UNIVERSIDADE FEDERAL DO PAMPA

Alex Severo Chervenski

Entendimento sobre Sistemas Legados à luz da Teoria Fundamentada em Dados

Alegrete
2019

Alex Severo Chervenski

Entendimento sobre Sistemas Legados à luz da Teoria Fundamentada em Dados

Trabalho de Conclusão de Curso
apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito para a obtenção do título de Bacharel em Engenharia de Software.

Orientador: Prof. Drª. Andréa Sabedra Bordin

Alegrete
2019

Alex Severo Chervenski

Entendimento sobre Sistemas Legados à luz da Teoria Fundamentada em Dados

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Software da Universidade Federal do Pampa como requisito para a obtenção do título de Bacharel em Engenharia de Software.

Trabalho de Conclusão de Curso defendido e aprovado em 29 de Novembro de 2019
Banca examinadora:

André F Bordin

Prof. Drª. Andréa Sabedra Bordin
UNIPAMPA

Prof. Drª. Amanda Meincke Melo
UNIPAMPA

Prof. Dr. João Pablo Silva da Silva
UNIPAMPA

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter me dado conhecimento e sabedoria para alcançar meus objetivos.

Aos meus familiares, especialmente à minha mãe Elisa e ao meu pai Luiz, que estavam sempre presentes quando mais precisei de carinho, incentivo e auxílio. Agradeço por estarem 24 horas por dia torcendo para meu sucesso.

Agradeço ao meu padrinho Gregson, que nunca mediu esforços para ajudar à minha família em todos os momentos que necessitamos, e principalmente por ser um padrinho que me tratou sempre como um filho.

A minha orientadora, professora Andréa Sabedra Bordin pela dedicação e suporte no desenvolvimento deste trabalho, com certeza não poderia ter sido melhor orientado.

Agradeço ao colega Mauricio Escobar, por ter desenvolvido uma ferramenta que auxiliou no alcance dos objetivos deste trabalho. Seu apoio foi primordial.

Aos meus colegas e amigos(as) da famosa '303', que junto comigo tiveram dias de preocupações, desesperos e alegrias.

Especialmente aos colegas e amigos Jonnathan, Lukas, Marcus, Naihara, Luana, Romário, Paula, Karina, Murilo, João, Adriel, Amanda, André e Bruno, que dedicaram tempo para me auxiliar no desenvolvimento deste trabalho.

RESUMO

Sistemas computacionais são desenvolvidos há um longo tempo com a intenção de atenderem aos objetivos de organizações. Porém, com o passar do tempo, as organizações tendem a mudar seus objetivos por variados motivos, fazendo com que esses sistemas passem por ciclos de manutenções para continuarem a ser úteis. Esses sistemas, que tendem a se degradar e causar problemas de manutenibilidade, normalmente são referidos como sistemas legados. No entanto, observa-se que na literatura existe um entendimento diversificado sobre o que torna um sistema legado, quais os problemas e quais as possíveis estratégias de evolução. Assim sendo, este trabalho teve o propósito de oferecer um entendimento mais conciso sobre esses elementos relacionados a sistemas legados. Para atingir esse objetivo, foram coletados dados da literatura e estes foram analisados de acordo com procedimentos da Teoria Fundamentada em Dados (*Grounded Theory-GT*), resultando em um modelo visual e textual que auxilia nesse entendimento mais conciso. A fase de codificação aberta da GT, foi realizada de forma colaborativa com o intuito de obter dados de forma consensual, confiável e celere, com um grupo de participantes que criou os códigos e dois especialistas avaliaram os resultados. Os resultados, explicitados através de um modelo visual e textual, mostram que elementos que denotam o desenvolvimento ou uso de tecnologias obsoletas, principalmente em relação à linguagens de programação, são definidores de um sistema ser considerado legado, e que legados ocasionam problemas de cunho financeiro, principalmente devido ao fato de necessitarem de vários ciclos de manutenções para continarem ativos e operacionais. Observou-se, ainda, que a migração de legados é a estratégia mais utilizada atualmente para evoluir tais sistemas, sendo especificamente a migração de legados para a plataforma em nuvem a mais encontrada. Assim, este estudo serviu de apoio para um melhor entendimento na identificação desses elementos que respondem se um sistema já é ou está se tornando legado, bem como as possíveis soluções para evolui-los.

Palavras-chave: Sistemas Legados. Teoria Fundamentada em Dados. *Grounded Theory*. Evolução de Software. Manutenção de Software.

ABSTRACT

Computer systems have been developed for a long time with the intention of meeting the goals of organizations. But over time, organizations tend to change their goals for a variety of reasons, causing these systems to go through maintenance cycles to continue to be useful. These systems, which tend to degrade and cause maintainability problems, are commonly referred to as legacy systems. However, it is observed that in the literature there is a diverse understanding of what makes a legacy system, what are the problems and what the possible strategies for evolution. Therefore, this paper was intended to provide a more concise understanding of these elements related to legacy systems. To achieve this goal, data were collected from the literature and analyzed according to Grounded Theory (GT) procedures, resulting in a visual and textual model that assists in this more concise understanding. GT's open coding phase was collaboratively conducted to obtain data in a consensual, reliable and timely manner with a group of participants who created the codes and two experts evaluated the results. The results, explained through a visual and textual model, show that elements that denote the development or use of obsolete technologies, especially in relation to programming languages, define a system to be considered legacy, and that legacies cause financial problems. , mainly due to the fact that they require several maintenance cycles to remain active and operational. It was also noted that legacy migration is the most widely used strategy today to evolve such systems, with legacy migration to the cloud platform being the most commonly found. Thus, this study supported a better understanding in identifying these elements that respond if a system is already or is becoming legacy, as well as the possible solutions to evolve them.

Key-words: Legacy Systems. Grounded Theory. Evolution of Software. Software Maintenance.

LISTA DE FIGURAS

Figura 1 – Exemplo de avaliação de sistema legado. 28
Figura 2 – Evolução de Pesquisas relacionadas sobre Grounded Theory (GT) em ES. 47
Figura 3 – Processo dos procedimentos metodológicos 50
Figura 4 – Processo Coleta e organização de dados 50
Figura 5 – Processo de Análise de Dados 54
Figura 6 – Processo de Codificação Aberta 54
Figura 7 – SubProcesso Identificar Códigos 55
Figura 8 – Processo Codificação Axial 57
Figura 9 – Número estudos pós triagem dos dados 59
Figura 10 – Vídeo explicativo dos procedimentos de codificação. 62
Figura 11 – Imagem do acesso à aplicação. 62
Figura 12 – Imagem dos trechos textuais. 63
Figura 13 – Criação de código. 63
Figura 14 – Salvar código. 64
Figura 15 – Tela inicial avaliador. 65
Figura 16 – Selecionar grupo e trecho textual. 65
Figura 17 – Selecionar códigos finais. 66
Figura 18 – Tabela contendo trecho textual e códigos criados. 66
Figura 19 – Resultado codificação colaborativa. 67
Figura 20 – Atividade de Agrupamento de códigos 68

LISTA DE TABELAS

Tabela 1 – Exemplos de definições de sistemas legados 26
Tabela 2 – Tabela Critério de Inclusão (CI) e Critério de Exclusão (CE): Estado da Arte. 40
Tabela 3 – Número de publicações retornadas. 40
Tabela 4 – Trabalhos relevantes ao estado da arte. - **(Continua)** 41
Tabela 5 – Trabalhos relevantes ao estado da arte. - **(Continua)** 42
Tabela 6 – Trabalhos relevantes ao estado da arte. - **(Continua)** 43
Tabela 7 – Trabalhos relevantes ao estado da arte. 44
Tabela 8 – Disciplinas/Áreas da ES mais Frequentes utilizando GT. 45
Tabela 9 – Etapas da GT alcançadas nos estudos. 46
Tabela 10 – Técnicas utilizadas nos estudos que utilizam GT. 46
Tabela 11 – String de Busca Genérica. 52
Tabela 12 – Tabela com CI e CE 53
Tabela 13 – Quantidade de trabalhos retornados nas bases de dados 58
Tabela 14 – Exemplos trechos extraídos. 60
Tabela 15 – Relação de grupos para codificação colaborativa. 61

LISTA DE SIGLAS E ABREVIATURAS

CE Critério de Exclusão

CI Critério de Inclusão

ES Engenharia de Software

GT Grounded Theory

MSL Mapeamento Sistemático na Literatura

QP Questão de Pesquisa

RSL Revisão Sistemática na Literatura

SUMÁRIO

1 INTRODUÇÃO 19
1.1 Objetivo Geral 22
1.2 Objetivos Específicos 22
1.3 Justificativa 22
1.4 Organização do Documento 23

2 FUNDAMENTAÇÃO TEÓRICA 25
2.1 Sistemas Legados 25
2.2 Avaliação de Sistemas Legados 27
2.3 Teoria Fundamentada em dados (Grounded Theory - GT) . . 30
2.3.1 Uso da Literatura na GT 36

3 ESTADO DA ARTE 39
3.1 Questão de Pesquisa 39
3.2 Base de Dados e String de Busca 39
3.3 Resultados Obtidos 40
3.4 Discussão dos Resultados 45

4 METODOLOGIA 49
4.1 Processo Metodológico 50
4.1.1 Coleta e organização de dados 50
4.1.1.1 Definir questões de pesquisa 50
4.1.1.2 Conduzir a busca 51
4.1.1.3 Bases de dados 51
4.1.1.4 Strings de busca 51
4.1.1.5 Triagem dos dados 52
4.1.1.6 Extração dos dados 53
4.1.2 Análise de dados 53
4.1.2.1 Codificação aberta 54
4.1.2.2 Identificar Códigos 55
4.1.2.3 Realizar codificação colaborativa 56
4.1.2.4 Identificar agrupamentos 56
4.1.2.5 Codificação axial 56
4.1.2.6 Identificar tipos de código 57
4.1.2.7 Identificação de relações entre os agrupamentos de conceitos . . . 57
4.1.3 Ferramentas metodológicas 57
4.2 Execução do Processo 58
4.2.1 Coleta e organização dos dados 58
4.2.1.1 Extração dos dados 59

4.2.2 Análise de dados . . . 60
4.2.3 Identificar Códigos . . . 60
4.2.3.1 Seleção de Participantes . . . 60
4.2.3.2 Convite aos Participantes . . . 61
4.2.3.3 Separação dos Participantes em Grupos . . . 61
4.2.3.4 Elaboração de Material Explicativo . . . 61
4.2.3.5 Realizar Codificação Colaborativa . . . 62
4.2.4 Identificar agrupamentos . . . 67
4.2.4.1 Grupo Definições . . . 68
4.2.4.2 Grupo Problemas . . . 69
4.2.4.3 Grupo Estratégias . . . 70
4.2.4.4 Identificar tipos de código . . . 71
4.2.4.5 Identificação de relações entre os agrupamentos de conceitos . . . 71
4.3 Ameaças à validade . . . 72

5 RESULTADOS . . . 73
5.1 Apresentação do modelo visual de definições . . . 73
5.2 Apresentação do modelo visual de problemas . . . 75
5.3 Apresentação do modelo visual de estratégias . . . 77
5.4 Apresentação do modelo visual geral . . . 79
5.5 Teoria Geral . . . 81

6 CONSIDERAÇÕES FINAIS . . . 83
6.1 Trabalhos Futuros . . . 84

REFERÊNCIAS . . . 85

Índice . . . 93

1 INTRODUÇÃO

Inúmeras soluções computacionais (*e.g.* softwares, aplicativos) são construídas há algumas décadas. Muito do crescimento da construção dessas soluções ocorre pela necessidade de automatizar processos para fins variados (*e.g.* sistemas bancários, sistemas de postos de combustíveis, etc.). Muitos processos necessitam ser informatizados nas organizações pois existe uma necessidade por aplicações que satisfaçam e economizem tempo em tarefas antes realizadas de forma manual e mais demorada.

Muitas das soluções construídas há longo tempo não foram criadas com a finalidade de que durassem longos períodos de tempo, porém ainda estão em operação, por serem de suma importância em suas organizações. Esses sistemas normalmente reconhecidos como legados à organização, necessitam de manutenções para contemplarem novas necessidades organizacionais.

No entanto, esses sistemas foram criados com tecnologias mais antigas, essas que na grande parte das vezes não são mais utilizadas atualmente, causando um transtorno para os programadores (mantenedores) de sistemas computacionais. Mesmo que manutenções em alguns sistemas sejam factíveis, sejam elas frequentes ou não, elas acabam tornando o sistema em si cada vez mais degradado, envelhecido e consequentemente difícil de manter. Uma analogia simples para esse processo é pegar uma vestimenta que gostamos e utilizamos muito, ou seja, cada vez que a vestimos ao longo do tempo necessitamos lavá-la e cada vez que a lavamos esta fica mais desbotada, o tecido fica mais sensível, causando com o tempo um desuso dessa peça.

Devido à dependência das organizações em seus sistemas computacionais, a identificação de condições causadoras de um sistema se tornar legado, os problemas que causam ou podem causar às organizações, bem como as possíveis estratégias de evolução para esse tipo de sistema são importantes, de forma que efeitos negativos sejam mitigados precocemente. Na literatura, buscaram-se por definições de sistemas legados, no entanto não foi encontrado um consenso através do qual seja possível determinar quais características um sistema deve possuir para ser considerado legado, ou seja, existem várias definições, o que causa dificuldade de se reconhecer se um sistema é ou está se tornando legado dentro de uma organização.

Exemplificando a problemática existente em pontuar características de um sistema legado, um estudo realizado por Martins, Chervenski e Bordin (2017) relatou a discrepância existente na área. Para chegar a essa conclusão, foi realizado um mapeamento sistemático na literatura a fim de coletar definições de sistemas legados onde os dados foram analisados através da técnica denominada Análise de Conteúdo. Obteve-se um resultado de 125 definições capturadas entre os anos de 1995 à 2015 (com um intervalo de 5 em 5 anos), a partir das quais foram extraídas 62 características distintas.

Existem definições que destacam o tempo que um sistema está em operação como um fator determinante para definir se o sistema é legado, tais como: “O software legado

é definido como um sistema de software de 10 a 15 anos de idade, que ainda é executável, mas resiste a modificações que causam a queda do software" Handigund e Kulkarni (2010), "sistemas legados podem ser quaisquer sistemas que, independente da idade, são úteis e utilizados atualmente" JURIC, Rozman e Hericko (2000). Esses exemplos demonstram as diferenças existentes entre as definições de legados que abordam aspectos temporais.

Além de características que destacam o aspecto temporal, existem outras características que abordam aspectos mais organizacionais, como: "sistemas legados são a espinha dorsal das empresas" Visaggio (2001), que denota a importância que sistemas legados possuem em suas organizações. Existem definições que abordam a dificuldade no entendimento do legado, considerado aspecto organizacional, como: "os sistemas legados se tornam mais difíceis de entender e manter após muitos anos de evolução" (ZOU, 2005).

Definições que explicitam o valor econômico que os legados representam para as organizações, como a de Visaggio (2001) que diz que "Os sistemas legados são a espinha dorsal das empresas e muitas vezes considerados como um ativo organizacional com um alto valor econômico" e a de Jatain e Gaur (2015) que afirma que "Apesar das limitações, a importância dos sistemas legados não pode ser ignorada, porque algumas de suas funções são muito valiosas para serem removidas e caras de se reproduzir" são alguns exemplos que exaltam os aspectos financeiros de um legado.

Definições que abordam aspectos relacionados a tecnologias obsoletas são encontradas, como: "esses sistemas geralmente são chamados de sistemas legados porque são baseados em, ou são compostos de processos, metodologias, tecnologias, peças e / ou aplicativos desatualizados (antigos) (SANDBORN; PRABHAKAR, 2015)". Ainda existem definições que exemplificam a linguagem utilizada em sistemas para caracterizá-los como legado, como: "Programas legados são programas escritos para um sistema de computador mainframe ou outro sistema de computador semelhante e escritos em uma linguagem de terceira geração anterior, como COBOL ou FORTRAN" (BOKKA, 2015).

Paralelamente existem definições que exaltam a resistência dos legados em relação a mudanças de qualquer origem, exaltando aspectos relacionados à manutenção de legados como: "sistemas legados resistem a quaisquer tipos de mudanças" (HANDIGUND; KULKARNI, 2010). Contrapondo essa definição existem relatos que legados têm evoluído, como: "sistemas legados têm evoluído com novas tecnologias emergentes" (GUO et al., 2005).

Pode-se observar que a existência e permanência de sistemas em organizações por longos períodos é de suma importância. O software legado é frequentemente um software crítico para os negócios, mantido porque é muito arriscado substituí-lo (ABDULAZIZ; EL-ABBASSY; HEGAZY, 2010), alterações de funcionalidades e até mesmo eventuais trocas destes sistemas, poderiam ocasionar a perda de dados insubstituíveis, o que traria uma situação caótica para os negócios. Sistemas legados são sistemas considerados obsoletos, mas que são cruciais para a operação de uma organização (BENNETT, 1995), trazendo

uma situação propícia para que empresas acabem se tornando "reféns" de seus sistemas. Um dos desafios mais difíceis de lidar por quem trabalha com sistemas legados, devido a inconsistência existente entre as definições, é identificar e determinar o "status de legado" de um sistema, pois não existem definições comumente aceitas (O'BYRNE; WU, 2000).

A maioria dos sistemas legados são muito caros para serem mantidos e também limitam o crescimento do desenvolvimento das organizações que os operam (RAJAVAT; TOKEKAR, 2014). Tendo em vista que tais sistemas costumam consumir muitos recursos financeiros para serem mantidos, processos de manutenções não podem ser negligenciados ou mal estruturados.

As definições encontradas no estudo de Martins, Chervenski e Bordin (2017) e o número de características distintas extraídas são exemplos da diversidade de variáveis envolvidas neste tema. No entanto, apenas extrair características e agrupá-las não é suficiente. Um estudo mais completo deve permitir um entendimento único e compartilhado sobre os elementos que compõem um sistema legado, apontando fatores causadores de um sistema se tornar legado, as consequências ou problemas desse tipo de sistema para as organizações, bem como as possíveis estratégias que podem ser adotadas em relação à evolução destes sistemas. Obter esse entendimento servirá, para que em tempo hábil, reflita-se sobre as maneiras de lidar e evitar problemas mais graves às organizações.

Entende-se que esse tipo de estudo se viabilize através de uma abordagem de pesquisa onde seja possível analisar as relações existentes entre esses elementos e que, complementarmente, o resultado seja materializado de uma forma visual, sustentada através dos dados coletados, onde a visualização seja nítida e clara para o entendimento desse tema.

A Teoria Fundamentada em Dados, do inglês *Grounded Theory* (GT) é um método utilizado que contempla as necessidades citadas anteriormente. Segundo STRAUSS e CORBIN (2008) a GT é um método de pesquisa qualitativa, onde a coleta de dados, análise e eventual teoria mantêm uma relação próxima entre si e, embora a finalidade desse método seja a construção de teorias, sua utilização não precisa necessariamente ficar restrita aos pesquisadores que têm esse objetivo de pesquisa. Ainda, para STRAUSS e CORBIN (1998) "o pesquisador pode usar alguns, mas não todos os procedimentos para satisfazer seus objetivos de pesquisa". Assim, a Teoria Fundamentada em Dados possibilita uma maneira de representar conceitos de conhecimento específico (sistemas legados) e as relações entre esses conceitos (condições causais, consequências, estratégias, etc).

Assim, tendo o trabalho de Martins, Chervenski e Bordin (2017) constatado a diversidade em elementos que envolvem um legado, como é possível estabelecer o entendimento sobre as características, bem como os problemas que apresentam e as possíveis estratégias de evolução de sistemas legados, através de procedimentos oriundos da Teoria Fundamentada em Dados?

1.1 Objetivo Geral

Dado o contexto, o objetivo deste trabalho é oferecer um entendimento mais conciso sobre os elementos que envolvem sistemas legados através de procedimentos da Teoria Fundamentada em Dados.

1.2 Objetivos Específicos

Para alcançar o objetivo geral, os seguintes objetivos são definidos:

- Identificar, na literatura científica, insumos para definir, identificar os problemas e as possiveis estratégias de evolução de sistemas legados;
- Desenvolver a análise dos dados extraídos para obter um consenso sobre os elementos que envolvem sistemas legados;
- Propor um modelo visual e textual para explicar o entendimento único, consensual e compartilhado sobre os elementos que envolvem sistemas legados.

1.3 Justificativa

Neste estudo, a GT foi utilizada como técnica de análise de dados para identificar, através de procedimentos de **codificação** aplicado a trechos de texto oriundos da literatura, elementos causadores, consequências, estratégias de evolução relacionadas a sistemas legados. Um importante papel da GT é sua utilização como um conjunto de procedimentos para servir como facilitador na análise de dados, que pode ser utilizado de uma forma flexível, para a construção de teorias bem fundamentadas.

Em relação a teorias, esse estudo possibilitou a geração de uma teoria bem fundamentada, apesar da amostra de dados contemplar apenas o ano de 2018. No entanto, outra contribuição desse estudo é o detalhamento de todo o processo de desenvolvimento da GT, visto que não foi possível encontrar na literatura um estudo que demonstrasse a aplicação dos procedimentos da mesma.

O desenvolvimento deste trabalho auxilia na elucidação dos elementos que envolvem um sistema legado e suas relações, o que por sua vez permitirá que se reconheça se um sistema é ou está se tornando legado, bem como os problemas que podem ser causados ou ja estão sendo causados e as possíveis alternativas de solução. A importância dos resultados deste trabalho reside na possibilidade de mitigar a tempo problemas com este tipo de sistema e que as melhores decisões sejam tomadas para possibilitar menores gastos com manutenções e evoluções ao longo do tempo.

1.4 Organização do Documento

No Capítulo 2, encontra-se a Fundamentação Teórica deste estudo, onde são abordados os temas sobre sistemas legados, avaliação de sistemas legados, teoria fundamentada em dados GT e o uso de literatura na GT.

