiros, além de sempre procurar atrair novos engenheiros, para que o seu ecossistema continue crescendo. Este estudo pode auxiliar tais empresas a atingir os

objetivos já citados, podendo fortalecer seus ecossistemas e procurarem sempre satisfazer os engenheiros que contribuem com aplicativos para o seu ecossistema.

Os resultados relatados nesta dissertação também podem servir para as empresas governantes dos ecossistemas planejarem melhor seus passos e conseguirem aumentar sua base de engenheiros de *software* desenvolvendo para o seu ecossistema.

6.3. Trabalhos Futuros

Como trabalhos futuros é interessante realizar mais entrevistas com desenvolvedores de todos os ecossistemas de *software* móveis existentes no mercado para adquirir perspectivas diferentes de seus motivos para adotarem os ecossistemas que não foram explorados por esta pesquisa. Também se torna interessante refazer todas as entrevistas com as mesmas pessoas e confirmar se os resultados relatados por eles continuam os mesmos e realizar uma comparação entre a opinião, principalmente dos desenvolvedores com menos de um ano de experiência, pois com a maturidade os problemas que os fazem permanecer em um determinado ecossistema podem mudar, ou até mesmo podem fazê-los trocar de ecossistema – sem contar que os que podem largar completamente o desenvolvimento para ecossistemas de *software* móveis.

Um possível trabalho futuro também pode ser a criação de um *framework* que guie os engenheiros de *software* para a escolha do ecossistema que melhor atenda as suas necessidades. Para isso além de novas entrevistas, um processo etnográfico aonde o pesquisador vá para empresas observar como se dá o processo de desenvolvimento de aplicações para diferentes ecossistemas, e como eles diferem entre si, pode ajudar na construção de um *framework*, além de possivelmente estudos de casos que validem a eficácia deste *framework*.

Outro estudo interessante seria procurar uma forma de mapear o pilar tempo da Teoria de Difusão da Inovação, visto que neste estudo apresentado nesta dissertação este pilar foi ignorado, pois não foi encontrada uma forma de mapeá-lo.

REFERÊNCIAS BIBLIOGRÁFICAS

Banks, S.; Louie, E.; Einerson, M. **Constructing Personal Identities in Holiday Letters**. Journal of Social and Personal Relationships, Vol. 17(3): 299–327, (2000).

Barbosa, O. & Alves, C. **A systematic mapping study on software ecosystems.** In Proceedings of the 3rd Workshop on Software Ecosystems. (2011).

Brinkkemper , S.; van Soest, I., & Jansen, S. **Modeling of product software businesses: Investigation into industry product and channel typologies**. In proceedings of the Sixteenth International Conference on Information Systems Development, pages 677–686. Springer-verlag. (2007).

Basili, V.; Shull, F. and F. Lanubile, **Building Knowledge through Families of Experiments**. IEEE Transactions on Software Engineering, 25, n. 4 (July), 456-473. (1999).

Bayer, J. & Melone, N. **A critique of diffusion theory as a managerial framework for understanding adoption of software engineering innovations.** In: System Sciences, 1988. Vol.II. Software Track, Proceedings of the Twenty-First Annual Hawaii International Conference. Kailua-Kona, HI, USA. (1988).

Biffl, S.; Aurum, A.; Boehm, B.; Erdogmus, H.; and P. Grünbacher, **Value-Based Software Engineering**. Springer-Verlag. (2006).

Bosch, J. and Petra, M. **Software product lines, global development and ecosystems: Collaboration in software engineering**. Collaborative Software Engineering, Edited by Ivan Mistrk, Andr van der Hoek, John Grundy, and Jim & Springer Berlin Heidelberg. (2010).

Boulding, K.: **General Systems Theory-the Skeleton of Science**. Management Science, 2 197-208. (1956).

Butle, M. **Android: Changing the Mobile Landsscape.** In: Pervasive Computing, IEEE (2011).

Campbell, P.R.J. & Ahmed, F., **A Three-Dimensional View of Software Ecosystems**. In: 2nd International Workshop on Software Ecosystems. (2010).

Charmaz, K. **Constructing Grounded Theory. A Practical Guide Through Qualitative Analysis**. London: Sage Publications. (2006).

Chitungo, S. K. & Munongo, S. **Extending the Technology Acceptance Model to Mobile Banking Adoption in Rural Zimbabwe**. Journal of Business Administration and Education, Vol. (3)1, pp. 51-79. (2013).

Ciolkowski, M.; Heidrich, J.; Münch, J,; Simon, F. & Radicke, M. **Evaluating Software Project Control Centers in Industrial Environments.** In: First International Symposium on Empirical Software Engineering and Measurement. Spain. (2007).

Christakis, N. & Fowler, J. **Connected: The Surprising Power of Our Social Networks and How They Shape Our Lives -- How Your Friends' Friends' Friends Affect Everything You Feel, Think, and Do.** In: Little Brown and Company. (2009)

Cusumano, M. **The Business of Software: What Every Manager, Programmer and Entrepreneur Must Know to Succeed in Good Times and Bad.** In: Free Press, New York. (2004).