No Capítulo 3 está o processo realizado para discutir sobre o estudo da arte relacionado à esta pesquisa.

No Capítulo 4 está discutido o andamento do trabalho, bem como os procedimentos metodológicos utilizados para obter os resultados obtidos neste trabalho.

No Capítulo 5 está discutido os resultados obtidos do trabalho.

No Capítulo 6 estão as considerações finais desta pesquisa, relatando o que foi possível analisar de benefícios do processo e dos resultados deste trabalho, complementarmente este capítulo traz os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste Capítulo, é abordada a base teórica deste trabalho, apresentando conceitos relacionados a sistemas legados, avaliação de sistemas legados e sobre a abordagem metodológica utilizada nesta pesquisa, denominada Teoria Fundamentada em Dados (do inglês *Grounded Theory*).

2.1 Sistemas Legados

Conforme exposto no Capítulo 1, sistemas computacionais são desenvolvidos há décadas. Esses sistemas tendem a durar por vários anos em suas organizações, pois foram construídos para atender aos objetivos de suas empresas. Porém, ao longo do tempo, as organizações tendem a mudar seus objetivos conforme algumas alterações, por exemplo em regras de negócio. Por sua vez, seus sistemas necessitam de adequações, causando assim a necessidade de existir vários ciclos de manutenção, para continuarem atendendo aos requisitos das suas organizações. Entretanto, segundo Bloomfield (2011) essas mudanças nos processos dentro de organizações podem causar problemas nos sistemas que devem obter renovações, fazendo que os mesmos possam não ter a capacidade de realizar tarefas necessárias atualmente.

Consequentemente, por esses sistemas terem sido desenvolvidos à um determinado tempo e por sofrerem excessos de manutenções ao longo de sua utilização, normalmente, são referidos como sistemas legados.

Entretanto, dentro da literatura não existe um consenso entre os autores sobre como definir um sistema como legado. Desde os promórdios na tentativa de caracterizar um legado, até os dias de hoje, existem inúmeras definições com diferentes características que devem ser levadas em consideração na hora de definir quando um sistema é legado, causando uma discrepância entre as diferentes definições desse âmbito. Diante do exposto, a Tabela 1 tem a finalidade de demonstrar (de maneira exemplificada) algumas definições ao longo dos anos sobre sistemas legados.

Tabela 1 – Exemplos de definições de sistemas legados.

Referência	**Definição**	**Ano**
(KHADKA et al., 2014)	Um sistema legado é um sistema de software que é crítico para as atividades de negócios do dia-a-dia e resiste a modificações	2014
(NILSSON, 2015)	"Os sistemas legados são sistemas de software que foram construídos há décadas com tecnologias e metodologias antigas. Atualmente, existem muitos desses sistemas on-line e as organizações acham que é difícil atualizá-los e mantê-los."	2015
(MALKI; BENSLIMANE; ABDELKADER, 2013)	"Legacy software are developed with outdated technologies such as C, C++, Cobol and maintained several times resulting in system with poor quality and obsolete or missing documentations but still doing a good job from the enterprise point of view."	2013
(BENNETT, 1995)	"Grandes sistemas de software que não sabemos como lidar, mas que são vitais para a nossa organização. "	1995
(BISBAL et al., 1999)	"Além disso, devido à sua idade avançada, os sistemas legados não possuem documentação adequada e tornam-se frágeis e sujeitos a erros ao longo do tempo."	1999

Fonte: O Autor.

Devido à frequência em problemáticas relatadas em legados, uma possível solução é construir um sistema novo e descartar o que já vem sendo usado, entretanto, existem algumas situações que não se viabiliza realizar essa mudança radical, são elas (BENNETT; WARD, 1995):

- Em sistemas que possuem anos de funcionamento na organização e suas tarefas não são replicadas em outros lugares, ou seja, os dados e as tarefas que a experiência de uso trouxe para esse sistema pelo tempo de funcionamento se perderia com a troca desse software;
- O processo manual que existia e foi substituído pelo sistema em vigência não existe mais, ou seja, a análise de uma construção nova teria que ser realizada de acordo com o sistema em uso;
- Um sistema novo pode obter um desempenho pior que o sistema que já vinha sendo utilizado, pelo menos nos primeiros dias de uso;
- Por um sistema legado possuir inúmeros usuários, os mesmos se "acostumam" a usar um sistema com interface "x", e a troca do mesmo pode ocasionar incômodos

aos usuários;

- Os usuários podem preferir uma abordagem evolucionária em vez de revolucionária.

Esses aspectos são corroborados por Warren (1999), que relata em seu trabalho que "um sistema legado, que é crítico para os negócios, deve permanecer operacional, de alguma forma, dentro de sua organização. No entanto, a manutenção contínua do sistema é dispendiosa e o escopo para implementar efetivamente mais mudanças é altamente restrito. Os custos de substituir o sistema do zero são proibitivos. Além disso, o conhecimento de negócios está embutido no sistema, há um risco significativo de perder esse conhecimento e desenvolver um sistema que não satisfaz seus requisitos".

Dado o contexto, simplesmente descartar sistemas que já são utilizados por um determinado tempo e com tais importâncias às suas organizações podem trazer situações prejudiciais. Uma substituição é indicada para sistemas que não conseguem mais se adaptar às necessidades do negócio, onde a modernização deste sistema não é mais factível a realidade do mesmo, seja qual for o fator (PINTO; BRAGA, 2004).

Porém, o possível crescimento em manutenções e evoluções de sistemas traz um certo receio, o trabalho de Seacord, Plakosh e Lewis (2003), por exemplo, relata que a modernização de um sistema é mais desafiadora que determinadas etapas anteoriores no ciclo de vida de software, o mesmo também expressa uma preocupação que o crescimento no desenvolvimento de novos sistemas seja maior que a capacidade de manter os mesmos.

Diante dessas constatações, uma avaliação de sistema legado deve ser realizada para decidir qual a melhor opção deve ser desenvolvida em relação a um sistema legado, pois a partir dessa avaliação é que se pode realizar estratégias de evoluções, melhorias ou adaptações no sistema.

2.2 Avaliação de Sistemas Legados

Como mencionado, o crescimento dos sistemas computacionais e as recorrentes manutenções, sejam de funcionalidades, sejam mudanças devido a requisitos da organização, fazem com que esses sistemas acabem cada vez mais degradados (PRESSMAN; MAXIM, 2006).

Entretanto, as próprias organizações enfrentam dificuldades em tomar decisões quanto ao futuro desses sistemas legados. Segundo Warren (1999) caso decisões sejam tomadas de maneira errônea as penalidades podem acarretar prejuízos financeiros grandes para a organização em questão.

Devido a limitações financeiras, organizações necessitam decidir quais melhores formas de obter retorno sobre os investimentos utilizados em seus sistemas legados, para isso, elas devem realizar uma avaliação realista de seus legados, para posteriormente decidirem qual a estratégia mais apropriada para a evolução dos seus sistemas (SOMMERVILLE, 2007).

Conforme o exposto, evoluções são inevitáveis durante o ciclo de vida dos softwares. Para que essas evoluções sejam feitas corretamente, necessita-se de avaliações adequadas. A avaliação do sistema deve ser uma atividade inicial para projetos de evolução (RANSOM; SOMMERVILLE; WARREN, 1998).

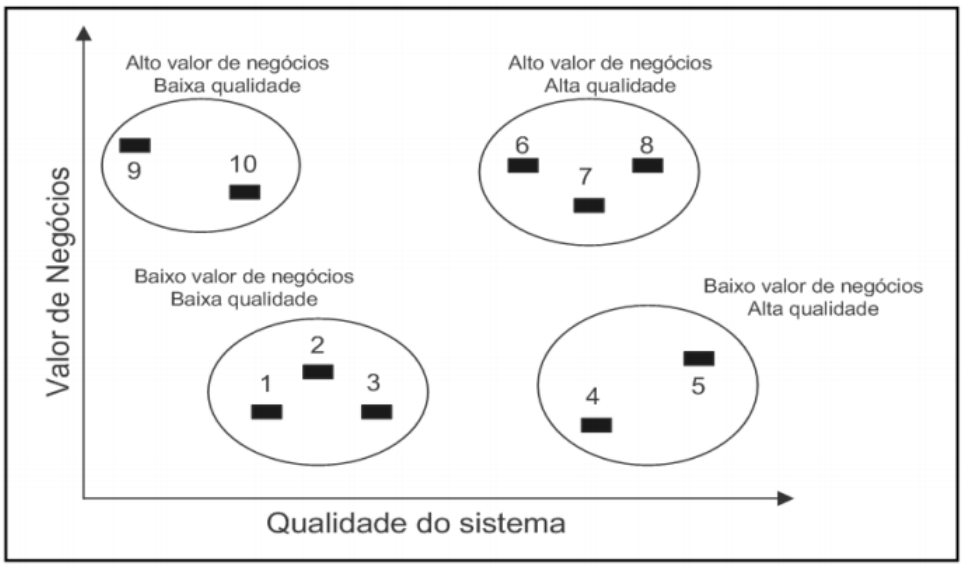
Um sistema legado possui várias maneiras de ser evoluído, porém, nenhuma dessas maneiras podem ser escolhidas de forma aleatória ou *ad hoc*. Para determinado sistema devem ser realizadas estratégias e estudos sobre que tipo de evolução se cabe à situação em questão.

Contudo, a avaliação de sistemas legados serve para encontrar a melhor estratégia de evolução, para isso deve ser analisada por parâmetros como, po exemplo, valor de negócio e a qualidade técnica do software. Do ponto de vista de mercado, é preciso avaliar se a empresa realmente precisa do sistema, sendo que, na perspectiva técnica, a avaliação compreende a qualidade do software da aplicação e o hardware que será utilizado como apoio do sistema (SOMMERVILLE, 2011).

Como um sistema legado pode evoluir de várias maneiras, existem fatores que devem ser levados em consideração de acordo com situações existentes como sua condição técnica, seu valor de negócio e as características das organizações envolvidas na manutenção e operação do sistema (RANSOM; SOMMERVILLE; WARREN, 1998).

No seu livro, Sommerville (2011) demonstra um exemplo básico de avaliar um sistema. Suponha que uma organização possua 10 sistemas legados, estes sistemas podem ser avaliados de acordo com a qualidade e o valor de negócio de cada, assim podendo criar um gráfico que demonstre o valor de negócio relativo e qualidade de sistema, a Figura 1 demonstra esse exemplo.

Figura 1 – Exemplo de avaliação de sistema legado.



Fonte: Sommerville (2011).

Para Sommerville (2011), a visão de valor de negócio e qualidade de sistema para determinar o caminho da evolução segue em duas perspectivas, alta e baixa, ou seja, o valor e a qualidade podem obter uma dessas perspectivas, resultando em uma dessas opções explícitadas a seguir:

- baixa qualidade e baixo valor de mercado: esses sistemas devem ser descartados. Sua manutenção é dispendiosa e a taxa de retorno sobre o negócio é baixa.
- baixa qualidade e alto valor de mercado: podem sofrer alguma forma de reengenharia ou ser substituídos, caso haja algum sistema comercial adequado. Como possuem baixa qualidade, acabam sendo dispendiosos para a empresa, porém, seu alto valor de negócio o impede de serem descartados.
- alta qualidade e baixo valor de mercado: fazer a manutenção sem mudanças dispendiosas. Devido seu baixo valor de mercado, o sistema não é de grande importância para a empresa, mas como sua manutenção é baixa, optar pela substituição ainda não vale a pena.
- alta qualidade e alto valor de mercado: prosseguir com a manutenção normal. Como a qualidade do sistema é alta, não há necessidade de substituição ou reengenharia. Assim a manutenção está correspondendo ao valor de negócio.

Assim sendo, quatro classificações podem ser obtidas como estratégias de evoluções de legados, que são (SOMMERVILLE, 2011):

- descartar o sistema complatamente;
- reengenharia do sistema para aprimorar sua facilidade de manutenção;
- substituir todo ou parte do sistema por um novo sistema;
- deixar o sistema sem alterações e contiuar com a manutenção regular.

Já para Warren (1999), existem três partes constituintes de um sistema legado. Assim, cada uma delas sendo contribuinte para a evolução do sistema. São elas:

- Negócios: O propósito da organização em relação à seus objetivos e metas de negócio são os geradores dos requisitos de seus sistemas, quanto mais radicais forem esses objetivos, mais dificil serão as acomodações de mudanças em seus sistemas. Ao menos que os sistemas sejam projetados para se acomodarem facilmente a mudanças, as mesmas só poderão ser realizadas após um extenso retrabalho.
- Organizacional: Onde um sistema legado é mal documentado e vários indivíduos importantes que mantiveram o sistema se aposentam, a organização normalmente

é menos capaz de evoluir ainda mais o sistema. Um sistema legado fornece serviços para sua organização "operacional". A atitude de uma organização para mudar afeta a capacidade de evolução do sistema. Algumas organizações relutam em aceitar mudanças, por exemplo. Se a mudança for imposta a uma força de trabalho pela alta gerência, a força de trabalho pode rejeitar o sistema de software que implementa essas mudanças.

- Técnico: Do ponto de vista técnico, pode-se decompor uma aplicação legada em aplicação de software, sistema de software e hardware, a condição e a qualidade do aplicação, incluindo sua documentação, são fatores significativos na determinação de quão bem um sistema legado pode evoluir. Uma arquitetura realizada de forma errônea ou uma documentação inconsistente, por exemplo, significa que o sistema não é prontamente evolutivo.

Dado o contexto, geralmente para uma avaliação de legado deve-se determinar seu valor comercial e qualidade técnica. Na visão de Warren (1999), acrescenta-se fatores organizacionais, estes podendo afetar a escolha da estratégia de evolução de um legado.

2.3 Teoria Fundamentada em dados (Grounded Theory - GT)

A GT é uma abordagem de pesquisa qualitativa criada para auxiliar o processo de captura, extração e análise de dados, contando com atividades de sumarização e amostragem dos dados de uma forma visual mais específica e clara para os leitores. Esta metodogolia serve para que o pesquisador consiga gerar explicações (e.g teorias) de um processo, ação ou uma interação moldada pelas visões de um grande número de participantes (CRESWELL, 2014).

De uma perspectiva histórica desta metodologia é possível destacar que seu desenvolvimento ocorreu em 1967 por dois sociólogos chamados Barney Glaser e Anselm Strauss. Ambos possuem origens de centros de pesquisa onde a abordagem qualitativa é fortemente utilizada e difundada no âmbito acadêmico, são elas, respectivamente, a Columbia University e a University of Chicago (GLASER; STRAUSS, 1967). Os autores não possuiam a mesma tradição filosófica e suas pesquisas até então vinham em cursos diferentes, porém, a contribuição de um para o outro foi de suma importância (STRAUSS; CORBIN, 2008).

Seguindo essa ideia, neste trabalho é proposto um estudo que vai além de um trabalho específico para profissionais da área de evolução de software, trabalha-se para que seja didaticamente auto explicativo, para que assim, quaisquer interessado no tema, consiga entender e possa verificar a autenticidade e praticidade dos resultados findados por este estudo. Este trabalho possui um vislumbre de resultar em um mecanismo que possa apontar condições causais que caracterizem um sistema como legado assim como problemas de possuir um legado e possíveis estratégias de evolução para estes sistemas.

Ao longo dos anos, a GT acabou obtendo algumas versões diferentes, sendo adaptada de acordo com os estudos em que eram utilizadas. Porém, é reconhecido que existem pelo menos três fluxos de GT, são eles (STOL; RALPH; FITZGERALD, 2016):

- GT versão Clássica ou Glaseriana (GLASER; STRAUSS, 1967);
- GT versão Straussiana (STRAUSS; CORBIN, 1998);
- GT versão Construtivista (CHARMAZ, 2009).

No trabalho realizado por Stol, Ralph e Fitzgerald (2016), foi possível identificar as variantes existentes da GT, bem como diferenças e semelhanças que os autores adotam em cada variante.

Em relação à Questão de pesquisa Questão de Pesquisa (QP) a GT clássica prioriza que a mesma deve emergir de acordo com o andamento da pesquisa, ou seja, não deve ser criada de antemão, pois Glaser afirma que a criação de uma QP não pode ser forçada. Já a GT Straussiana mostra que a QP pode sim ser criada de antemão, derivada da literatura ou sugerida por terceiro, e que geralmente a QP é ampla e aberta. Diferentemente, a GT construtivista revela que a pesquisa começa com QPs iniciais e as mesmas vão evoluindo ao longo do estudo proposto.

Em relação aos procedimentos de codificação, as três linhas que são mais conhecidas em GT utilizam basicamente 3 fases que contemplam a análise de dados, porém com algumas diferenças notáveis, são elas (STOL; RALPH; FITZGERALD, 2016):

1. Primeira fase, conhecida como **codificação aberta** na GT Clássica e GT Straussiana e **codificação inicial** na GT construtivista;
    - GT Clássica: "fratura" dos dados; recomenda-se a codificação linha a linha para obter uma cobertura teórica completa, mas não rejeita frases ou parágrafos de codificação ou documentos completos.
    - GT Straussiana: Consiste em geração de categorias e como são variadas dimensionalmente e, assim como na GT clássica, pode ser analisada linha por linha, frase, parágrafo ou documentos completos.
    - GT construtivista: Contempla um exame de dados palavra por palavra, linha por linha ou incidente por incidente. É semelhante à codificação aberta de Glaser, pois é proposto a codificação Linha por Linha porém, a autora Charmaz recomenda que essa codificação seja realizada através de "codificação com gerúndio", segundo Tweed e Charmaz (2011) este serve como um dispositivo heurístico para trazer o pesquisador mais ao encontro dos dados, interagindo com os mesmos e estudando cada fragmento dele utilizando os gerúndios contidos nos trechos de texto.

2. Segunda fase, conhecida como **codificação seletiva** na GT Clássica, **codificação axial** na GT Straussiana e codificação focada na Construtivista;

    - GT Clássica: começa a partir da identificação da categoria central por meio da codificação aberta. Nessa etapa, o pesquisador passa a codificar seletivamente em prol da categoria central e categorias relacionadas, de modo que os dados tidos como não relevantes podem ser ignorados.
    - GT Straussiana: tem como objetivo especificar as propriedades e as dimensões de uma categoria e consiste em um processo de reagrupamento dos dados.
    - GT Construtivista: seleciona categorias mais frequentes ou os códigos mais importantes e usa os mesmos para categorizar os dados, não requer uma única categoria principal.

3. Terceira fase, conhecida como **codificação teórica** na GT Clássica e GT Construtiva, e **codificação seletiva na GT Straussiana**;

    - GT Clássica: estabelece relações conceituais entre códigos substantivos, resultando no desenvolvimento de hipóteses. Glaser propõe várias "famílias de codificação".
    - GT Straussiana: tem como objetivo integrar e refinar categorias em um modelo analítico, que consiste na definição da categoria central para em seguida descrever os conceitos em termos de propriedades e dimensões em busca de consistência interna.
    - GT Construtivista: Serve para especificar relação entre as categorias para integrá-las em uma teoria coesa.

Para analisar adequadamente os dados, a GT adota a utilização de questões que devem ser levantadas durante essa análise. Para cada linha da GT, existem questões distintas a serem levadas em consideração, são elas (STOL; RALPH; FITZGERALD, 2016):

- GT Clássica: Estes dados são um estudo de quê? Que categoria ou propriedade este incidente indica? O que está acontecendo atualmente nos dados?
- GT Straussiana: Questões do estilo quem, quando, onde, como, com quais consequências ou sob quais condições algum fenômeno ocorreu, ajudam a descobrir ideias importantes para a teoria.
- GT Construtivista: Estes dados são um estudo de quê? O que este dado sugere? Sob o ponto de vista de quem? A qual categoria teórica esta unidade de dado indica pertencer?

Os autores do método GT (Glaser e Strauss) identificaram os componentes que, aos olhos dos mesmos, são determinantes para a prática da GT. Apoiar-se no uso desses componentes ajuda pesquisadores a obterem mais facilidade de controlar seus processos de pesquisa e também apoia na ampliação do poder analítico de seus estudos, são eles (CHARMAZ, 2009):

- Envolvimento simultâneo na coleta e análise de dados.
- Poder de construir **códigos** e **categorias** analíticas a partir dos dados coletados, e não apenas criar hipóteses preconcebidas e deduzidas através de lógicas.
- Utilização do método comparativo da teoria, que possibilita a elaboração de comparações em quaisquer passo da análise.
- O avanço constante no desenvolvimento da teoria em cada passo realizado da coleta e análise de dados.
- A amostragem final dirigida à construção da teoria sobre o tema, não visando uma mera representividade populacional.

O termo teoria fundamentada foi cunhado para significar que uma teoria é derivada a partir de dados, onde os mesmos são reunidos e analisados através de processos de pesquisa (STRAUSS; CORBIN, 2008). Assim sendo, um pesquisador inicia um estudo, na maioria das vezes, com o objetivo de "criar" uma teoria sobre o assunto proposto, o mesmo não inicia um estudo possuindo uma teoria já criada, salvo o caso onde o pesquisador queira elaborar e estender alguma teoria já existente na literatura. Em outras palavras, uma teoria fundamentada de um tópico estudado começa com dados concretos e termina renderizando-os em uma teoria explicativa.

Conforme o exposto, na maioria das vezes é gerada uma teoria, porém o valor da metodologia GT reside na sua capacidade não apenas de gerar teoria, mas também de basear essa teoria em **dados**. Tanto a teoria quanto a análise de dados envolvem interpretação, mas pelo menos é uma interpretação baseada em investigação sistematicamente realizada (STRAUSS; CORBIN, 1998).

A teoria fundamentada em dados, como seu nome induz, é uma teoria que assemelha-se mais à "realidade" do que a teoria derivada, reunindo uma série de conceitos baseados na experiência ou somente através da especulação (como se pensa que as coisas deveriam funcionar) (STRAUSS; CORBIN, 1998). Teorias fundamentadas, por serem extraídas a partir dos **dados** coletados, tendem a oferecer intuições, aprimorando o entendimento e fornecendo um guia significativo para a ação.

Em relação aos pesquisadores teóricos fundamentados, os mesmos devem possuir ou desenvolver algumas habilidades específicas da área, porém, existem habilidades mais

específicas que podem auxiliar aos pesquisadores iniciantes a entender e desenvolver tais características(STRAUSS; CORBIN, 1998):

- a capacidade de dar um passo atrás e analisar criticamente situações;
- a capacidade de reconhecer a tendência para o viés;
- a capacidade de pensar abstratamente;
- a capacidade de ser flexível e aberto a críticas úteis;
- sensibilidade às palavras e ações dos entrevistados;
- uma sensação de absorção e devoção ao processo de trabalho.

Entretanto, essas características não irão ser desenvolvidas se forem utilizadas como passos de um processo que devem ser seguidos sequencialmente, o intuito é que sejam usadas de forma flexível e criativa, para que sejam aguçadam e trabalhadam tais características (STRAUSS; CORBIN, 1998).

Por ser uma construção de teoria realmente fundamentada em **dados**, é algo que não pode ser refutado completamente por outros dados ou subtituída por alguma outra teoria, o produto construído através dessa metodologia tende a durar mais tempo que demais resultados de estudos realizados por outras metodologias, por estar ligada intimamente nos dados que são coletados (GLASER; STRAUSS, 1967).

Assim explicado, a teoria serve como uma estratégia de derivação de uma idéia geral, ou seja, a partir de um conceito explicitado como fundamental/geral (e.g. Sistemas Legados) usa-se o procedimento desta teoria para, a partir dos **dados coletados**, conseguir melhor interpretá-los e organizá-los, identificando relações entre os conceitos encontrados, categorias, diferenças e causas dos mesmos (CRESWELL, 2014).

> A intenção dessa teoria é descobrir um modelo conceitual que explique o fenômeno a ser investigado e possibilite ao investigador desenvolver e relacionar conceitos, cuja ênfase está na compreensão do fenômeno tal como ele emerge dos dados e não no embasamento em conceitos e teorias do pesquisador(CRESWELL, 2014, 5.3)

.

Dentro de um estudo utilizando a teoria fundamentada, os dados que irão compor o trabalho podem ser capturados através de algumas formas, como em entrevistas (mais comumente), observações, documentos (onde se encaixam dados extraídos da literatura) ou combinações dessas fontes (COYNE; COWLEY, 2006).

Na GT existe um processo onde o texto é analisado criteriosamente, e do mesmo é extraído categorias, para este processo é definido uma nomenclatura denominada **codificação**. Este processo é uma ligação fundamental entre a coleta de dados e o desenvolvimento de uma teoria emergente que funcione como explicação para tais dados coletados (CHARMAZ, 2009).

Esta etapa de codificação contempla alguns procedimentos que servem de auxílio para garantir algum tipo de padronização e rigor para o processo, entretanto, não é necessário seguir à risca, são procedimentos para serem usados de maneira criativa e flexível pelos pesquisadores, são eles (STRAUSS; CORBIN, 2008):

- construir em vez de testar;
- fornecer aos pesquisadores ferramentas analíticas para lidar com as massas de dados brutos;
- ajudar os analistas a considerar significados alternativos para os fenômenos;
- ser sistemático e criativo simultaneamente;
- identificar, desenvolver e relacionar os conceitos que são os blocos de construção da teoria.

Em paralelo, a GT possui um método de comparação dos dados coletados, conhecido como **análise comparativa constante**, onde se constitui um processo de 3 fases distintas (CRESWELL, 2014): codificação aberta, codificação axial e codificação seletiva.

A codificação aberta é a fase de codificação em que o pesquisador necessita analisar o texto capturado a fim de identificar conceitos expostos dentro de cada definição que demonstram ideias gerais conforme o contexto generalizado do estudo. A análise dos textos é realizada através de questionamentos, comparações e reflexões sobre o tema geral. A todo momento essas reflexões são analisadas através de perguntas realizadas constantemente, e com a realização criteriosa dos dados pode-se obter respostas (e.g. quem, quando, onde, como, etc) (GIBBS, 2009), com o intuito de descobrir questões teóricas que estão dentro do texto analisado e mostrando um aprofundamento melhor na percepção dos dados analisados. Conforme os conceitos são capturados, inicia-se uma etapa de sumarização dos dados, conforme suas especialidades ou igualdades semânticas, ou seja, o pesquisador inicia um trabalho de revisão dos conceitos identificados na fase aberta, a fim de criar formas de agrupamentos (categorias) entre os conceitos, a forma que estas categorias vão sendo identificadas, as mesmas são comparadas constantemente entre si, e são aumentadas de acordo com a aparição de novos dados. Este processo reduz o número de categorias e as torna de forma mais hierarquicamente organizadas.

A codificação axial é a fase onde o pesquisador constrói um modelo de referência conceitual utilizando os dados já coletados, para tentar descobrir um problema no tema geral da pesquisa. O passo principal desta etapa é descobrir o principal processo, chamado nesta teoria como categoria central, que explica a ação na cena social, com isso, apoia o pesquisador a construir relações e ligações entre as categorias identificadas, reunindo de forma mais organizada os dados coletados anteriormente, ou seja, é definida uma categoria principal, que é identificada e o restante da teoria é escrita e detalhada “ao redor” desta

categoria, sendo que a mesma pode já ser uma categoria existente na teoria ou criada posteriormente. Após a identificação desta categoria principal, inicia-se um processo de compreensão das categorias, ou seja, é realizada uma minuciosa fase de reorganização dos dados, a fim de identificar relações, ligações entre a categoria central e as demais categorias capturadas.

Por fim, a codificação seletiva é a fase mais abstrata do processo, onde se subsidiará a formação de um esquema organizacional maior, e os resultados compõem a teoria, ainda nesta fase, conceitos e agrupamentos são revistos a fim de alterar algo caso seja necessário, ou seja, é uma fase de refinamento do processo da grounded theory.

De acordo com STRAUSS e CORBIN (2008) este estilo de comparação constante é utilizado para estimular o pensamento analítico sobre propriedades e dimensões e para dirigir a amostragem teórica.

Seguindo o exposto até o momento, a GT foi escolhida como metodologia que será utilizada pois nos fornece uma análise de dados bem definida e criteriosa. Ainda dentro desse conhecimento, teorias fundamentadas tendem a proporcionar maior discernimento e melhor entendimento. Assim, fornecendo um guia importante para ações. Utilizando muito da criatividade, onde se manifesta na capacidade dos pesquisadores de nomear apropriadamente categorias, fazer perguntas estimulantes, fazer comparações e extrair um esquema inovador, integrado e realista das massas de dados brutos desorganizados (STRAUSS; CORBIN, 1998).

2.3.1 Uso da Literatura na GT

Durante a realização do estudo, o pesquisador sempre encontrará artefatos na literatura (e.g biografias, relatórios, estudos) que tendem a possuir relevância para à área que está sendo investigada com a GT, o que deve ser levado em consideração é de que forma esses artefatos irão apoiar e não atrapalhar o desenvolvimento da teoria (STRAUSS; CORBIN, 2008).

Decisões sobre quando utilizar a literatura dependem de dois fatores (CUTCLIFFE, 2000):

- O pesquisador possui um pouco de conhecimento sobre os fenômenos e processos nos quais ele está interessado e não tem certeza sobre a melhor abordagem, ou está ciente de que há falta de conhecimento e decide usar um método de teoria fundamentada.

- É necessário a utilização da literatura se um pesquisador quiser esclarecer conceitos e construir uma nova teoria sobre eles.

Segundo o livro de Corbin e Strauss (2014), pode-se haver dois tipos de literatura relevantes a discussão de GT, são elas:

- Literatura Técnica: Refere-se a relatórios de pesquisa, artigos teóricos ou filosóficos e outras informações características de trabalhos profissionais e disciplinares. A mesma pode ser utilizadas na realização de comparações, na melhoria da sensibilidade analítica do pesquisador, no fornecimento de materiais descritivos, respondendo perguntas para observações iniciais e entrevistas, estimulando questões analíticas e confirmando descobertas.

- Literatura Não Técnica: Refere-se à cartas, biografias, diários, relatórios, fitas de vídeo, memórias, jornais, catálogos, memorandos e uma variedade de materiais. A literatura não técnica pode ser usada para todas as finalidades listadas. Além disso, este tipo de literatura tem a capacidade de ser utilizada como **dados primários** ou **secundários** na pesquisa vigente.

3 ESTADO DA ARTE

Neste Capítulo, é explicitada a condução da busca de trabalhos que realizaram a aplicação do método GT na Engenharia de Software. Foi identificado o trabalho de (BARBOSA, 2017) que realiza uma análise do uso de GT em Engenharia de Software (ES) através de uma Revisão Sistemática na Literatura (RSL) utilizando os procedimentos definidos por (KITCHENHAM, 2004). No entanto, esse trabalho apresenta os resultados obtidos entre os anos de 2005 à 2015. Assim sendo, este trabalho realiza uma extensão da RSL de Barbosa (2017), extraindo resultados de estudos nos anos de 2016, 2017, 2018 e 2019.

3.1 Questão de Pesquisa

As questões de pesquisas importantes para investigar o estado da arte do presente estudo são as mesmas do trabalho de Barbosa (2017):

QP1: **A Grounded Theory tem sido aplicada mais frequentemente em quais disciplinas da ES?**

QP2: **Como tem sido a aplicação das etapas e técnicas de coleta de dados previstas no processo da Grounded Theory nos trabalhos de ES? Quão completa tem sido a utilização de GT? Há o uso de software de apoio?**

QP3: **Tem havido aumento das publicações em ES usando o método Grounded Theory nos últimos anos?**

Com essas questões respondidas é possível demonstrar o estado da arte desta pesquisa, bem como a evolução do uso da GT em ES.

3.2 Base de Dados e String de Busca

No trabalho de Barbosa (2017) foi utilizada apenas uma base de dados, o que não influenciou negativamente o retorno quantitativo de trabalhos desse tipo, contrariando assim pesquisas que afirmam haver poucos trabalhos usando GT na ES. Assim sendo, neste estudo foi utilizada a mesma base, IEEE Xplore

Após a escolha da base de dados, foi aplicada a mesma string de busca de Barbosa (2017), sendo ela:

"grounded theory"AND "software engineering"

- Critérios de Inclusão e Exclusão

Para a triagem dos estudos relevantes à nosso objetivo, foram definidos critérios de inclusão e exclusão de acordo com o estudo Barbosa (2017), com a Tabela 2 é possível visualizar esses critérios.

Tabela 2 – Tabela CI e CE: Estado da Arte.

Tipo de critério	**Critério**
CI	CI1. O estudo é escrito em inglês. CI2. tratar do uso do método Grounded Theory em pesquisas da área de ES especificamente. CI3. O estudo deve ter sido publicado dentre os anos de 2016 e 2019. CI4. O estudo deve conter no mínimo uma resposta há alguma das questões de pesquisa.
CE.	CE1. O estudo não atende algum dos CI.

Fonte: O Autor.

3.3 Resultados Obtidos

A Tabela 3 explicita a quantidade de publicações retornadas no repositório escolhido. Observa-se, portanto, que foram retornadas 46 publicações de 2016 à 2019, e após as aplicações dos CI e CE resultou em 34 artigos relevantes à pesquisa.

Tabela 3 – Número de publicações retornadas.

Repositório	**Quantidade antes CI e CE**	**Quantidade pós CI e CE**
IEEE	46	34

Fonte: O Autor.

As Tabelas 4, 5, 6 e 7 foram criadas para apresentar os resultados da RSL, onde pode-se observar os 34 estudos que atenderam aos CI e CE. Em cada linha da tabela é possível visualizar a referência do artigo, título, a disciplina ou área que a GT é utilizada em ES, às observaçoes sobre o uso da GT tais como técnicas utilizadas, ferramentas de apoio e por fim na última coluna alguns aspectos mais gerais sobre o objetivo do uso da GT.

Tabela 4 – Trabalhos relevantes ao estado da arte. - **(Continua)**

Referência	Título	Disciplina da ES	Observações uso GT	Uso GT
Sousa et al. (2018)	Identifying Design Problems in the Source Code	Design em código Fonte	Método: Gravação de áudio e captura de telas de computador em vídeo. Codificação Aberta, axial e seletiva.	Foi aplicado os princípios de GT para entender e explicar como os desenvolvedores identificam problemas de design no código-fonte.
Song et al. (2017)	How to Support Customisation on SaaS: A Grounded Theory from Customisation Consultants	Customização de software como Serviço	Método: Entrevista. Houve identificação de categoria central. (codificação seletiva)	A Gt serve para investigar a expectativa de consultores especializados em customizar sistemas de software corporativo.
Matsubara e Silva (2017)	Game elements in a software engineering study group: a case study	Games	Método: Entrevistas. Codificação aberta e axial foram conduzidas, e uma história central final foi construída.	O foco em utilizar os preceitos da GT estava em obter insights sobre as percepções dos alunos sobre seu processo de aprendizado com a experiência de aprendizado com gamificação
Giardino et al. (2016)	Software Development in Startup Companies: The Greenfield Startup Model	Desenvolvimento de Software	Método: Entrevistas semiestruturadas, questionários Codificações aberta, axial e seletiva.	Melhorar a compreensão das estratégias de desenvolvimento de software empregadas pelas startups.
Fonseca, Soares e Seaman (2017)	Describing What Experimental Software Engineering Experts Do When They Design Their Experiments - A Qualitative Study	Experimentos Controlados	Método: Entrevistas semiestruturadas. Codificação Aberta e Axial.	Foi realizada uma pesquisa qualitativa GT para entender como experientes pesquisadores de ES planejam seus experimentos controlados.
Zakaria, Ibrahim e Mahrin (2016)	Using Grounded Theory Approach to Identify Value-Based Factors in Software Development	Desenvolvimento de software	Método: RSL codificação inicial e intermediária.	Identificar fatores baseados em valor no desenvolvimento de software utilizando a teoria fundamentada como estratégia de análise de dados.
Hoda et al. (2017)	Socio-Cultural Challenges in Global Software Engineering Education	Engenharia de software global	Método: Entrevistas. Codificação Aberta	Foi utilizado o procedimento de codificação aberta da GT para descobrir aspectos socioculturais do GSEE (Engenharia de software global).
Sedano, Paul e Péraire (2017)	Lessons Learned from an Extended Participant Observation Grounded Theory Study	Desenvolvimento de software	Método: Entrevistas Não especifica técnicas.	O artigo tem como objetivo estudar sobre a GT construtivista, faz-se uso da GT em um grupo de desenvolvimento de software.
Hoda e Noble (2017)	Becoming Agile: A Grounded Theory of Agile Transitions in Practice	Desenvolvimento ágil	Método: Entrevistas. Uso de codificação aberta e comparação constante.	A gt é utilizada para criar uma teoria de como é realizada a transição de desenvolvimento de software para desenvolvimento ágil.
(HOLMES; ALLEN; CRAIG, 2018)	Dimensions of Experientialism for Software Engineering Education	Ensino na ES	Método: Entrevista. Codificação Aberta.	A codificação aberta da GT foi realizada para entender os benefícios que os alunos sentiam que recebiam da UCOSP (Programa de Projetos de Código Aberto de Iniciação Científica).

Fonte: O Autor.

Tabela 5 – Trabalhos relevantes ao estado da arte. - **(Continua)**

Referência	Título	Disciplina da ES	Observações uso GT	Uso GT
Garousi e Herkiloglu (2016)	Selecting the right topics for industry-academia collaborations in software testing: an experience report	Teste de Software	Método: Reuniões Codificação aberta, axial e seletiva.	O artigo tem como objetivo propor diretrizes baseadas em experiência em teste de software, a GT foi utilizada para encontrar (convergir para) tópicos que seriam "interessantes"e úteis tanto do ponto de vista industrial quanto acadêmico.
Valentim, Silva e Conte (2017)	The Students' Perspectives on Applying Design Thinking for the Design of Mobile Applications	Design de software	Método: Questionários. Codifcação aberta e axial.	Procedimentos da GT foram utilizados para analisar dados sobre experiência de alunos de pós graduação em utilizar Design Thinking (DT).
Prechelt, Schmeisky e Zieris (2016)	Quality Experience: A Grounded Theory of Successful Agile Projects Without Dedicated Testers	Desenvolvimento ágil	Método: Entrevistas. Codificação aberta, axial e seletiva.	Procedimentos da GT são utilizados para analisar resultados de equipes que trabalham com desenvolvimento convencional de software e equipes ágeis, para demonstrar como as mesmas trabalham com equipe de testadores.
Gralha, Damian e Wasserman (2018)	The Evolution of Requirements Practices in Software Startups	Engenharia de requisitos	Método: Observação e entrevistas Codificação aberta e axial.	GT utilizada para verificar a evolução das práticas da engenharia de requisitos dentro de Startups.
Cereci e Karakaya (2018)	Need for a Software Development Methodology for Research-Based Software Projects	Desenvolvimento de software	Método: Entrevistas semi estruturadas. Não especifica técnicas.	A GT foi utilizada como objetivo de compreender e categorizar os problemas enfrentados pelos acadêmicos em projetos de pesquisa.
Alatawi, Mendoza e Miller (2018)	Psychologically-Driven Requirements Engineering: A Case Study in Depression Care	Engenharia de requisitos	Não possui identificação	O artigo propõe uma nova abordagem de engenharia de requisitos, orientada psicologicamente, que captura e modela os valores, motivações e emoções pessoais dos envolvidos. Usando uma abordagem de GT, foi analisada as informações sobre as partes interessadas e o domínio.
Cunha et al. (2016)	Decision-Making in Software Project Management: A Qualitative Case Study of a Private Organization	Gerenciamento de projeto	Método: Entrevista semi estruturada. Não especifica as atividades.	GT analisa os dados a fim de identificar os fatores individuais e externos que levam a decisões de projeto mais ou menos bemsucedidas.
Rodriguez, Emilia e Turhan (2018)	Key Stakeholders' Value Propositions for Feature Selection in Software-intensive Products: An Industrial Case Study	Gerenciamento de software	Método: Entrevista. Codificação aberta e seletiva.	O artigo fornece uma análise aprofundada das proposições de valor dos principais interessados ao selecionar recursos para o produto de software intensivo de uma grande empresa de telecomunicações.
Zieris e Prechelt (2016)	Observations on Knowledge Transfer of Professional Software Developers during Pair Programming	Desenvolvimento de software	Método: Gravações. Codificação aberta, axial e seletiva.	GT utilizada para entender como a transferência de conhecimento durante a programação em pares funciona e, eventualmente, fornecer orientação para os profissionais.

Fonte: O Autor.

Tabela 6 – Trabalhos relevantes ao estado da arte. - **(Continua)**

Referência	Título	Disciplina da ES	Observações uso GT	Uso GT
Santos et al. (2017)	Towards a Theory of Simplicity in Agile Software Development: A Qualitative Study	Desenvolvimento de software	Método: entrevista semi estruturada. Comparações constantes, relações entre categorias e memorandos foram escritos.	GT utilizada para entender como gerentes de projetos e engenheiros de software interpretam suas experiências em projetos ágeis de software, considerando os problemas de simplicidade no desenvolvimento de software ágil.
Rahman, Stallings e Williams (2018)	Poster: Defect Prediction Metrics for Infrastructure as Code Scripts in DevOps	Desenvolvimento de software	Método: uso de commits relacionados a defeitos extraídos de sistemas de controle de versão. GT Construtivista.	GT serve para apoiar e definir métricas de apoio ao desenvolvimento de scripts de infraestrutura como código (IaC) propondo métricas relacionadas ao modelo de previsão de defeitos.
Abad, Ruhe e Noaeen (2016)	Requirements Engineering Visualization: A Systematic Literature Review	Engenharia de requisitos	Método:RSL. Não especifica técnicas utilizadas da GT.	A GT auxilia na análise de resultados obtidos através de uma RSL relacionada a visualização de engenharia de requisitos.
Beller et al. (2016)	Analyzing the State of Static Analysis: A Large-Scale Evaluation in Open Source Software	Projetos de software	Método: Documentos online. Não especifica técnicas utilizadas da GT.	Utiliza a GT para analisar dados sobre o uso de ferramentas automatizadas de análise estática em projetos de software.
Kurtanovic e Maalej (2017)	Mining User Rationale from Software Reviews	Gerenciamento de software	Método: leitura de avaliações on-line de App's. Codificação aberta, axial e seletiva.	Por meio de uma abordagem de teoria fundamentada e análise de conteúdo por pares, investigamos como os usuários discutem e justificam suas decisões, por exemplo, sobre a atualização, instalação ou alternância de aplicativos de software.
Bick et al. (2018)	Coordination Challenges in Large-Scale Software Development: A Case Study of Planning Misalignment in Hybrid Settings	Desenvolvimento de software	Métodos: Entrevistas. Codificação aberta, seletiva e teórica.	A GT serve para analisar dados que exploram como e por que uma combinação de planejamento tradicional em nível interequipe e desenvolvimento ágil em nível de equipe pode resultar em coordenação ineficaz.
Júnior (2018)	Toward a Theory of Communication in Distributed Software Development Teams: A Research Proposal	Desenvolvimento de Software Distribuído	Método: estudos online Codificação inicial e focada e a proposta de categorias, práticas de memorandos escritos e codificação axial.	A GT apoia para apresentar uma proposta de pesquisa de doutorado para o estudo da comunicação em equipes de DDS (Desenvolvimento de Software Distribuído), para propor uma nova teoria e, assim, estabelecer uma base teórica para estudos futuros.
Rodeghero (2017)	Behavior-Informed Algorithms for Automatic Documentation Generation	Design de código fonte	Método: Observação em campo. Codificação aberta.	A GT serve para apoiar e determinar se fatores externos ao código influenciam o resumo do código-fonte, levando em consideração que códigos fontes devem ser bem documentados para ajudar programadores a entender os mesmos.