Davis, Fred D. **Perceived usefulness, perceived ease of use and user acepptance of information technology**. Mis Quarterly, v. 13, n. 3, p. 319-340, (1989).

Dennis, A. R.; S. K. Pootheri, and V. L. Natarajan **'Lessons from the Early Adopters of Web Groupware'**. J. Manage. Inf. Syst., vol. 14, no. 4, pp. 65–86. (1998)

Dewalt, K. & Dewalt, B. **Participant Observation - A Guide for Fieldworkers**. AltaMira Press, Walnut Creek, CA, 2002.

Dhungana, D.; Groher, I.; Schludermann, E. & Bi, S. **Software ecosystems vs. natural ecosystems: learning from the ingenious mind of nature**. In Proceedings of the Fourth European Conference on Software Architecture: Companion Volume, pages 96-102.ACM. (2010).

Fanning, R. M. & Gaba, D. M. **The Role of Debriefing in Simulation-Based Learning** In: Society for Simulation in Healthcare. (2007).

Ferreira, R.; Miranda, M.; de Souza, C. R. B.; Figueira Filho, F.; Treude, C. & Singer, L. **Os Aspectos Sociais dos Ecossistemas de Software.** In: Simpósio Brasileiro de Sistemas Colaborativos. (2015)

Forte, A.; Kittur, N.; Larco, V.; Zhu, H.; Bruckman, A.; & Kraut, R. E. **Coordination and Beyond: Social Functions of Groups in Open Content Production**. In: Proc. of the Conf. on Computer Supported Cooperative Work. pp. 417–426. (2012).

Gawer, A. & Cusumano, M. A. **Platform Leadership**. Harvard Business School Press, Boston, MA, USA, (2002).

Goadrich, M. H. & Rogers, M P. **Smart smartphone development: iOS versus android**. In: Proceedings of the 42nd ACM technical symposium on Computer science education. (2011).

Gómez, C.; Cleary, B. & Singer, L. **A study of innovation diffusion through link sharing on stack overflow** In. Proceedings of the 10th Working Conference on Mining Software Repositories. (2013).

Hylén, J. **Open Educational Resources: Opportunities and Challenges**. OECD's Centre for Educational Research and Innovation. (2006).

Iansiti, M. & Levien, R. **Strategy as ecology**. In: Harvard BusinessReview. (2004).

Jansen, S. & Finkelstein, A. and Brinkkemper, S. **A Sense of Community: A Research Agenda for Software Ecosystems**, In: International Conference on Software Engineering, Vancouver, Canada, IEEE CS Press. (2009).

Jansen, S.; Finkelstein, A. & Brinkkemper, S. **Providing transparency in the business of software: A modelling technique for software supply networks**. In Proceedings of the 8th IFIP Working Conference on Virtual Enterprises, pages 677–686, (2007).

Joorabchi, M. E.; Mesbah, A. & Kruchten, P. **Real challenges in mobile app development**. In: Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement. (2013).

Keung, J.; Jeffery, R. & Kitchenham, B. **The Challenge of Introducing a New Software Cost Estimation Technology into a Small Software Organisation.** In: Australian Software Engineering Conference. (2004).

Lampe, C.; Wash, R.; Velasquez, A & Ozkaya, E.. **Motivations to Participate in Online Communities**. In: Proc. of the Conf. on Human Factors in Computing Systems. pp. 1927–1936. (2010)

Larsen, E. A. & Kautz, K. **Quality assurance and software process improvement in Norway.** In: Software Process: Improvement and Practice, Volume 3, Issue 2. (1997).

Linares-Vásquez, M.; Bavota, G.; Bernal-Cardenás, C.; Di Penta, M.; Oliveto & R. Poshyvanyk, D. **API change and fault proneness: a threat to the success of Android apps.** In: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. (2013)

Maxwell, E.: **Open Standards, Open Source, Open Innovation: Harnessing the Benefits of Openness**. Innovations, 119-176. (2006).

Messerschmitt, D. G., Szyperski, C. S**oftware Ecosystem: Understanding an Indispensable Technology and Industry.** The MIT Press. (2003).

Miranda, M.; Ferreira**,** R.; de Souza, C. R. B.; Figueira Filho, F. & Singer, L. **An exploratory study of the adoption of mobile development platforms by software engineers**. In: the 1st International Conference, 2014, Hyderabad. Proceedings of the 1st International Conference on Mobile Software Engineering and Systems - MOBILESoft 2014. New York: ACM Press, 2014. p. 50-53.

Moore, J. F. **The death of competition: Leadership and strategy in the age of business ecosystems**. HarperBusiness, New York. (1996).

Neves, J. L. **Pesquisa Qualitativa – Caracteristicas, Usos e Possibilidades**, In: Caderno de Pesquisas em Administração, São Paulo, V.1, N° 3, 2° SEM. (1996).