Fonte: O Autor.

Tabela 7 – Trabalhos relevantes ao estado da arte.

Referência	Título	Disciplina da ES	Observações uso GT	Uso GT
Rahman e Williams (2018)	Characterizing Defective Configuration Scripts Used for Continuous Deployment	Verificação e validação	Método: Recursos textuais. Não especifica as atividades da GT.	O estudo tem objetivo de ajudar os profissionais de software a priorizar os esforços de validação e verificação de scripts de infraestrutura como código (IaC), identificando as características dos scripts IaC com defeito. Usando os recursos extraídos e aplicando GT, caracterizamos os scripts IaC com defeito.
Zanatta, Machado e Steinmacher (2018)	Competence, Collaboration, and Time Management: Barriers and Recommendations for Crowdworkers	Desenvolvimento de software	Método: RSL. Não especifica as técnicas. Ferramenta de apoio: Nvivo®.	O estudo visa investigar as dificuldades que os trabalhadores da multidão enfrentam em plataformas de desenvolvimento de software de crowdsourcing, e a GT serve para analisar dados coletados sobre este tema.
Barke e Prechelt (2018)	Some reasons why actual cross-fertilization in cross-functional agile teams is difficult	Desenvolvimento ágil	Método: Entrevistas e Observações. Codificação focada e codificação teórica. Ferramenta de apoio: MAXQDA.	A contribuição é explicar como e por que a fertilização cruzada é difícil aplicando a Metodologia da Teoria Fundamentada nos Dados observacionais e de entrevistas a partir de diversos contextos de equipes ágeis.
Mertz e Nunes (2017)	A Qualitative Study of Application-level Caching	Desenvolvimento de software	Método: Investigação de aspectos em aplicações *Web*. Codificação aberta, axial e seletiva.	O estudo apresenta os resultados de um estudo qualitativo de como os desenvolvedores lidam com a lógica de armazenamento em cache em seus aplicativos da Web.
Nicolás et al. (2018)	On the Risks and Safeguards for Requirements Engineering in Global Software Development: Systematic Literature Review and Quantitative Assessment	Engenharia de requisitos	Método: RSL. Codificação aberta, axial e seletiva.	Identificar os riscos para RE (engenharia de requisitos) que ocorrem em GSD (desenvolvimento de software global), juntamente com as salvaguardas com as quais gerenciar esses riscos. Inspirados por uma abordagem de teoria fundamentada.
Schramm e Daneva (2016)	Implementations of Service Oriented Architecture and Agile Software Development: What Works and What Are the Challenges?	Desenvolvimento de software	Método: análise de posts de blogs profissionais publicados na plataforma Developerworks da IBM. Codificação aberta, axial e seletiva, usada as práticas de comparação constante. Ferramenta de apoio: Atlas.ti 7.	Este estudo da teoria fundamentada encontrou armadilhas ao usar SOA e desenvolvimento ágil de software conjuntamente.
Kugele et al. (2017)	On Service-Orientation for Automotive Software	Desenvolvimento de software	Método: Entrevista. Codificação aberta, axial e seletiva.	Foi aplicado o método da teoria fundamentada para obter uma teoria e um conjunto de requisitos para uma abordagem SOA automotiva.

Fonte: O Autor.

3.4 Discussão dos Resultados

Em relação à QP1 o estudo de Barbosa (2017) obteve como resultado a observação que dentro da ES os temas que mais utilizam a GT como metodologia de pesquisa são as áreas, por exemplo, de metodologias ágeis. Contudo, foi possível observar, segundo os resultados de Barbosa, uma carência de utilização da GT em **Evolução de software**.

A Tabela 8 foi elaborada para responder a QP1, de acordo com os resultados obtidos nesta extensão da RSL de Barbosa (2017) demonstrando as áreas e disciplinas que foram identificadas dentre os 34 estudos resultantes da RSL.

Tabela 8 – Disciplinas/Áreas da ES mais Frequentes utilizando GT.

Disciplinas/Área de ES	**Quantidade de pesquisas relacionadas**
Desenvolvimento de software	12
Engenharia de Requisitos	04
Desenvolvimento ágil	03
Gerenciamento de Software	02
Gerenciamento de Projetos	02
Design em Código Fonte	02
Customização de Software como Serviço	01
Experiência em Gamificação	01
Experimentos Controlados	01
Engenharia de Software Global	01
Ensino na ES	01
Teste de Software	01
Design de Software	01
Desenvolvimento de Software Distribuído	01
Verificação e Validação	01

Fonte: O Autor.

Diante o exposto na Tabela 8, é possível visualizar que o campo de Desenvolvimento de Software é onde a GT vêm sendo mais utilizada em pesquisas, sendo seguida pelas disciplinas de Engenharia de Requisitos e Desenvolvimento Ágil. Este estudo sobre sistemas legados se encaixa na área de Evolução de Software. Assim, foi possível visualizar que, ao menos na base IEEE Xplore, não foi possível encontrar estudos que utilizem GT nessa disciplina, assim corroborando com os resultados obtidos por Barbosa (2017). Com essa observação resultante, é possível verificar que não foi possível analisar trabalhos relacionados.

Os resultados que respondem à QP2, levam em consideração a aplicação das etapas e ténicas de coleta da GT em cada estudo, se o processo está sendo usado completamente ou apenas em partes e se há o uso de ferramentas de apoio.

Em relação ao uso parcial ou total das etapas ou procedimentos da GT, a Tabela 9 mostra até onde os estudos aplicam as atividades previstas em cada versão da GT.

Tabela 9 – Etapas da GT alcançadas nos estudos.

Etapa GT reportada	**Número de artigos**
Codificação aberta	05
Codificação aberta e axial	05
Codificação aberta, axial e seletiva	09
Codificação aberta e seletiva	01
Codificação seletiva	01
Codificação inicial e focada	01
Codificação teórica	01
Codificação focada e teórica	01
codificação inicial e intermediária	01
Codificação aberta, seletiva e teórica.	01
Não especifica as atividades	08

Fonte: O Autor.

Seguindo o exposto na Tabela 9, é possível verificar que a maioria dos estudos, aqueles que relatam a utilização das atividades da GT, reportam o uso de todas as etapas da versão Straussiana da GT, seguidos por uma frequência alta de uso ao menos de 2 etapas principais desta versão de GT (Codificação aberta e axial), etapas que este estudo também seguirá para concluir seus objetivos de análise.

Em relação às técnicas de coleta, a Tabela 10 demonstra que o uso de entrevista como técnica de coleta de dados é a mais frequente dentre as demais técnicas, este resultado reforça os obtidos por Barbosa (2017), onde também é possível visualizar que as entrevistas são as técnicas mais utilizadas.

Tabela 10 – Técnicas utilizadas nos estudos que utilizam GT.

Técnica Utilizada	**Número de artigos**
Entrevistas	17
Questionário	2
RSL	4
Observações	3
Documentos online	3
Reuniões	1
Gravações	2
Uso de commits	1
Leitura de avaliações on-line de App's	1
Investigação de aspectos em aplicações Web	1
Recursos textuais	1
Não identificada	1

Fonte: O Autor.

Conforme o exposto, apesar da diferença de frequência no uso de entrevistas em relação com as demais técnicas, é possível verificar que o uso de **documentos online** somam-se aos estilos de técnicas utilizadas. Vale destarcar que este estudo irá utilizar deste caminho para coletar os dados que farão parte da análise posteriomente.

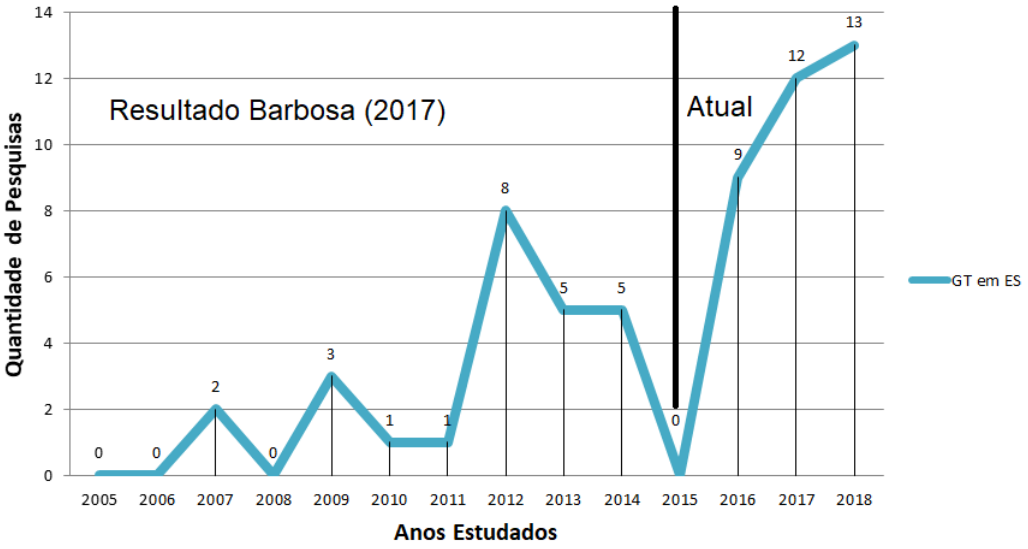
Em relação à utilização de ferramentas de apoio no processo da GT, as seguintes foram encontradas:

- Atlas.ti 7.
- Nvivo®.

- MAXQDA.

A figura 2 responde a QP3, pois demonstra a evolução de pesquisas dentro da ES que abordam o uso de GT. é mostrado os resultados obtidos no estudo de Barbosa (2017) (entre 2005 à 2015) e os resultados obtidos nessa replicação nos anos de 2016, 2017 e 2018. O ano de 2019 não é mostrado pois nenhum estudo deste ano atendeu aos CI.

Figura 2 – Evolução de Pesquisas relacionadas sobre GT em ES.



Fonte: O Autor.

Conforme o exposto, o trabalho de Barbosa (2017) obteve resultados que coincidem com os resultados que esta replicação de sua RSL obteve, são eles:

- as disciplinas que contêm mais pesquisas de uso da GT em ES, mostrando a carência de pesquisas que utilizem esta metodologia na área de Evolução de Software (onde este estudo se encaixa);
- o uso em parte dos procedimentos da GT são mais frequentes, invés de utilizar a GT de ínicio ao fim, coincidindo com o que é proposto neste estudo;
- evolução ao longo dos anos na frequência de pesquisas que utilizem a metodologia GT dentro de áreas e disciplinas da ES.

Dessa forma, já que o trabalho de Barbosa (2017) não obteve como resultado nenhuma pesquisa que tenha sido na área de Evolução de Software, assim como esta replicação, este estudo não possui trabalhos relacionados.

4 METODOLOGIA

Na metodologia de realização deste estudo, foi adotada uma abordagem de pesquisa com ênfase em dados qualitativos que utilizou elementos da Teoria Fundamentada em Dados, conhecida como *Grounded Theory (GT)*. A GT foi utilizada para apoiar a análise dos dados com o objetivo de identificar características, problemas e estratégias de evolução de sistemas legados.

Essa metodologia foi aplicada nesse estudo, por fornecer uma forma de análise de dados bem definida e criteriosa. Assim, proporcionou um maior discernimento e melhor entendimento dos elementos que envolvem um sistema legado.

Seguindo as noções apresentadas na seção 2.3, de acordo com as principais diferenças entre as vertentes da GT, nesse trabalho foi utilizada a versão da GT conhecida como Straussiana desenvolvida por STRAUSS e CORBIN (2008). O motivo da escolha da linha Straussiana se justifica porque a pesquisa, desde o ínicio, partiu de uma questão que identificou claramente o problema estudado.

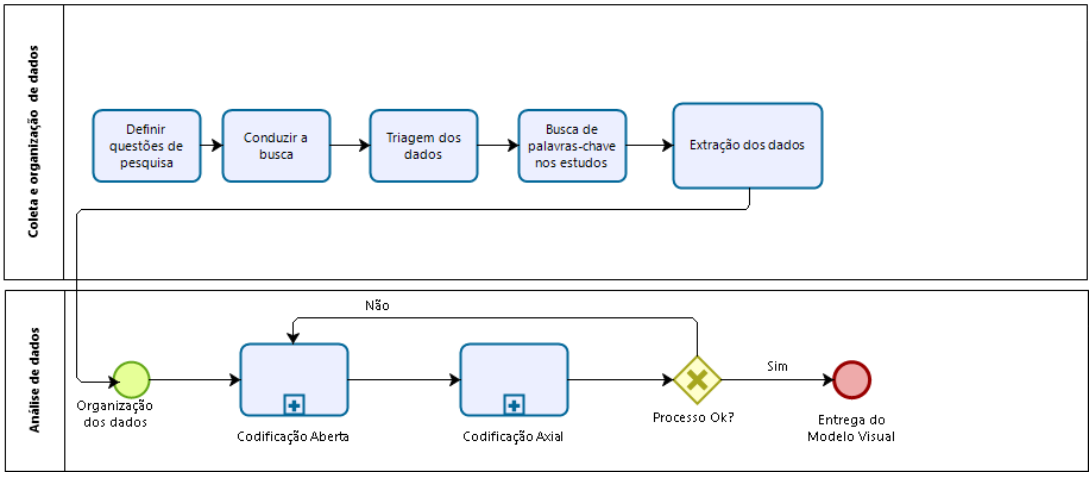
Complementarmente, Santos et al. (2018) contribuiram com a escolha da linha Straussiana, justificando a escolha da seguinte forma:

> A perspectiva Straussiana pode ser considerada uma opção mais indicada para pesquisadores iniciantes no método, pois apresenta um sistema de análise de dados mais sistemático em relação às outras vertentes da GT. Para a adoção das perspectivas clássica e construtivista, pode ser necessário um tempo maior para o desenvolvimento da pesquisa, tendo em vista a abstração teórica necessária para interpretação dos dados e elaboração da teoria sem a adoção de um modelo paradigmático norteador. Nesse sentido, o tempo destinado para a coleta e análise é um fator a ser considerado, principalmente, por estudantes e orientadores de graduação e pós-graduação (SANTOS et al., 2018).

A Figura 3 demonstra o processo geral dos procedimentos metodológicos necessários para cumprir os objetivos deste trabalho, contendo a fase de coleta de dados e a fase de análise de dados de acordo com alguns procedimentos definido pela Grounded Theory versão Straussiana STRAUSS e CORBIN (1998).

Ao final deste Capítulo, é apresentado uma seção que explana sobre o uso de uma ferramenta metodológia desenvolvida especialmente para a codificação aberta.

Figura 3 – Processo dos procedimentos metodológicos



Fonte: Adaptado de Kitchenham (2004) e (STRAUSS; CORBIN, 1998).

4.1 Processo Metodológico

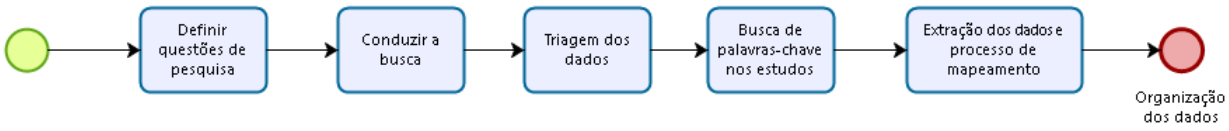
Nessa Seção, são expostos os processos realizados no desenvolvimento da metodologia desse trabalho de conclusão de curso.

4.1.1 Coleta e organização de dados

A fase de coleta e organização dos dados, relacionados à literatura científica da área de sistemas legados, deu início ao processo da metodologia. Essa fase foi realizada conforme as atividades propostas para um protocolo de Mapeamento Sistemático na literatura Mapeamento Sistemático na Literatura (MSL) por (PETERSEN et al., 2008).

A Figura 4 demonstra como foi realizado o protocolo do mapeamento sistemático.

Figura 4 – Processo Coleta e organização de dados



Fonte: Adaptado de Kitchenham (2004).

4.1.1.1 Definir questões de pesquisa

Como o objetivo era encontrar trechos textuais que contivessem definições, problemas ou estratégias de evolução de sistemas legados as seguintes questões de pesquisa

foram elaboradas:

- **QP1**: O que é um sistema legado?
- **QP2**: Quais os problemas de um sistema legado?
- **QP3**: Quais estratégias existem para a evolução de um sistema legado?

4.1.1.2 Conduzir a busca

A condução da busca foi realizada em 2 passos distintos, sendo eles: definir as bases de dados para o estudo e construir a string de busca utilizada para encontrar estudos relevantes à pesquisa.

4.1.1.3 Bases de dados

Foram escolhidas bases de dados que pudessem retornar os resultados de textos completos, ou seja, não apenas título, resumo e keywords, pois o objetivo era capturar o maior número possível de definições, problemas e possíveis estratégias de evolução de sistemas legados.

Assim, as bases escolhidas para este trabalho foram a ACM Digital library, IEEE Xplore e Science Direct, pois são as que permitem busca em texto completo, além de serem bases conhecidas e utilizadas no âmbito acadêmico.

4.1.1.4 Strings de busca

Foram definidos 3 conjuntos de strings de busca, um para cada base de dados, sendo que cada string foi alterada de acordo com os parâmetros exigidos pelos mecanismos de busca de cada base, contendo as palavras-chave consideradas imprescindíveis para os resultados desejados.

Como um dos objetivos era capturar definições não bastaria apenas buscar com palavras gerais (e.g., legacy system, legacy software), pois as mesmas acabariam retornando um número de trabalhos expressivo, o que tornaria o processo não factível no tempo disponível para a realização do trabalho. Para mitigar isso, adaptou-se a string para que retornasse estudos que, em seu texto completo, possuíssem sentenças como *legacy system is, legacy software is, legacy systems are*.

Para a captura de estudos que abordassem também problemas e estratégias de evolução de sistemas legados, foram escolhidos termos mais conhecidos como por exemplo, *trouble*, *problem* e *strategy*, *solution*, *evaluation*.

Conforme o exposto, as strings de busca foram concebidas utilizando-se as frases escolhidas, bem como seus plurais e sinônimos, a fim de retornar trabalhos com a qualidade necessária. As strings podem ser visualizadas na Tabela 11.

Tabela 11 – String de Busca Genérica.

String Genérica
("legacy system is"OR "legacy systems are"OR "legacy software is"OR "legacy code is"OR "legacy codes are"OR "legacy application is"OR "legacy applications are"OR "legacy information system is"OR "legacy information systems are"OR "legacy software system is"OR "legacy software systems are") AND (("difficulty"OR "trouble"OR "problem"OR "complication") OR ("framework"OR "decision making"OR "decision-making"OR "assessment"OR "evaluation"OR "management"OR "model"OR "strategies"OR "strategy"OR "solution")
String IEEE Xplore
(("Full Text & Metadata":"legacy system is"OR "legacy systems are"OR "legacy software is"OR "legacy code is"OR "legacy codes are"OR "legacy application is"OR "legacy applications are"OR "legacy information system is"OR "legacy information systems are"OR "legacy software system is"OR "legacy software systems are") AND (("Full Text & Metadata":"difficulty"OR "trouble"OR "problem"OR "complication") OR ("Full Text & Metadata":"framework"OR "decision making"OR "decision-making"OR "assessment"OR "evaluation"OR "management"OR "model"OR "strategies"OR "strategy"OR "solution")))
String ACM
content.ftsec: (("legacy system is"OR "legacy systems are"OR "legacy software is"OR "legacy code is"OR "legacy codes are"OR "legacy application is"OR "legacy applications are"OR "legacy information system is"OR "legacy information systems are"OR "legacy software system is"OR "legacy software systems are") AND (("difficulty"OR "trouble"OR "problem"OR "complication") OR ("framework"OR "decision making"OR "decision-making"OR "assessment"OR "evaluation"OR "management"OR "model"OR "strategies"OR "strategy"OR "solution")))
String Science Direct
(("legacy system is"OR "legacy software is"OR "legacy code is"OR "legacy application is"OR "legacy software system is") AND (("difficulty"OR "trouble"OR "problem"OR "complication") OR ("framework"OR "evaluation"OR "strategy"OR "solution")))

Fonte: O Autor.

4.1.1.5 Triagem dos dados

De acordo com as atividades do processo de mapeamento, os CI e CE servem para realizar uma triagem dos estudos que foram retornados pelas bases de dados escolhidas, removendo estudos que não são relevantes para a pesquisa. Os critérios adotados por este trabalhos são os listados na Tabela 12.

Tabela 12 – Tabela com CI e CE

Tipo de critério	Critério
Critério de inclusão	CI1. O estudo deve ter sido publicado no ano de 2018. CI2. O estudo deve conter em seu corpo no mínimo uma definição sobre sistema legado ou um problema ou uma estratégia de evolução. CI3. O estudo deve conter no mínimo uma resposta a alguma das questões de pesquisa.
Critério de exclusão	CE1. O estudo não é fornecido por completo pela base da dados. CE2. O estudo é duplicado. CE3. O estudo não atende algum dos CI.

Fonte: O Autor.

4.1.1.6 Extração dos dados

O MSL serviu como base para encontrar estudos cujos trechos de dados relacionados à definições, problemas e estratégias de evolução de sistemas legados foram extraídos.

A Tabela 13 e a Figura 9, na seção 4.2.1, mostram os resultados obtidos em cada base de dados, utilizando cada *string* de busca e, os resultados finais de cada ciclo de aplicação dos critérios de inclusão e exclusão. As seções a seguir demonstram mais detalhadamente como é realizado o processo da análise de dados.

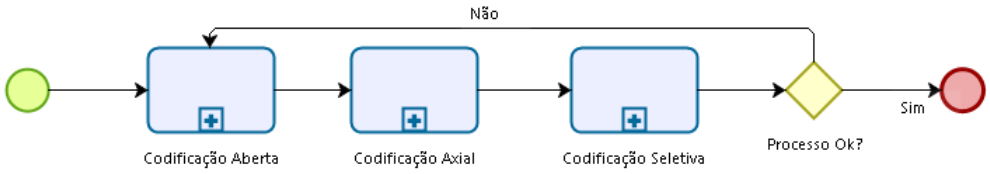
4.1.2 Análise de dados

Após a extração dos trechos os mesmos foram analisados de acordo com os preceitos de codificação. A codificação dentro de uma pesquisa qualitativa consiste na capacidade de extrair, de dentro de um texto, dados que clarifiquem o tema que está sendo estudado. Codificar tem um propósito de reconhecer que não há apenas exemplos diferentes de coisas em textos, mas que existem diferentes tipos de coisas às quais pode-se fazer referência (GIBBS, 2009), ou seja, é uma habilidade de identificar categorias ou agrupamentos contendo códigos semelhantes que exemplificam características sobre o tema estudado.

Seguindo estas proposições, codificar trechos de textos sobre sistemas legados, dá uma oportunidade de categorizar de forma mais processual um tema que ainda não é consenso na literatura e, consequentemente, na comunidade.

Este estudo terá como norte o uso do processo de codificação desenvolvido e utilizado por STRAUSS e CORBIN (1998), que propõem a utilização de 3 fases, descritas de forma sucinta na seção 2.3 e melhor explicada nesta seção. A Figura 5 explicita processo utilizado.

Figura 5 – Processo de Análise de Dados



Fonte: Adaptado de STRAUSS e CORBIN (1998).

Observa-se que nesta pesquisa foram utilizadas 2 fases, sendo elas: **Codificação Aberta** e **Codificação Axial**. A fase de Codificação Seletiva não foi considerada relevante para os objetivos desta pesquisa. Essa decisão foi tomada, pois essa fase contempla atividades como, identificação de categoria central e revisão do modelo visual. Ambas atividades foram desenvolvidas ao longo do processo e não necessitaram de uma fase no final da aplicação da GT.

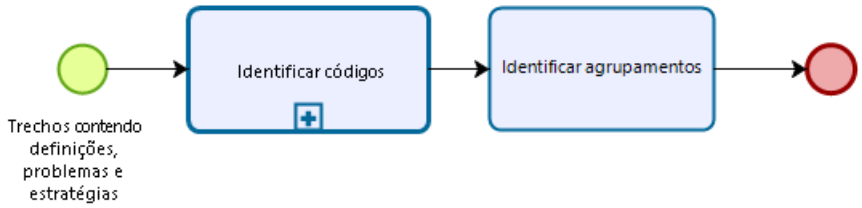
4.1.2.1 Codificação aberta

Nesta etapa buscou-se analisar cada trecho coletado a fim de identificar conceitos (códigos), ou seja, buscou-se através da leitura do trecho separar palavras-chave que refletissem a ideia principal do trecho.

Também nesta etapa, foi realizado o agrupamento dos códigos, atribuindo um nome conceitual ou abstrato para cada agrupamento realizado. Grupos que possuiam alguma semelhança semântica foram agrupados em um grupo de maior nível.

Dessa forma, esta etapa é composta por 1 subprocesso e 1 atividade, os quais são visualizados na Figura 6 e explicados nas subseções seguintes.

Figura 6 – Processo de Codificação Aberta

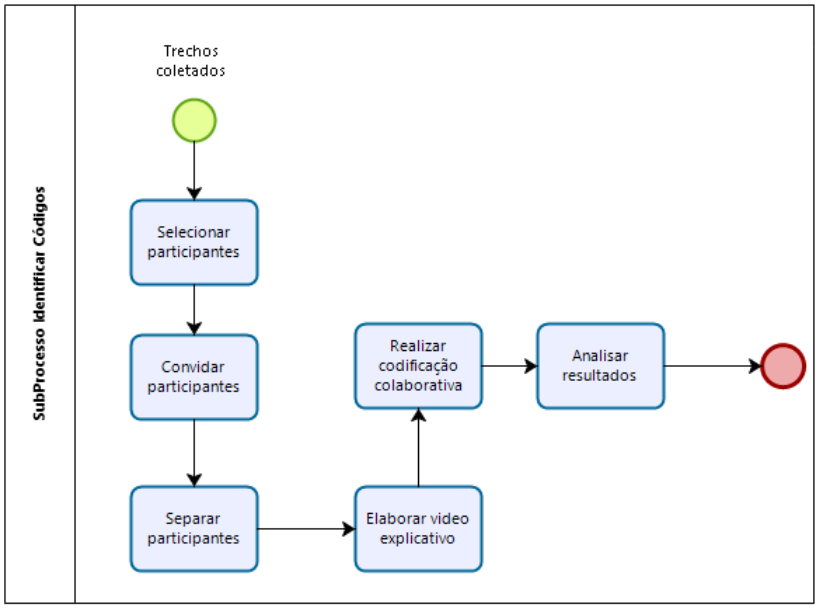


Fonte: O Autor.

4.1.2.2 Identificar Códigos

A identificação dos códigos foi realizada através de um **processo colaborativo**, com o objetivo de tornar o processo mais confiável e factível em um período de tempo. O processo consistiu na escolha de uma quantidade "x" de pessoas (codificadores) para analisarem os trechos e capturarem os códigos dos mesmos. O processo utilizado nessa codificação é demonstrado na Figura 7.

Figura 7 – SubProcesso Identificar Códigos



Fonte: O Autor.

Esse método de codificação colaborativa foi desenvolvido seguindo alguns preceitos do método *Delphi*. Segundo os seus criadores Dalkey e Helmer (1963), esse método vislumbra a descoberta de opiniões através de um consenso mais confiável fornecido por um grupo de especialistas.

De acordo com Munaretto, Corrêa e Cunha (2013) o método Delphi possui algumas características distintas, as quais algumas serviram de inspiração para a elaboração do processo de codificação colaborativa deste trabalho, são elas:

- **Consenso**: Utilizar da melhor forma o grupo de pessoas para obter um consenso mais consiso sobre o tema exposto, assim criando sinergia de opinião;
- **Uso de especialistas**: Assim, é formado conceitos, resultados mais confiáveis a respeito do tema porposto;
- **Anonimato**: Com este conceito é possível analisar o entendimento e as respostas oriundas de cada participante.

4.1.2.3 Realizar codificação colaborativa

A codificação colaborativa foi realizada utilizando a aplicação web ***Open Code Tool*** como ferramenta de apoio. A ferramenta foi planejada e desenvolvida para atender as necessidades deste trabalho. A execução desse processo, através da *Open Code Tool*, está detalhada e mostrada na seção 4.2.3.5.

4.1.2.4 Identificar agrupamentos

Após as identificações de conceitos (códigos), inicia-se uma fase para "agrupar" os mesmos em grupos considerados mais amplos, identificando características comuns entre os conceitos capturados. No exemplo da Figura 18 pode-se visualizar os códigos iniciais. Na Figura 20 pode-se visualizar os agrupamentos que foram definidos para esse exemplo.

O objetivo principal desta fase é diminuir numericamente o número de conceitos obtidos. Os autores STRAUSS e CORBIN (1998) afirmam que a categorização permite reduzir o número de unidades de análise trabalhadas, objetivando a clareza. A execução desse processo é mostrada na seção 4.2.4.

4.1.2.5 Codificação axial

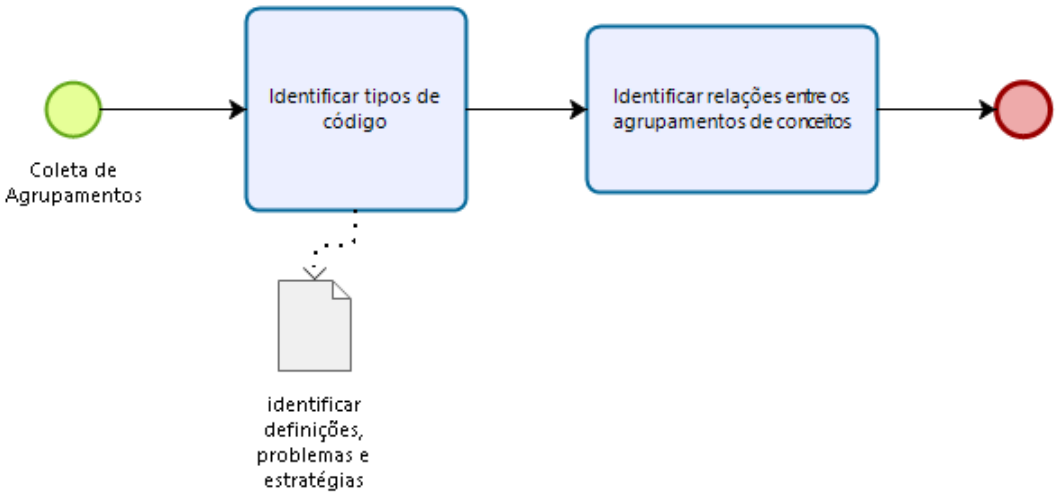
Essa é fase da codificação em que os grupos, identificados na fase anterior, foram novamente analisados a fim de identificar relações entre eles. Objetivou-se associar os grupos aos subgrupos e verificar o modo como as mesmas estão relacionadas (CHARMAZ, 2009). Assim, esta fase é o ínicio do reagrupamento dos dados que foram identificados e separados em categorias na fase da codificação aberta, priorizando o estabelecimento de conexões entre os agrupamentos.

Nesta fase de codificação axial, STRAUSS e CORBIN (1998) sugerem que os códigos sejam estruturados através da criação de **modelos**. Gibbs (2009) relata que "Modelos são estruturas que tentam explicar o que foi identificado como aspectos fundamentais de um fênomeno em estudo".

Assim, **Mapas Conceituais** foram o estilo de modelo escolhido para apoiar o entendimento sobre os aspectos de cada grupo em relação à sua categoria central (Sistemas Legados).

Na Figura 8, pode-se visualizar como é realizado o processo de análise dos dados dentro da codificação axial.

Figura 8 – Processo Codificação Axial



Fonte: Adaptado de STRAUSS e CORBIN (1998).

Nas subseções posteriores, será melhor explicado sobre cada uma das atividades que compõem este processo de codificação.

4.1.2.6 Identificar tipos de código

Nesta fase ocorrerá a identificação dos tipos de códigos. STRAUSS e CORBIN (1998) sugerem a criação de 6 tipos de códigos, onde cada elemento tenha influência causal sobre o texto, são eles: Condições causais, Fenômeno, Estratégias, Contexto, Condições que influenciam, Ação/interação e Consequências.

Para esclarecimento, nesta pesquisa é identificado apenas esses elementos pois estão de acordo com as string's de busca.

4.1.2.7 Identificação de relações entre os agrupamentos de conceitos

Nesta fase, cria-se relações entre os grupos de conceitos que foram criados nas fases anteriores e à categoria central. A execução dessa atividade está exposta na seção 4.2.4.5.

4.1.3 Ferramentas metodológicas

Esta seção é destinada para a ferramenta utilizada neste trabalho.

A *Open Code Tool* é uma ferramenta desenvolvida para a etapa de codificação aberta. A mesma foi explicitada na seção 4.2.3.5.

A *Open Code Tool* foi elaborada visando que a mesma possua algumas funcionalidades específicas para este estudo, são elas:

- Importação dos Trechos textuais;

- Visualização dos trechos textuais;
- Criação de códigos;
- Edição de códigos;
- Exclusão de códigos;
- Finalização de codificação;
- Análise da codificação.

A *Open Code Tool* pode ser melhor acessada no link localizado no Rodapé[1].

4.2 Execução do Processo

Em relação a essa seção, é exposto como foi realizado a execução das atividades que compõem as fases de codificação aberta e codificação axial, onde estão expostas à partir da seção 4.1.1.

4.2.1 Coleta e organização dos dados

A Tabela 13 foi elaborada para mostrar os resultados obtidos em cada base de dados, utilizando as respectivas versões da string de busca mostradas na Tabela 11.

Tabela 13 – Quantidade de trabalhos retornados nas bases de dados

Base de dados	**Número de trabalhos**
ACM Digital Library	944
IEEE Xplore	478
Science Direct	6.140

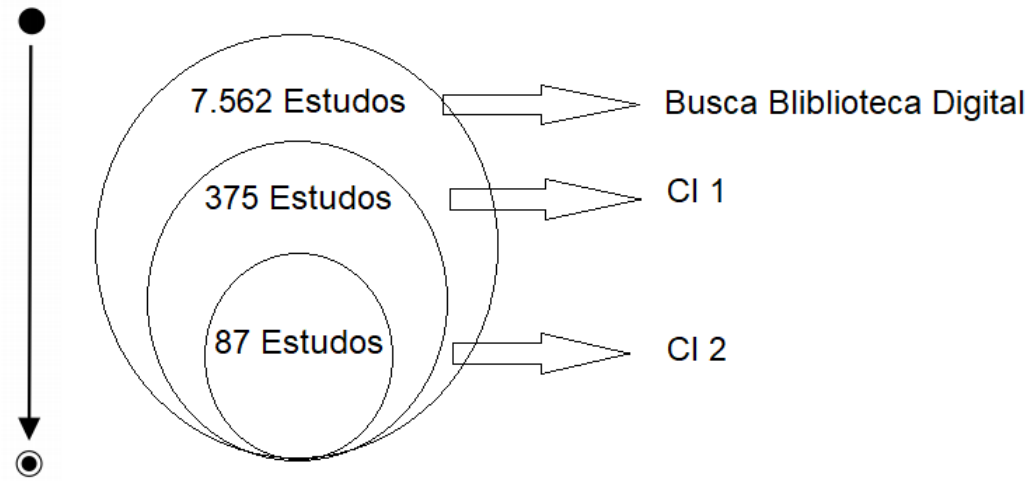
Fonte: O Autor.

[1] <http://200.132.136.14/legacysystemreport/public/>

Totalizou-se 7.562 artigos retornados nesta primeira pesquisa nas bases digitais.

Assim sendo, após as aplicações dos critérios mostrados na seção 4.1.1.5, obteve-se como resultado um número de 375 artigos após a aplicação do CI1 e após a aplicação do CI2 foi obtido o resultado final de 87 estudos, como pode ser visualizado na Figura 9.

Figura 9 – Número estudos pós triagem dos dados



Fonte: O Autor.

Dos 87 estudos selecionados no mapeamento, obteve-se um total de 30 trechos de definições sobre sistemas legados, 30 trechos com problemas e 61 trechos com possíveis estratégias, conforme o exposto no link disponibilizado para acesso aos trechos textuais.

4.2.1.1 Extração dos dados

Conforme a seção 4.1.1.6 é possível visualizar alguns trechos na Tabela 14. Todos os demais trechos encontram-se no link do Rodapé[2].

[2] <https://docs.google.com/spreadsheets/d/1eDWVi4df6aHpk5hXoY_IDm91MGFs7T3kVeg9atL32_4/edit?usp=sharing>

Tabela 14 – Exemplos trechos extraídos.

Referência	**Trecho**
Zhang et al. (2018)	"Component-aging is unavoidable in legacy systems"
Zhang e Yu (2018)	"In a legacy system, component-aging is unavoidable"
Khanye, Ophoff e Johnston (2018)	"Reference [x] defines legacy IS as any information system that significantly resists modification and evolution"
Oliveira, Vargas e Rodrigues (2018)	"Legacy systems are commonly based on obsolete technologies that can make it difficult to reuse SOA"
Knol e Tan (2018)	"Legacy systems are large software systems that we don't know how to cope with but that are vital to our organization [and they are typically] written in assembly or an early version of a third-generation language"

Fonte: O Autor.

4.2.2 Análise de dados

Nesta seção, é apresentada a execução das atividades expostas na seção 4.1.2.

4.2.3 Identificar Códigos

Conforme já exposto na seção 4.1.2.2, a codificação colaborativa foi elaborada com intuito de tornar mais confiável a fase de codificação. Assim, não foram apenas 2 pesquisadores analisando os trechos e extraindo códigos, e sim um número maior de pessoas (codificadores) analisando e identificando códigos. O maior benefício desta forma é permitir analisar trechos e identificar códigos de uma forma mais consensual.

O processo de codificação colaborativa ficou definido desta maneira: seleção de participantes, convite aos participantes, separação dos participantes em grupos, elaboração de material explicativo, o desenvolvimento da codificação colaborativa e por fim a análise dos resultados, conforme ilustrado na Figura 7.

4.2.3.1 Seleção de Participantes

Os participantes foram escolhidos de acordo com sua experiência acadêmica. Preferencialmente os mesmos deviam já ter cursado determinadas disciplinas. Os participantes do curso de Engenharia de Software já deveriam ter cursardo a disciplina de Evolução de Software e os participantes do curso de Ciência da Computação a disciplina de Engenharia de Software 2.

Partiu-se do princípio que, por serem veteranos em seus cursos ou já serem formados, os mesmos possuem maior senso crítico para a avaliação dos trechos pois já possuem um entendimento básico sobre sistemas legados. Assim, ao ler trechos textuais na língua inglesa, têm maior facilidade em analisá-los e extrair o que é necessário para o estudo.

4.2.3.2 Convite aos Participantes

Foi realizado um convite formal para os possíveis participantes. Assim que a resposta veio individualmente, foi contabilizado o número de aceitações. Foram obtidas 14 aceitações de colaboração nesta atividade.

4.2.3.3 Separação dos Participantes em Grupos

Os participantes foram separados em grupos homogêneos, os quais são descritos na Tabela 15.

Tabela 15 – Relação de grupos para codificação colaborativa.

Grupo	Número de participantes	Quantidade de Trechos
Grupo A	3	31
Grupo B	3	30
Grupo C	2	30
Grupo D	3	15
Grupo E	3	15

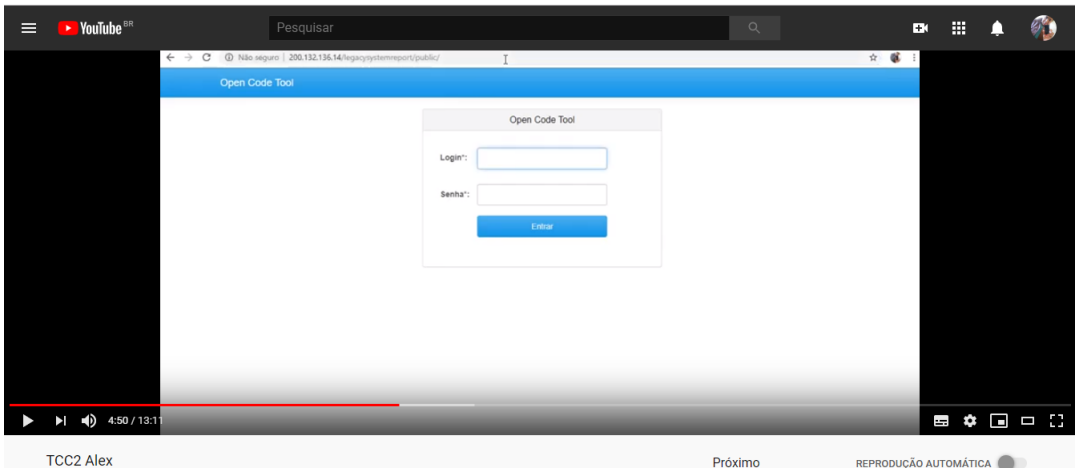
Fonte: O Autor.

4.2.3.4 Elaboração de Material Explicativo

Foi elaborado um material explicativo, do tipo vídeo, sobre os procedimentos que deviam ser seguidos para a realização do processo de codificação colaborativa. Esse vídeo foi disponibilizado para cada participante através de um *link* na plataforma *Youtube*, sendo visualizado Figura 10. O link para acesso está disponibilizado no Rodapé[3].

[3] <https://www.youtube.com/watch?v=KA5Dw49qK88&t=290s>

Figura 10 – Vídeo explicativo dos procedimentos de codificação.



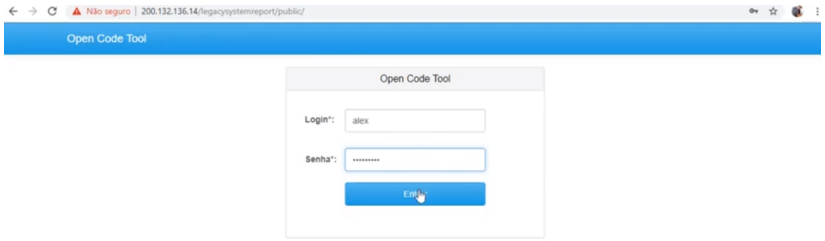
Fonte: O Autor.

No vídeo, também foram expostas explicações teóricas do processo, ou seja, o que são **trechos textuais** e o que são **códigos**, exemplos de identificação de códigos, bem como uma breve demonstração do uso da aplicação, de cada funcionalidade, para que assim cada participante fosse instruído de uma forma didática.

4.2.3.5 Realizar Codificação Colaborativa

Na seção 4.1.2.3, é explicado que a codificação colaborativa foi realizada através da ferramenta *Open Code Tool*. Para acessar a aplicação cada participante recebeu um *login* e senha, conforme exposto na Figura 11.

Figura 11 – Imagem do acesso à aplicação.

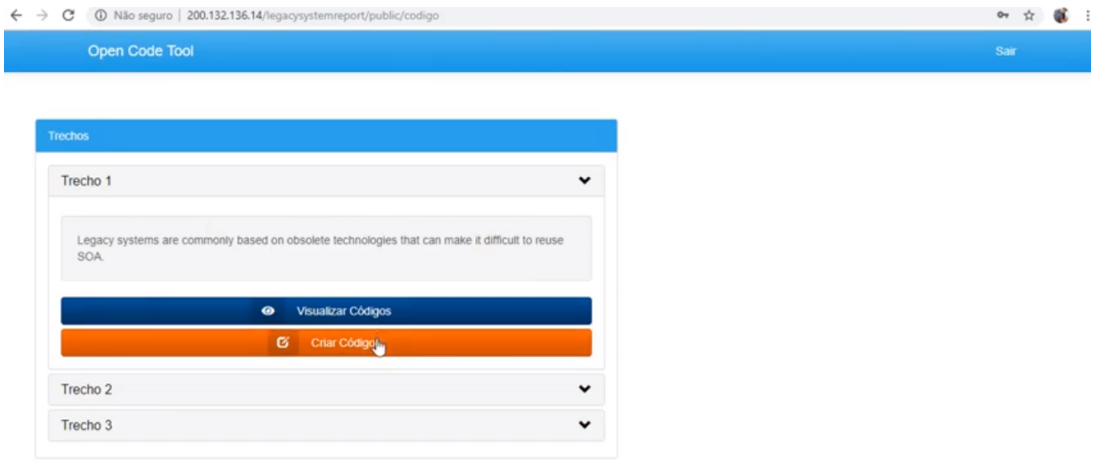


Fonte: O Autor.