Perez G., Zwicker, R., Zilber, M. A. and Júnior, A, M. **Adoption of technological innovations in the field of health: a study on information systems in the perspective of the theory of diffusion**. In: Journal of Information Systems and Technology Management. (2010).

Phan, K. & Daim, T. **Exploring technology acceptance for mobile services.** Journal of Industrial Engineering and Management 4(2): 339-360. (2011).

Preece, J. **Sociability and usability in online communities: determining and measuring success**. Behaviour & IT, vol. 20, no. 5, pp. 347–356. (2001)

Ridings, C. M. & Gefen, D. **Virtual Community Attraction: Why People Hang Out On-line**. J. Computer-Mediated Communication, vol. 10, no. 1. (2004).

Rogers, E. M. **Diffusion of innovations**. 5th ed. New York: Free Press, (2003).

Santos, R.P., & Werner, C.M.L.. **A Proposal for Software Ecosystems Engineering**. In: *Proceedings of the 3rd International Workshop on Software Ecosystems, 2nd International Conference on Software Business,* Brussels, pages 40-51, June. (2011a).

Santos, R.P. & Werner, C.M.L.. **Treating Business Dimension in Software Ecoystems**. In: *Proceedings of the 3rd ACM/IFIP International Conference of Management of Emergent Digital EcoSystems*, San Francisco, pages 197-201, November. (2011b).

Seaman, C. **Qualitative Methods**. In: Guide to Advanced Empirical Software Engineering. Londres:Springer-Verlag. (2008).

Singer, J., Sim, S., Lethbridge, T. **Software Engineering Data Collection for Field Studies**. In: Guide to Advanced Empirical Software Engineering. Londres: Springer-Verlag, p. 9-34. (2008).

Sproull, L. & Faraj, S.: **'Atheism, Sex, and Databases: The Net As a Social Technology'**. In: B. Kahin and J. Keller (eds.): Public Access to the Internet. MIT Press, pp. 62–81. (1995).

Strauss, A. & Corbin, J. "**Pesquisa Qualitativa: Técnicas e Procedimentos para o Desenvolvimento de Teoria Fundamentada**". In: Artmed. (2008).

te Molder. J.; van Lier, B.; Jansen, S: **Clopenness of Systems: The Interwoven Nature of Ecosystems.** Proceedings of the Workshop on Software Ecosystems, 52-64. (2011).

Vieira, S.R.C; Viana, D. ; do Nascimento, R. & Conte, T. **Evaluating a Technique for Requirements Extraction from Business Process Diagrams through Empirical Studies.** In: XXXVIII Conferencia Latinoamericana En Informatica (CLEI), Medellin, Colombia, 2012.

VisionMobile, **Developer Economics Q3 2013**. Developer Economics. (2013).

von Bertalanffy, L.: **The Theory of Open Systems in Physics and Biology.** Science, 111 23-29. (1950).

Woo, F.; Mikusauskas, R. ; Bartlett, D. ; Law, R. **Is OO the Systems Development Technology for Your Organization?** In: Fourth International Conference on Software Engineering, Research, Management and Applications. Seattle, Washington, USA. (2006)

APENDICE 1

Roteiro das Entrevistas

01°. Qual seu nome, idade e formação ?

02°. Para quais plataformas (iOS, Android, Mozilla OS, etc.) você desenvolve ? Há quanto tempo ?

03°. Como você conheceu a plataforma ?

04°. Por que você escolheu esta plataforma ?

05°. O quão simples ou complicado é a utilização da plataforma ?

06°. Quais são os recursos necessários para desenvolver *aplicações* para a plataforma ? Por exemplo, equipamentos específico, conhecimento de alguma linguagem específica ?

07°. Aproximadamente, quanto tempo você levou ou leva para desenvolver *aplicações para a plataforma ?*

08. Foi fácil ou difícil para você realmente começar a desenvolver para a plataforma, em outras palavras, como foi a curva de aprendizado ?

09°. O que você faz para que suas aplicações sejam conhecidas ?

10°. Você participa de qualquer comunidade que trate de questões que estão relacionadas a plataforma ? Que tipo de comunidade é essa ?

11. Você contribui para a comunidade ? Por exemplo, respondendo perguntas de outros desenvolvedores ? Se sim, com que frequência ?

12°. Quanto você acredita que a participação nessas comunidades ajudou ou ajuda a desenvolver aplicações ? Qual é o tempo de resposta na comunidade para esclarecer dúvidas ?

APENDICE 2

Debriefing Questions

Location:

Date:

Observer:

1. What were the key points observed about the focus ?

2. What did you find to be most surprising about this observation ?

3. What did you see or hear that was pretty much what you expected (Or like other sites that you have seen) ?

4. What did you learn about the problem and “fixes” that you didn’t know before ? That you did ?

5. What would you ask if we could go back ? Would you ask the next participant this as well ?

6. What worked really well ?

7. What didn’t work so well or what should be changed ?

8. Other comments ?