Após realizado o acesso, conforme exposto na Tabela 15, cada participante foi

alocado a um grupo, sendo que a esse grupo já havia sido alocado um conjunto de trechos textuais.

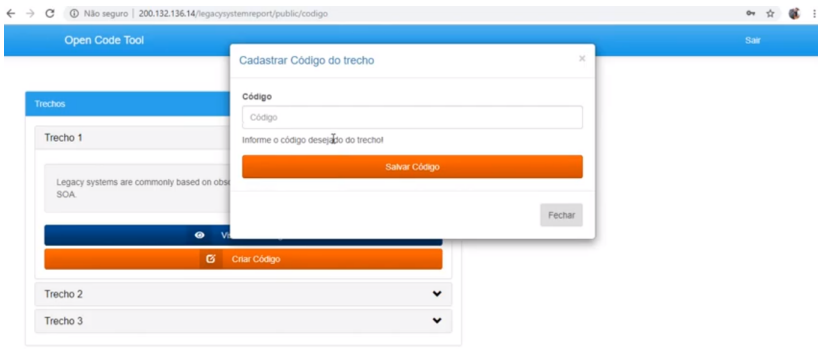
Figura 12 – Imagem dos trechos textuais.



Fonte: O Autor.

Como exposto na Figura 12, o participante tinha a oportunidade de analisar cada trecho para, assim, começar a criar os códigos referentes à sua visão analítica. Seguindo esse exemplo da Figura 12, caso o participante identificasse um código, este deveria selecionar a opção "Criar Código".

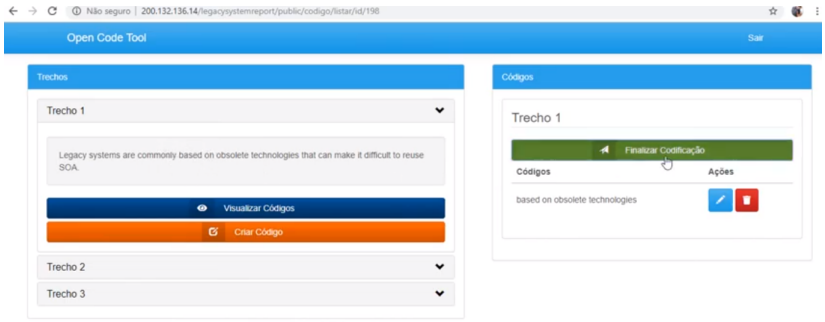
Figura 13 – Criação de código.



Fonte: O Autor.

Como mostrado na Figura 13, o participante codificador, a partir de cada análise individual, criava um ou mais códigos relacionados aquele trecho textual. Cada código criado deveria ser salvo conforme explicitado na Figura 14.

Figura 14 – Salvar código.



Fonte: O Autor.

Vale destacar que a construção dessa aplicação *web* possibilitou a todos os participantes a oportunidade de acessarem a ferramenta e realizarem o processo de criação de códigos de qualquer lugar e em qualquer momento que desejassem. Por exemplo, caso o participante estivesse trabalhando em um trecho "x" e precisasse fechar a ferramenta, quando retornasse a acessá-la, seu trabalho estaria onde o participante parou. Conforme o término das análises textuais e criações de códigos, cada particpante finalizava as codificações. Assim, os códigos criados eram salvos e foram analisados e agrupados conforme explicitado posteriormente.

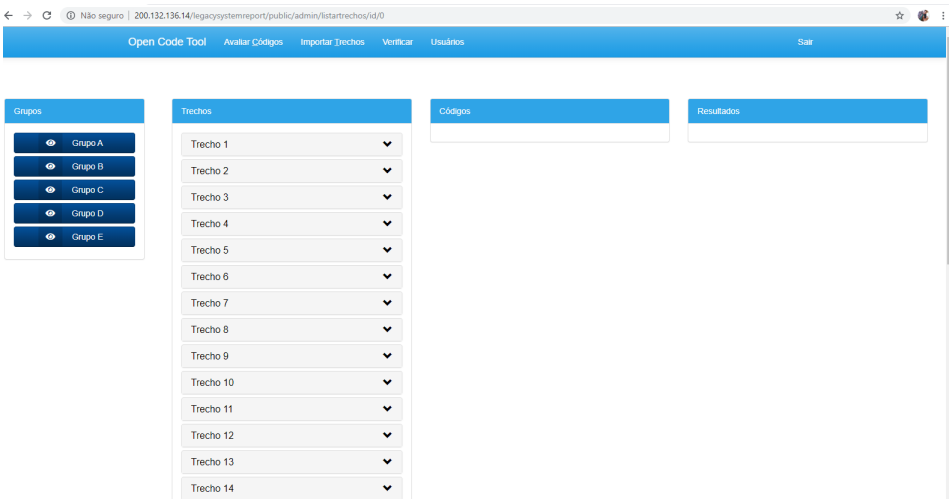
- Analisar resultados

A análise dos códigos foi realizada por avaliadores especialistas no domínio, através da aplicação web *Open Code Tool*.

Como mostra a Figura 15 os avaliadores possuem um perfil (admin), onde era possível analisar os códigos criados em cada trecho, por cada participante.

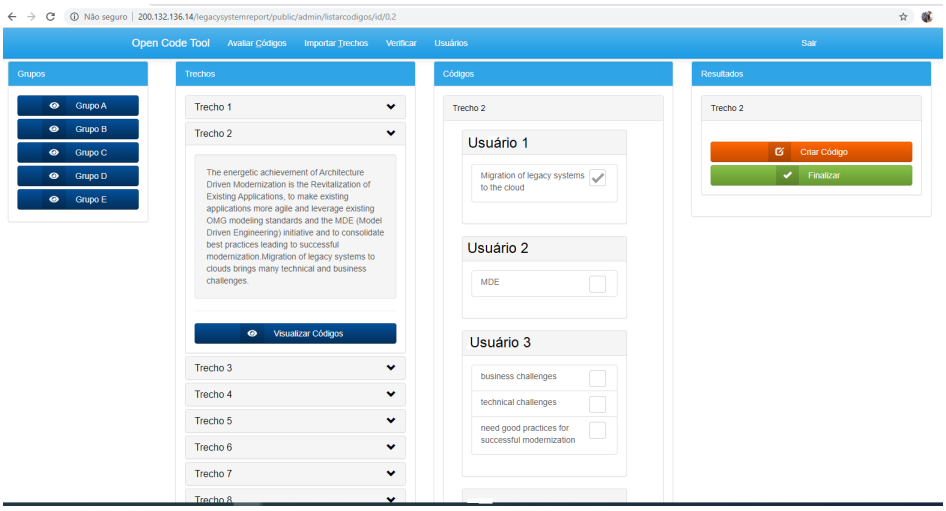
Após a escolha do grupo a ser avaliado, os trechos alocados ao grupo eram visualizados. Os avaliadores selecionavam o trecho que seria analisado e escolhiam a opção "Visualizar Códigos". A aplicação retornava **todos** os códigos criados por **todos** todos os participantes para um determinado trecho. Cabe salientar que, conforme a Figura 16 este processo era único para cada **trecho textual**, ou seja, se o avaliador selecionasse o trecho textual 2, a aplicação retornava os códigos criados por todos os participantes para o trecho 2.

Figura 15 – Tela inicial avaliador.



Fonte: O Autor.

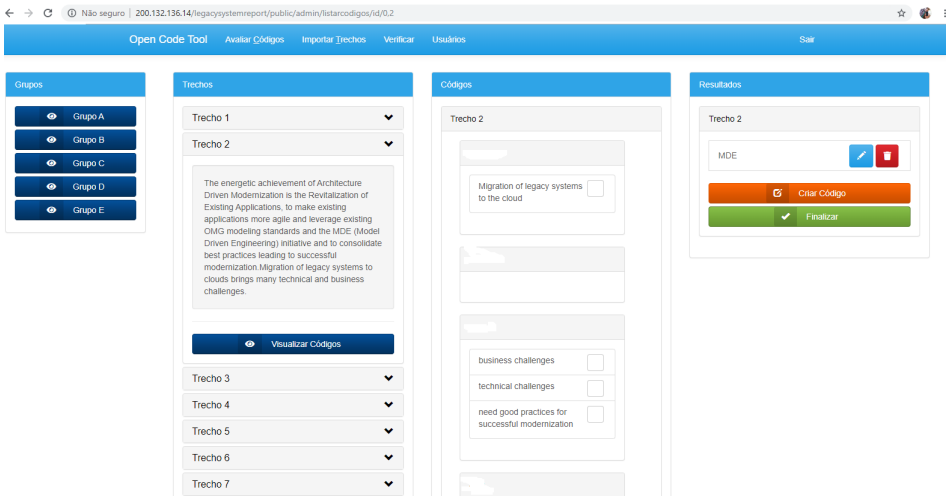
Figura 16 – Selecionar grupo e trecho textual.



Fonte: O Autor.

Os avaliadores tinham a opção de aceitar (selecionando o *check box*) ou não cada código criado. Os avaliadores também podiam criar códigos de acordo com suas análises, bem como editar códigos já criados pelos participantes. A Figura 17 demonstra esse processo.

Figura 17 – Selecionar códigos finais.



Fonte: O Autor.

A Figura 18 mostra um *snapshot* do resultado da análise dos códigos, apresentando 5 exemplos de trechos textuais e os códigos encontrados e analisados para cada trecho. Vale salientar, que não foram aprovados todos os códigos criados pelos participantes. Os códigos eram avaliados pelos especialistas, para verificar quais eram mais corretos de acordo com o trecho em que os mesmos foram criados.

Figura 18 – Tabela contendo trecho textual e códigos criados.

	A	B	C
1	**ID Trecho**	**Trecho**	**código**
2	92	Component-aging is unavoidable in legacy systems.	component-aging unavoidable
3	93	In a legacy system, component-aging is unavoidable.	component-aging is unavoidable
4	94	Reference [x] defines legacy IS as any information system that significantly resists modification and evolution.	resists modification and evolution
5	95	Legacy systems are commonly based on obsolete technologies that can make it difficult to reuse SOA.	based tecnolge obsolete
6	96	Legacy systems are large software systems that we don't know how to cope but that are vital to our organization [and they are typically] written in assembly or an early version of a third-generation language.	large software systems
7			vital to our organization
8			typically written in assembly
9			written in a third-generation language

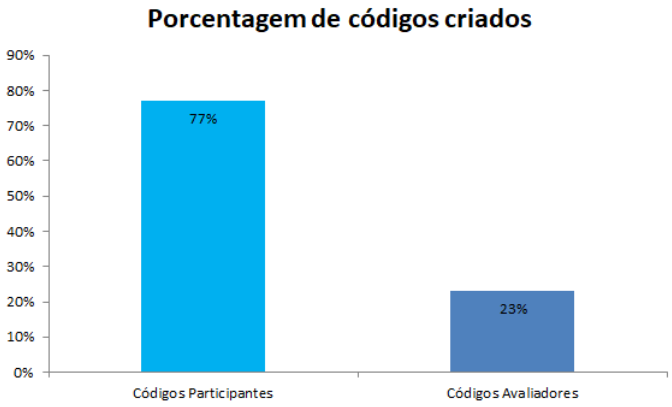
Fonte: O Autor.

A coluna **ID Trecho** contém o identificador daquele trecho dentro da aplicação *Open Code Tool*. Na coluna **Trecho** contém o trecho textual de acordo com o seu tipo (definição, problema ou estratégia). Na coluna **código** são representados os códigos ou palavras que melhor expressam o conteúdo escrito pelo autor, na visão dos participantes. O restante dos códigos extraídos estão disponibilizados no *link* do Rodapé[4].

[4] <https://docs.google.com/spreadsheets/d/1KN3bgTs5zF9c3KbUFpzTMzJmAYLfkaeskCl2vXyuLkU/

Obteve-se um total de 237 códigos avaliados e compilados, desse total cerca de 23% foram códigos criados pelos avaliadores, conforme explicado anteriormente. Na Figura 19, é exposto esse resultado. Assim, é demonstrada a forte contribuição da codificação colaborativa para o desenvolvimento desse trabalho, trazendo assim um dos objetivos expostos, de analisar e obter códigos de uma forma mais consensual.

Figura 19 – Resultado codificação colaborativa.



Fonte: O Autor.

4.2.4 Identificar agrupamentos

Conforme o explicitado na seção 4.1.2.4, encontra-se nessa seção os resultados obtidos nesta atividade. Cabe salientar que foi possível definir 4 tipos de agrupamentos, são eles:

- Grupo: é o agrupamento que está ligado diretamente com o **fenômeno**, no caso deste estudo os grupos que possuem essa ligação são definições, problemas e estratégias.
- Subgrupo nível 1: são os agrupamentos que contém ligação direta com o grupo, que está um nível acima.
- Subgrupo nível 2: são os agrupamentos que contém ligação direta com o Subgrupo nível 1, que está um nível acima.
- Apenas códigos: são apenas códigos que não foram agrupados como subgrupo, por não possuírem frequência acima de 1. Entretanto, possuem ligação direta com qualquer subgrupo.

Assim, a Figura 20 demonstra que os agrupamentos podem ser nomeados de acordo com a análise do pesquisador.

edit?usp=sharing>

Figura 20 – Atividade de Agrupamento de códigos

	A	B	C	D	E
1		código (ingles)	SubGrupo- nivel 2	SubGrupo Nivel 1	Grupo definição
2	92	component-aging unavoidable		Obsolete Technology	
3	93	component-aging is unavoidable		Obsolete Technology	
4	94	resists modification and evolution		Hard to maintain	
5	95	based in obsolete tecnology		Obsolete Technology	
6	96	large software systems		Large system	
7		vital to our organization		Vital to the Organization	
8		typically written in assembly	Obsolete language	Obsolete Technology	
9		written in a third-generation language	Obsolete language	Obsolete Technology	

Fonte: O Autor.

Nesta etapa da codificação aberta foi possível encontrar alguns agrupamentos distintos, são eles:

4.2.4.1 Grupo Definições

O grupo **definições** foi construído para agrupar códigos que caracterizavam legados de alguma forma. Em relação aos elementos que compõem esse grupo, foi possível encontrar os seguintes subgrupos:

- ***Obsolete Technology***: Este foi o subgrupo nível 1 encontrado com mais frequência dentre os dados oriundos deste grupo de definições. Cerca de 30% dos códigos analisados possuem características relacionadas ao uso ou construção com tecnologia obsoleta.

  Vale ressaltar que ligados a este subgrupo foram identificados subgrupos nível 2, que possuem códigos que denotam características mais específicas deste subgrupo nível 1, são eles: *Obsolete language*, subgrupo nível 2 que contém códigos relacionados ao uso de linguagens de programação obsoletas em sistemas legados. *Obsolete paradigm*, subgrupo nível 2 que contém códigos relacionados, por exemplo, a procedimentos de programação.

- ***Old system***: Este subgrupo nível 1 é relacionado a códigos que enfatizam o tempo como fator de identificação de sistemas legados. Cerca de 15% dos códigos analisados possuem características relacionadas a este subgrupo.

- ***Vital to the Organization***: Este subgrupo nível 1 é relacionado à códigos que caracterizam o sistema legado sendo vital à organização como fator determinante na identificação desse tipo de sistema. Cerca de 9% dos códigos analisados possuem características relacionadas a este subgrupo.

- ***Currently Used***: Este subgrupo nível 1 é relacionado a códigos que caracterizam o sistema legado como sendo utilizado atualmente, assim tratando esta característica

como fator determinante na identificação desse tipo de sistema. Cerca de 7% dos códigos analisados possuem características relacionadas a este subgrupo.

- ***Large system***: Este subgrupo nível 1 é relacionado a códigos que caracterizam o sistema legado como sendo um sistema grande, possuindo códigos que denotam a quantidade excessiva de linhas de código, de componentes etc. Cerca de 4% dos códigos analisados possuem características relacionadas a este subgrupo.

- ***Hard to maintain***: Este subgrupo nível 1 é relacionado à códigos que caracterizam a dificuldade que existe em manter um sistema legado como fator determinante na identificação desse tipo de sistema. Cerca de 3% dos códigos analisados possuem características relacionadas à este subgrupo.

  O restante dos códigos encontrados neste grupo de definições não foram agrupados, pois não possuem frequência acima de 1, como explicado no início desta seção. Cerca de 20% dos códigos analisados se encaixam nessa situação de não possuir agrupamento em algum subgrupo seja nível 1 ou 2. Entretanto, todos estão ligador diretamente ao grupo definição, e se encaixam nas ligações ***type, consequence, etc.***

4.2.4.2 Grupo Problemas

O grupo **Problemas** foi construído para agrupar códigos identificados como problemas de um sistema ser legado, problemas que o sistema causa por ser legado, etc. Em relação aos elementos que compõem o grupo esse grupo, foi possível encontrar os seguintes subgrupos:

- ***Economic value***: Este subgrupo nível 1, que teve mais frequência dentre os encontrados neste grupo de problemas, está relacionado a códigos que caracterizam o valor econômico que um sistema legado possui como um problema que esse tipo de sistema. Cerca de 8% dos códigos analisados possuem características relacionadas à este subgrupo.

- ***Documentation***: Este subgrupo nível 1 contém códigos relacionados à documentação, caracterizando o grupo como um tipo de problema causado por sistema legados. Cerca de 6% dos códigos analisados possuem características relacionadas a este subgrupo.

- ***Maintenance***: Este subgrupo nível 1 está relacionado a códigos que denotam a manutenção que o sistema legado necessita como um problema que esse tipo de sistema possui. Cerca de 6% dos códigos analisados possuem características relacionadas a este subgrupo.

- ***Old hardware***: Este subgrupo nível 1 contém uma problemática relacionada ao uso de hardware antigo em sistemas legados, ocasionando sintomas negativos nesses tipos de sistemas. Cerca de 3% dos códigos analisados possuem características relacionadas à este subgrupo.

  O restante dos códigos encontrados neste grupo de problemas não foram agrupados pois, seguindo o caso anterior explicitado na seção 4.2.4.1, não possuem frequência acima de 1. Cerca de 70% dos códigos analisados se encaixam nessa situação de não possuir agrupamento em algum subgrupo seja nível 1 ou 2. Entretanto todos estão ligados diretamente ao grupo problemas ou em algum dos subgrupos nível 1 se encaixando nas ligações ***type, consequence, etc.***

4.2.4.3 Grupo Estratégias

O grupo **Estratégias** foi elaborado para agrupar códigos que relatavam possíveis estratégias que podem ser tomadas para evoluir um sistema considerado legado, bem como alguns processos para realização de tais estratégias, benefícios e consequências que podem ser obtidas ao tomar certas decisões. Em relação aos elementos que compõem o grupo esse grupo, foi possível encontrar os seguintes subgrupos:

- ***Migration***: Este subgrupo nível 1 é o que teve mais frequência dentre os encontrados neste grupo de estratégias, está relacionado a códigos que demonstram que o uso de migração de sistema legado é uma possível estratégia que deve ser levada em consideração para evoluir tais sistemas. Cerca de 30% dos códigos analisados possuem características relacionadas a este subgrupo. ***Cloud*** foi o tipo de migração mais citada entre esta estratégia, obtendo cerca de 28% dos códigos ligados diretamente à este subgrupo nível 1, assim, se tornando um subgrupo nível 2 que está ligado à *migration*.

- ***Replacement***: Este subgrupo nível 1 possui códigos que estão relacionados à estratégia do tipo substituição, ou seja, denotam que substituir um sistema legado, ou partes do mesmo, é uma estratégia viável para evolução de tais sistemas . Cerca de 9% dos códigos analisados possuem características relacionadas a este subgrupo. Sendo ***Obsolete Component*** o código mais encontrado, assim se tornando um subgrupo nível 2 que está ligado diretamente ao nível 1, demonstrando que substituição de componentes obsoletos um tipo mais viável de substituição em legados.

- ***Integration***: Este subgrupo nível 1 contém códigos relacionados a estratégias de integração de legados à novas tecnologias, processos, sistemas e etc. Cerca de 6% dos códigos analisados possuem características relacionadas à este subgrupo.

- ***Conversion***: Este subgrupo nível 1 contém códigos relacionados há estratégias de conversão de legados novos protocolos, tipos de linguagens, etc. Cerca de 2% dos códigos analisados possuem características relacionadas à este subgrupo.

- ***Prefer to start fresh.***: Este subgrupo nível 1 contém como tipo de estratégia a criação de um sistema novo no lugar de sistemas legados. Cerca de apenas 1% dos códigos analisados possuem características relacionadas à este subgrupo. Sendo assim, é possível notar que continuar mantendo um sistema legado é mais vantajoso que criar um do zero.

  O restante dos códigos encontrados neste grupo de estratégias não foram agrupados pois, seguindo os casos explicitados na seção 4.2.4.1, não possuem frequência acima de 1. Cerca de 45% dos códigos analisados se encaixam nessa situação de não possuir agrupamento em algum subgrupo seja nível 1 ou 2. Entretanto todos estão ligados diretamente ao grupo problemas ou em algum dos subgrupos nível 1 se encaixando nas ligações ***type, consequence, etc.***

4.2.4.4 Identificar tipos de código

Como explicitado na seção 4.1.2.6, essa atividade contempla a identificação dos tipos de códigos criados. Assim, os tipos de códigos definidos para esse estudo são ***Definition, Problem, Strategy***.

Definition caracteriza ou influencia um fenômeno (sistema legado), sendo algumas definições agentes causadores de *problem*, e *Strategy* são geradas para resolver *problems* de evolução de sistemas legados.

4.2.4.5 Identificação de relações entre os agrupamentos de conceitos

Conforme explicitado na seção 4.1.2.7, essa atividade contempla a identificação de relações entre os grupos, subgrupos e códigos. Assim, essas relações receberam uma nomenclatura de acordo com a percepção dos avaliadores. Foram criadas tais nomenclaturas para as relações:

- ***Consequence***: esta relação foi concebida para relacionar quando um elemento é consequência de outro elemento.

- ***Type***: esta relação foi concebida para relacionar quando um elemento é tipo de outro elemento.

- ***Benefits***: esta relação foi concebida para relacionar quando um elemento é benefício de outro elemento.

- ***Cause***: esta relação foi concebida para relacionar quando um elemento é o causador de outro elemento.

- ***Has process***: esta relação foi concebida para relacionar quando uma sentença foi descrita como processo de realização de algum elemento. elemento.

4.3 Ameaças à validade

Conforme o desenvolvimento deste trabalho foi possível identificar algumas ameaças a sua validade, são elas:

- Desenvolvimento dos procedimentos da GT: Por esse trabalho possuir procedimentos de um método de codificação qualitativa, até então pouco conhecido no âmbito da computação, realizou-se um estudo exaustivo sobre sua origem, vertentes, processos de desenvolvimento como explicitado na seção 2.3. Contudo, sabe-se que mesmo com estes estudos exaustivos, pode-se haver melhorias a serem feitas, entendimentos que podem ter sido aplicados de forma equivocada.

- Strings de busca: Conforme exposto na seção 4.1.1.4, foram criados 3 modelos de string de busca, cada uma de acordo com os parâmetros de cada base digital. Entretanto, na base digital *Science Direct* houve alterações na string pois há limitações no mecanismo de busca da base digital. Assim, sabe-se que por este motivo trabalhos podem ter ficado fora da gama de estudos que foram analisados conforme no MSL.

- Participantes: Conforme exposto na primeira atividade descrita na subseção 4.1.2.2, os participantes foram escolhidos com a premissa de possuirem maior experiência acadêmica. Entretanto, sabe-se que esta característica não é suficiente para afirmar que realmente possuem tal experiência e que a mesma os apoiou no desenvolvimento da codificação.

5 RESULTADOS

Neste Capítulo são explicitados os resultados obtidos através da aplicação de procedimentos oriundos da GT com o intuito de oferecer um entendimento mais conciso sobre os elementos que envolvem sistemas legados. Este capítulo está separado de acordo com os elementos que foram analisados neste trabalho, são eles: **Definições**, **Problemas** e **Estratégias**. Através do link disponibilizado no Rodapé[1] é possível visualizar os mapas conceituais.

5.1 Apresentação do modelo visual de definições

Em relação às codificações e análises, para o agrupamento de ***Definitions*** foi construído um modelo visual que demonstra os dados inicialmente explicados na seção 4.2.4.1. O mesmo pode ser melhor visualizado na Figura abaixo.

De acordo com o modelo é possível obter algumas análises. O subgrupo nível 1 que caracterizou a tecnologia obsoleta como característica de um legado, subgrupo este que obteve maior frequência, demonstrou o uso de linguagens obsoletas (subgrupo nível 2) como maior parte destas tecnologias, expondo modelos de linguagens específicas como **Cobol**, **Fortran**, **C**, **C++**, **Assembly**.

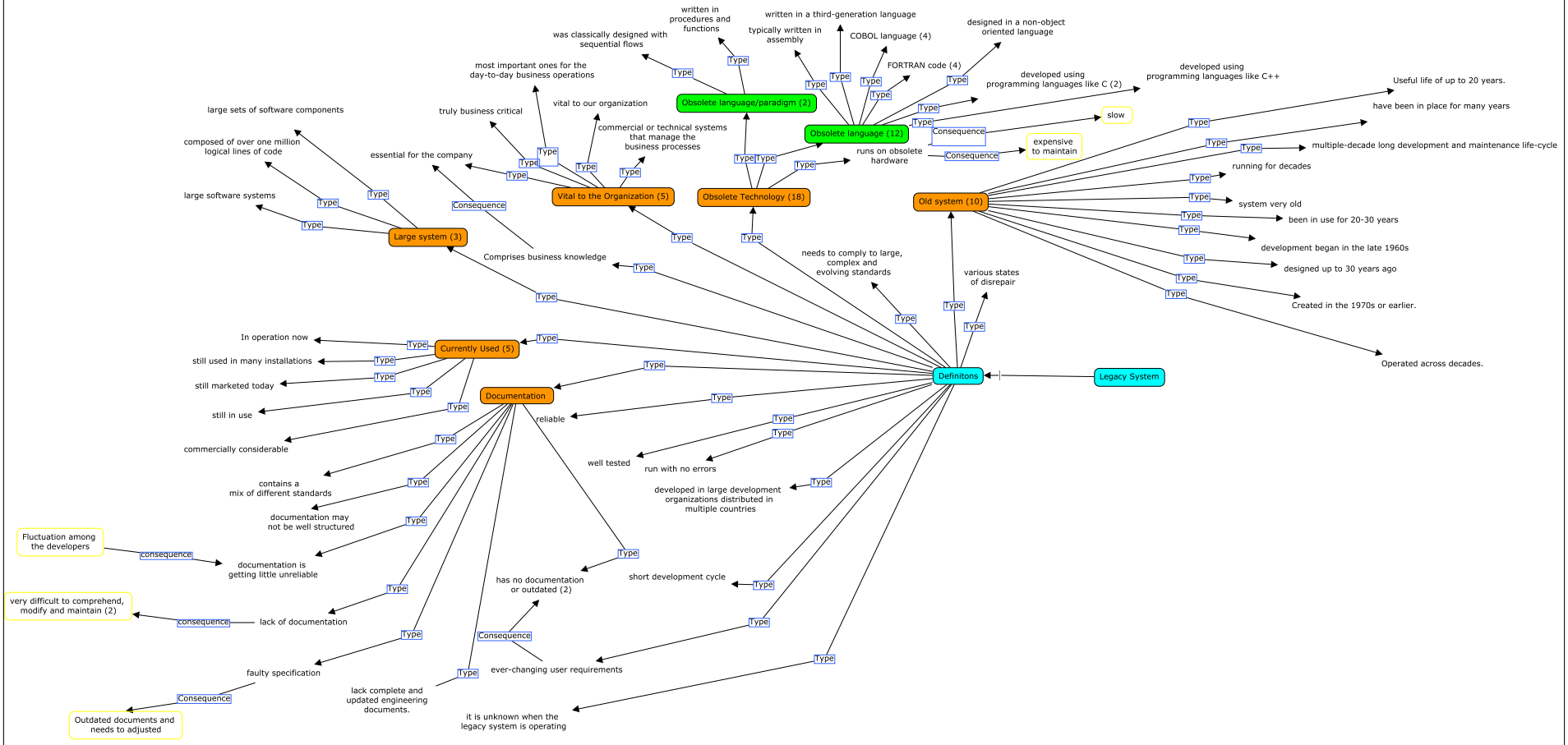
O subgrupo nível 1, que caracteriza um sistema ser antigo como legado, demonstra que um tempo de no mínimo 20 anos de sua concepção já leva que um sistema seja identficado como legado. Outra característica identificada que além de serem sistemas antigos, os mesmos ainda estão em operação atualmente. Assim como o tamanho que o sistema possui, seja por composição de linhas de código ou por quantidade de componentes que esses sistemas possuem.

A documentação também foi uma característica muito citada nas análises, assim demonstrando que existe uma preocupação em relação a quão bem está a documentação de um sistema, caso a mesma esteja incompleta, desatualizada ou até mesmo seja inexistente já caracteriza um sistema como legado.

A característica que denota a importância que o sistema possui em sua organização também é explicitada nesse modelo, essas características foram agrupadas no subgrupo vital à organização. Assim foi possível visualizar que a criticidade que o sistema possui em relação aos processos de negócio é levada em consideração na hora de identificar um legado.

Vale salientar que exitem alguns códigos que estão neste modelo, porém foram identificados como tipos de códigos dentro do grupo de problemas e estão aqui explicitados também para demonstrarem as relações que existem entre os grandes grupos (explicitados na seção 5.4), é possível identificá-los pois os mesmos estão envoltos de uma linha amarela.

[1] <https://drive.google.com/drive/folders/1Mxi4JxU5NvPtijhugYXIn9CV0bt-VhjK?usp=sharing>


Definitons
Legacy System
Old system (10)
Useful life of up to 20 years.
have been in place for many years
multiple-decade long development and maintenance life-cycle
running for decades
system very old
been in use for 20-30 years
development began in the late 1960s
designed up to 30 years ago
Created in the 1970s or earlier.
Operated across decades.
Obsolete Technology (18)
Obsolete language (12)
slow
expensive to maintain
runs on obsolete hardware
Consequence
Obsolete language/paradigm (2)
written in procedures and functions
was classically designed with sequential flows
written in a third-generation language
typically written in assembly
COBOL language (4)
FORTRAN code (4)
designed in a non-object oriented language
developed using programming languages like C (2)
developed using programming languages like C++
Vital to the Organization (5)
most important ones for the day-to-day business operations
truly business critical
vital to our organization
commercial or technical systems that manage the business processes
Large system (3)
large sets of software components
composed of over one million logical lines of code
large software systems
essential for the company
Comprises business knowledge
needs to comply to large, complex and evolving standards
various states of disrepair
Currently Used (5)
In operation now
still used in many installations
still marketed today
still in use
commercially considerable
Documentation
reliable
well tested
run with no errors
developed in large development organizations distributed in multiple countries
short development cycle
has no documentation or outdated (2)
ever-changing user requirements
lack complete and updated engineering documents.
it is unknown when the legacy system is operating
lack of documentation
very difficult to comprehend, modify and maintain (2)
documentation is getting little unreliable
Fluctuation among the developers
documentation may not be well structured
contains a mix of different standards
faulty specification
Outdated documents and needs to adjusted
Type
consequence

5.2 Apresentação do modelo visual de problemas

Em relação às codificações e análises, para o agrupamento de ***Problems*** foi construído um modelo visual que demonstra os dados inicialmente explicados na Seção 4.2.4.2.

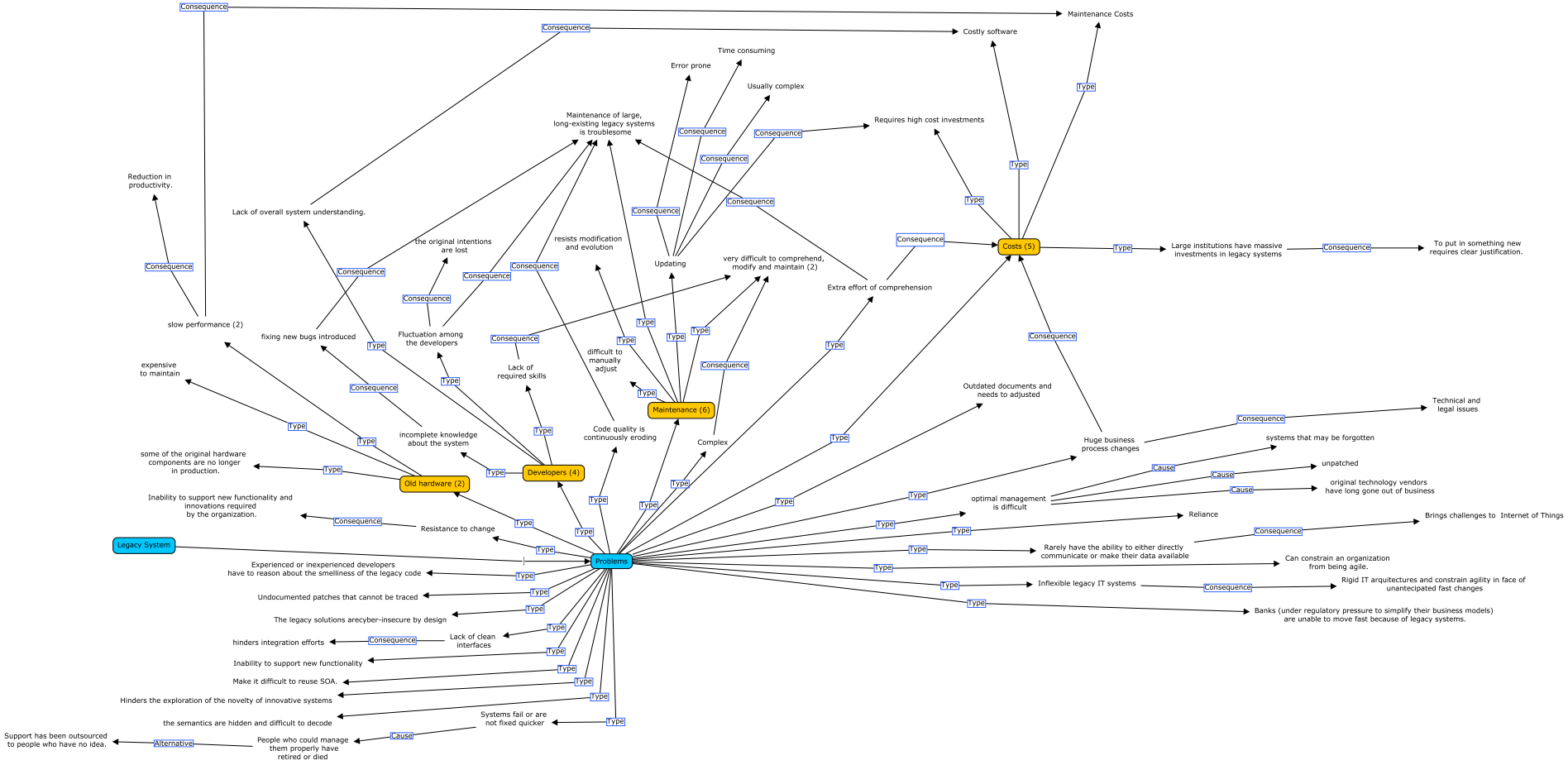
De acordo com o modelo é possível identificar alguns aspectos que envolvem tais elementos. A manutenção que um legado necessita é um problema que possuiu alta frequência de apontamentos principalmente pela mesma ser complexa e de difícil realização, por necessitar de altos investimentos financeiros, por consumir um tempo excessivo de realização, etc.

Os custos financeiros que um legado demanda à sua organização não são apenas por manutenções necessárias, esses custos também são necessários pois legados necessitam muitas vezes de esforço extra de compreensão, por necessitar de grandes mudanças nos processos de negócios, entre outros.

Um problema também identificado é em relação as necessidades de não existir recursos humanos capacitados para se envolverem com legados, pois existem muitos casos de flutuação de desenvolvedores que acabam causando que intenções originais do sistema sejam perdidas com o tempo, manutenções mais caras financeiramente, além de ocasionarem uma maior dificuldade em compreender as intenções do sistema, modificar e manter os mesmos.

O hardware de legados ser considerado antigo também foi identificado como problema. Pois o mesmo acaba ocasionando consequências como perfomance lento causando redução de produtividade, e muitas vezes esse hardware não é mais produzido.

Outro elementos foram identificados como problemas existentes em legados, como por exemplo a inflexibilidade que tais sistemas possuem trazendo como consequência arquiteturas rígidas. possuírem códigos que vão obtendo erosões ao longo do ciclo de vida, retornando como consequência uma manutenção problemática, entre outros. O mesmo pode ser melhor visualizado na Figura abaixo.


Legacy System
Problems
Maintenance Costs
Costly software
Time consuming
Error prone
Usually complex
Maintenance of large, long-existing legacy systems is troublesome
Requires high cost investments
Reduction in productivity.
Lack of overall system understanding.
the original intentions are lost
resists modification and evolution
Updating
very difficult to comprehend, modify and maintain (2)
Extra effort of comprehension
Costs (5)
Large institutions have massive investments in legacy systems
To put in something new requires clear justification.
slow performance (2)
fixing new bugs introduced
Fluctuation among the developers
Lack of required skills
difficult to manually adjust
Maintenance (6)
Complex
Outdated documents and needs to adjusted
expensive to maintain
incomplete knowledge about the system
Code quality is continuously eroding
Huge business process changes
Technical and legal issues
systems that may be forgotten
unpatched
original technology vendors have long gone out of business
some of the original hardware components are no longer in production.
Old hardware (2)
Developers (4)
optimal management is difficult
Reliance
Brings challenges to Internet of Things
Inability to support new functionality and innovations required by the organization.
Resistance to change
Rarely have the ability to either directly communicate or make their data available
Can constrain an organization from being agile.
Experienced or inexperienced developers have to reason about the smelliness of the legacy code
Inflexible legacy IT systems
Rigid IT arquitectures and constrain agility in face of unantecipated fast changes
Undocumented patches that cannot be traced
Banks (under regulatory pressure to simplify their business models) are unable to move fast because of legacy systems.
The legacy solutions arecyber-insecure by design
hinders integration efforts
Lack of clean interfaces
Inability to support new functionality
Make it difficult to reuse SOA.
Hinders the exploration of the novelty of innovative systems
the semantics are hidden and difficult to decode
Systems fail or are not fixed quicker
Support has been outsourced to people who have no idea.
People who could manage them properly have retired or died
Consequence
Type
Cause
Alternative

5.3 Apresentação do modelo visual de estratégias

Em relação às codificações e análises, para o agrupamento de ***Strategy*** foi construído um modelo visual que demonstra os dados inicialmente explicitados na seção 4.2.4.3. O mesmo pode ser melhor visualizado na Figura abaixo.

Com a visualização dos dados e a construção do modelo visual foi possível identificar que a migração é a estratégia mais utilizada atualmente, contendo vários exemplos de como realizar a mesma. Em relação aos tipos de migração, a utilização da nuvem é a que mais frequentemente foi relatada nos estudos analisados. Identificou-se alguns modelos de estratégias para utilização da migração em nuvem (*e.g* paralelismo).

Ainda em relação à migração de legados para plataformas em nuvem, foi possível identificar possíveis consequências que podem acontecer quando este tipo de estratégias são escolhidas. O custo é uma dessas consequências, assim como os desafios que em relção ao gerenciamento bem como desafios técnicos e de negócios.

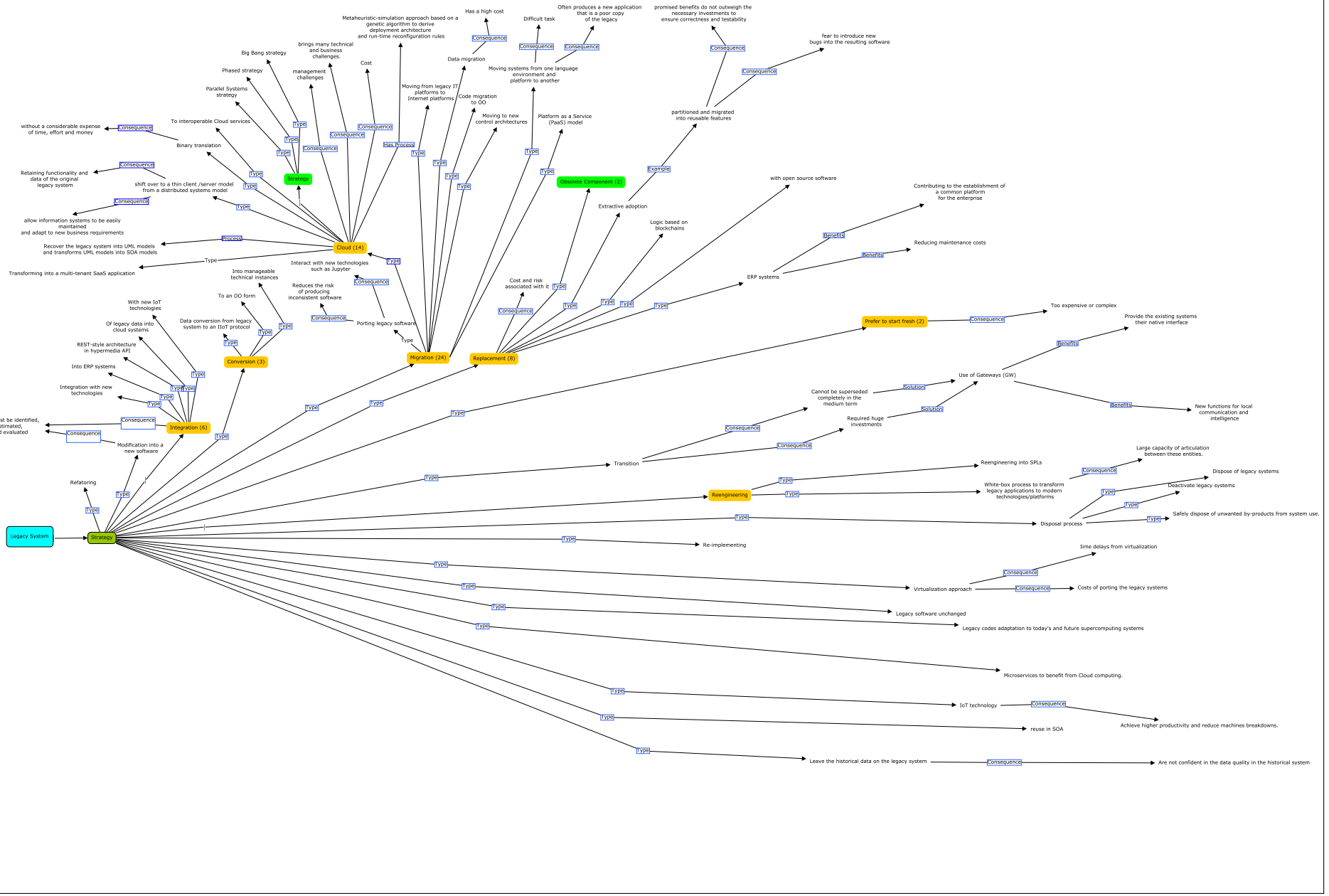
Foi possível identificar também processos que podem ser realizados para tais tarefas (*e.g* tranformação de modelos UML em modelos SOA).

Uma estratégia também relatada com alta frequência foi a substituição de legado. Alguns exemplos encontrados dessa estratégia são de apenas substituir componentes, substituir legados por lógicas baseadas em *blockchains*, substituir utilizando uma adoção extrativa como particionar um legado migrando o mesmo para recursos reutilizáveis. Substituir legados por software de código aberto, substuir legado por sistemas que integram todos os dados e processos da organização em um único sistema, conhecidos como sistemas *ERP - Enterprise Resource Planning*, retornando benefícios como redução nos custos de manutenções e contribuindo para o estabelecimento de uma plataforma comum para a empresa.

Integração de legados para algum elemento específico também obteve uma frequência considerável nesse estudo. Alguns exemplos que obteve-se desta estratégia são: integração de legados em novas tecnologias, sistemas *ERP*, tecnologias *IoT - Internet of Things*, integrar sistemas legados para nuvem (diferente de migrar), utilizar arquitetura no estilo REST na API hipermídia para realizar esta integração.

Converter um legado também foi um elemento identificado como estratégia de evolução, seja para formulário orientado a objetos, protocolos *IIoT - Industrial Internet of Things*.

Alguns tipos de estratégias foram pouco citados, mas são relevantes ao estudo, como: reimplementação, constreuir sistema do zero, refatorar, descartar, entre outros.


Legacy System
Strategy
Refatoring
Integration (6)
Risks must be identified, estimated, and evaluated
Modification into a new software
Integration with new technologies
Into ERP systems
REST-style architecture in hypermedia API
Of legacy data into cloud systems
With new IoT technologies
Conversion (3)
Data conversion from legacy system to an IIoT protocol
To an OO form
Into manageable technical instances
Cloud (14)
Recover the legacy system into UML models and transforms UML models into SOA models
Transforming into a multi-tenant SaaS application
allow information systems to be easily maintained and adapt to new business requirements
Retaining functionality and data of the original legacy system
shift over to a thin client /server model from a distributed systems model
Binary translation
without a considerable expense of time, effort and money
To interoperable Cloud services
Parallel Systems strategy
Phased strategy
Big Bang strategy
Strategy
brings many technical and business challenges.
management challenges
Cost
Metaheuristic-simulation approach based on a genetic algorithm to derive deployment architecture and run-time reconfiguration rules
Moving from legacy IT platforms to Internet platforms
Has Process
Interact with new technologies such as Jupyter
Reduces the risk of producing inconsistent software
Porting legacy software
Migration (24)
Code migration to OO
Moving to new control architectures
Data migration
Has a high cost
Moving systems from one language environment and platform to another
Difficult task
Often produces a new application that is a poor copy of the legacy
Platform as a Service (PaaS) model
Obsolete Component (2)
Cost and risk associated with it
Replacement (8)
Extractive adoption
Example
partitioned and migrated into reusable features
promised benefits do not outweigh the necessary investments to ensure correctness and testability
fear to introduce new bugs into the resulting software
Logic based on blockchains
with open source software
ERP systems
Benefits
Contributing to the establishment of a common platform for the enterprise
Reducing maintenance costs
Prefer to start fresh (2)
Too expensive or complex
Provide the existing systems their native interface
Use of Gateways (GW)
Benefits
New functions for local communication and intelligence
Solution
Cannot be superseded completely in the medium term
Required huge investments
Transition
Reengineering
Reengineering into SPLs
White-box process to transform legacy applications to modern technologies/platforms
Disposal process
Large capacity of articulation between these entities.
Dispose of legacy systems
Deactivate legacy systems
Safely dispose of unwanted by-products from system use.
Re-implementing
Virtualization approach
Iime delays from virtualization
Costs of porting the legacy systems
Legacy software unchanged
Legacy codes adaptation to today's and future supercomputing systems
Microservices to benefit from Cloud computing.
IoT technology
Achieve higher productivity and reduce machines breakdowns.
reuse in SOA
Leave the historical data on the legacy system
Are not confident in the data quality in the historical system
Type
Consequence
Process

5.4 Apresentação do modelo visual geral

Seguindo o exposto na seção 1, existe uma falta de consenso sobre os elementos que envolvem um sistema legado, devido ao fato de existir uma gama muito extensa de definições e características distintas na literatura.

Essa falta de consenso causa uma dificuldade de reconhecer quando um sistema é ou está se tornando legado, fazendo que na maioria das vezes esse tipo de sistema seja identificado apenas quando o mesmo já esta causando problemas à organização.

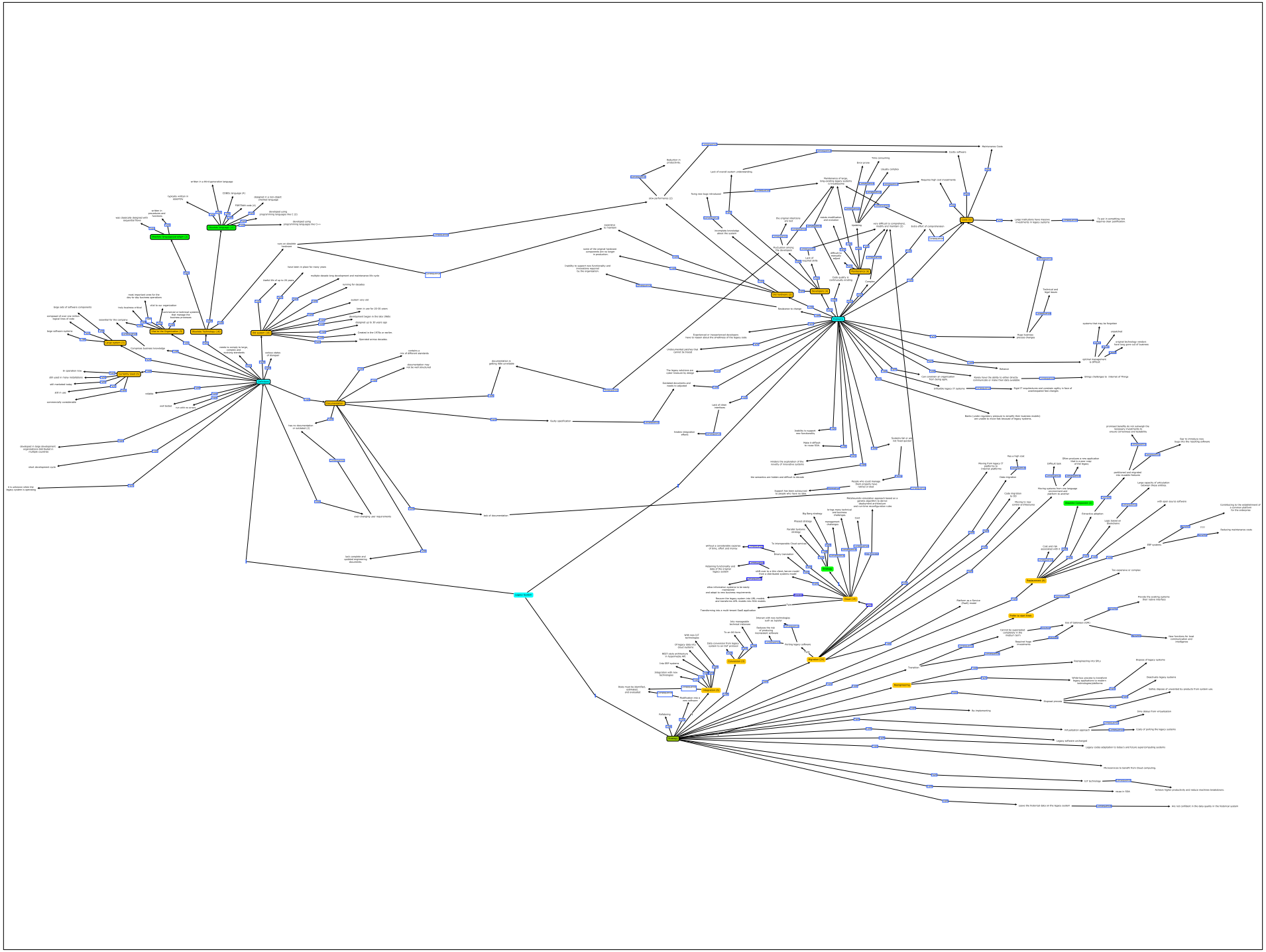
Para apoiar a resolução desta problemática, o modelo visual exposto na imagem abaixo é fruto do processo de codificação da GT, contendo seus agrupamentos e relacionamentos. O mesmo tem como objetivo apoiar na elucidação de elementos que envolvem um sistema legado e as relações que existem entre esses elementos, assim auxiliando no entendimento sobre quando um sistema é ou pode estar se tornando legado, assim como possíveis estratégias que podem ser tomadas para evoluir tais sistemas.

O modelo visual aqui apresentado além de conter os agrupamentos e relações entre grupos, subgrupos e códigos, contém o que foi possível identificar de relações entre os grandes agrupamentos desta pesquisa (definições, problemas e estratégias).

Assim, obteve-se um entendimento mais consiso sobre que tipos de relacionamentos existem entre uma característica e um problema ou estratégia. Foi possível identificar que muitos problemas, ocasionados por estes sistemas, são consequências de características que muitos autores denotam como definidoras de sistemas legados, por exemplo, através do modelo visual foi possível identificar que sistemas legados possuem perfomance lenta (problema) e são caros de manter (problema) pelo fato de funcionarem com hardware obsoleto (definição). Assim como os legados possuírem uma documentação desatualizada que necessita de ajustes (problema) por serem consequência de falhas na especificação dos mesmos (definição).

O inverso também foi possível identificar, ou seja, a documentação de um legado esta ficando pouco confiável (definição) por ser consequência de uma flutuação de desenvolvedores ao longo do tempo (problema).

Estes apenas são exemplos do que foi possível identificar com o desenvolvimento e entendimento deste modelo visual geral. O modelo geral pode ser melhor visualizado na imagem abaixo.



Este possui uma estrutura didática e clara para que possam ser lidas e identificadas as proposições que são realizadas de acordo com os resultados da pesquisa e codificação, este modelo é apresentado nesta fase final do processo, para que assim, possa ser validado e estudado posteriormente.

5.5 Teoria Geral

Com base na Teoria Fundamentada em Dados foi possível identificar quais os elementos que definem um sistema legado, quais principais problemas que sistemas legados causam e quais as estratégias de evolução existentes.

Um sistema pode ser considerado legado através da tecnologia obsoleta utilizada para o desenvolvimento do mesmo. O uso de linguagens de programação como (*e.g* Cobol, Fortran, C) detalham o que é considerado tecnologia obsoleta. O fato de o sistema ter sido desenvolvido ou estar sendo utilizado há um tempo determinado (mais de 20 anos) também é um fator de definição de sistema legado. A importância que um legado tem para a organização também se mostra um elemento determinante em definir esse tipo de sistema, visto que o mesmo é essencial à empresa por ainda conter as operações de negócios da organização, operações essas que são consideradas críticas e vitais nas empresas onde esses sistemas operam.

Os sistemas legados possuem alguns problemas, entre eles os altos custos financeiros que os mesmos proporcionam, devido ao fato que manter esse tipo de sistema ativo e operacional demanda investimentos significativos à organização. Muitos desses problemas são ocasionados pela dificuldade existente em manutenir esses sistemas, por serem resistentes à modificações, ocasionarem erros, serem complexos e consumirem um tempo excessivo para realizar manutenções. Por existir uma alta flutuação de recursos humanos as intenções originais que o sistema possuia acaba se perdendo, bem como tornam as manutenções problemáticas.

Um sistema legado que opera com hardware obsoleto (definição) tem como consequência uma perfomance lenta desse sistema (problema) e posteriormente uma manutenção cara (problema). Legados com falta de documentação (definição) causam problemas como a dificuldade em compreender, modificar e manter um sistema legado. A documentação com falha tem como consequência documentos desatualizados que necessitam ser ajustados.

Sistemas legados possuem algumas estratégias de evolução. Atualmente as organizações preferem correr riscos atualizando, reengenhando, substuindo componentes, migrando para plataformas novas do que desenvolvendo um sistema novo.

Dentre essas alternativas de evolução está a migração para a nuvem, que possui alguns beneficios, como a possibilidade de que sistemas legados sejam facilmente mantidos e adaptados aos novos requisitos de negócios assim como a manutenibilidade das funcionalidades originais do sistema. A substituição é uma outra altenativa de estratégia. Alguns

exemplos são substituições de componentes obsoletos, substituir para lógicas baseadas em *blockchains*, substituir para *ERP systems* que traz como beneficios a redução dos custos de manutenção. A integração de legados também é uma estratégia demonstrada, integração para novas tecnologias *IoT*, integração de dados para sistemas em nuvem, integração para sistemas *ERP* exemplificam este subgrupo de estratégias.

Assim, é proposta uma definição sobre sistema legado, sendo ela: Um sistema legado é aquele desenvolvido com tecnologia obsoleta, como a linguagem de programação (*e.g* Cobol, Fortran ou C), cujo custo financeiro é um dos maiores problemas às organizações, e a migração é a estratégia mais realizada atualmente, principalmente para plataformas em nuvem.

6 CONSIDERAÇÕES FINAIS

Esse trabalho teve como objetivo oferecer à literatura um entendimento mais consiso sobre os elementos que envolvem sistemas legados e as possíveis alternativas de solução. Para isso utilizou-se procedimentos da GT versão Straussiana a fim de analisar trechos textuais extraídos da literatura. Assim, criou-se um modelo visual e textual que auxilia nesse entendimento de forma compartilhada e única.

Os desafios encontrados durante a condução desta pesquisa relacionam-se ao uso da metodologia qualitativa de análise de dados GT. Há uma carência de trabalhos que mostrem a aplicação do desenvolvimento desta metodologia. Sendo assim, esse estudo foi desafiador pois foi necessário realizar os procedimentos da GT sem nenhum exemplo completo de utilização dessa metodologia.

Outro desafio foi em relação ao tempo hábil para realizar todo o processo de extração dos trechos textuais e, posteriormente, o processo de codificação. Em relação às extrações de trechos, o processo foi realizado em 2 momentos, com um dos especialistas realizando as extrações manualmente e o outro especialista analisando os trechos para verificar se estavam adequados. Em relação à codificação, os participantes criavam os códigos e os mesmos eram posteriormente analisados por 2 especialistas. A fase de agrupamento também foi realizada em 2 momentos, com um dos especialistas identificando os agrupamentos e o outro analisando os resultados. Esse processo de dois momentos também foi realizado na fase de identificação de relacionamento entre os agrupamentos. Todo esse processo justifica a escolha de realizar o mapeamento contendo estudos publicados somente no ano de 2018.

Ao final do trabalho, os resultados obtidos mostram que os objetivos propostos foram alcançados. Foi gerado um entendimento único, consensual e compartilhado sobre elementos que envolvem um sistema legado. Esse entendimento foi explicitado com a construção de modelos visuais e uma teoria que elucidam os elementos (definições, problemas e estratégias) que envolvem um sistema legado, bem como as relações existentes entre esses elementos. As relações existentes demonstram quais elementos causam determinados problemas.

Como contribuicões diretas desse trabalho destaca-se que o reconhecimento de definições permite a identificação de sistemas legados, algo que é uma dificuldade atualmente visto a falta de consenso sobre as características que definem esse tipo de sistema. Por sua vez, a identificação de problemas proporciona um entendimento sobre o que o sistema está causando ou poderá causar, bem como o que é possível mitigar a tempo. As possíveis estratégias proporcionam o reconhecimento de alternativas para a evolução do sistema, mantendo o mesmo ativo.

Também foi possível visualizar que os resultados obtidos no grupo de estratégias acabaram revelando uma visão diferente de autores clássicos, como Warren (1999) e Sommerville (2011). O último alega existir apenas 4 possíveis estratégias de evolução, como

explicitado na seção 2.2, e as mesmas foram pouco relatadas nos estudos publicados atualmente. A migração de legados, esta que sequer é relatada pelos autores citados, foi a estratégia mais frequente. Esse resultado pode ser interpretada pelo motivo temporal e evolucionário das tecnologias, ou seja, em 2011 (ano da citação de Sommerville) a migração não era uma estratégia utilizada. Com a chegada de plataformas em nuvem (tipo de migração mais relatada) e os benefícios originados pela mesma, muitas organizações acabaram escolhendo esta nova opção de estratégia.

6.1 Trabalhos Futuros

Em relação aos trabalhos futuros, é possível realizar uma extração de dados mais ampla, contemplando assim uma número maior de anos, o que tornaria o entendimento acerca de sistemas legados mais robusto e confiável.

Adicionalmente, para viabilizar uma extração de trechos textuais mais ampla, propõe-se o desenvolvimento de uma aplicação que possibilite essa extração de forma automática ou semi-automática. Destaca-se que esses resultados são a entrada para o processo de codificação dentro da ferramenta *Open Code Tool*.

Ainda em relação à ferramenta *Open Code Tool*, entende-se que a evolução da mesma é uma oportunidade de trabalho futuro, visto que existem poucas ferramentas gratuitas que realizam o processo de codificação e nenhuma delas é colaborativa.

Por fim, é possível vislumbrar o desenvolvimento promissor de uma aplicação que identifique se um sistema é ou está se tornando legado, através dos elementos (definições e problemas) encontrados com esta pesquisa. A aplicação pode ser concebida como um painel, que deve ser alimentado com dados reais de acordo com os elementos definidores e causadores de problemas, e deve emitir um aviso ou alerta quando um determinado número ou percentual de relacionado a proximadade do sistema ser ou se tornar legados for atingido. A aplicação também pode indicar o que pode ser feito para evoluir esse sistema.

REFERÊNCIAS

ABAD, Z.; RUHE, G.; NOAEEN, M. Requirements engineering visualization: A systematic literature review. **2016 IEEE 24th International Requirements Engineering Conference**, 2016. Citado na página 43.

ABDULAZIZ, S. G.; EL-ABBASSY, A.; HEGAZY, A. A. Developing a software architecture comparison analysis method for critical systems at international airports. **Journal of ACS: Advances in Computer Science (2010)**, 2010. Citado na página 20.

ALATAWI, E.; MENDOZA, A.; MILLER, T. Psychologically-driven requirements engineering: A case study in depression care. **2018 25th Australasian Software Engineering Conference (ASWEC)**, 2018. Citado na página 42.

BARBOSA, M. Uma anÁlise do uso de grounded theory em engenharia de software. **Revista Produção Online, Florianópolis, SC,**, v. 17, p. 26–48, 2017. Citado 4 vezes nas páginas 39, 45, 46 e 47.

BARKE, H.; PRECHELT, L. Some reasons why actual cross-fertilization in cross-functional agile teams is difficult. **2018 IEEE/ACM 11th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)**, 2018. Citado na página 44.

BELLER, M. et al. Analyzing the state of static analysis: A large-scale evaluation in open source software. **2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)**, 2016. Citado na página 43.

BENNETT. Legacy systems: coping with success. **IEEE Software**, v. 12, n. 1, p. 19–23, 1995. Citado 2 vezes nas páginas 20 e 26.

BENNETT, K. H.; WARD, M. P. Formal methods for legacy systems. **Journal of Software: Evolution and Process**, v. 07, p. 203–219, 1995. Citado na página 26.

BICK, S. et al. Coordination challenges in large-scale software development: A case study of planning misalignment in hybrid settings. **IEEE Transactions on Software Engineering**, v. 44, p. 932–950, 2018. Citado na página 43.

BISBAL, J. et al. Legacy information systems: issues and directions. **IEEE Software**, v. 16, 1999. Citado na página 26.

BLOOMFIELD, R. J. **What Counts and What Gets Counted**. [S.l.]: SSRN, 2011. Citado na página 25.

BOKKA, R. System and method for receiving services provided by distributed systems. **Charles Schwab Co., Inc. (San Francisco, CA, US)**, 2015. Citado na página 20.

CERECI, I.; KARAKAYA, Z. Need for a software development methodology for research-based software projects. **2018 3rd International Conference on Computer Science and Engineering (UBMK)**, 2018. Citado na página 42.

CHARMAZ, K. **A CONSTRUÇÃO DA TEORIA FUNDAMENTADA - Guia Prático para Análise Qualitativa**. [S.l.]: Porto Alegre: Artmed, 2009. Citado 4 vezes nas páginas 31, 33, 34 e 56.

CORBIN, J.; STRAUSS, A. **Basics of Qualitative Research - Techniques and procedures for Developing Grounded Theory 4. ed.** [S.l.]: SAGE, 2014. Citado na página 36.

COYNE, I.; COWLEY, S. Using grounded theory to research parent participation. **Journal of Research in Nursing 11, no. 6 (November 2006)**, 2006. Citado na página 34.

CRESWELL, J. W. **Investigação Qualitativa e Projeto de Pesquisa - Escolhendo entre Cinco Abordagens**. [S.l.]: Porto Alegre: Penso, 2014. Citado 3 vezes nas páginas 30, 34 e 35.

CUNHA, J. et al. Decision-making in software project management: A qualitative case study of a private organization. **2016 IEEE/ACM Cooperative and Human Aspects of Software Engineering (CHASE)**, 2016. Citado na página 42.

CUTCLIFFE, J. R. Methodological issues in grounded theory. **Journal of Advanced Nursing**, v. 6, p. 1476–1488, 2000. Citado na página 36.

DALKEY, N.; HELMER, O. An experimental application of the delphi method to the use of experts. **Management Science**, v. 9, n. 3, p. 458–467, 1963. Citado na página 55.

FONSECA, L.; SOARES, S.; SEAMAN, C. Describing what experimental software engineering experts do when they design their experiments – a qualitative study. **ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)**, 2017. Citado na página 41.

GAROUSI, V.; HERKILOGLU, K. Selecting the right topics for industry-academia collaborations in software testing: An experience report. **2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)**, 2016. Citado na página 42.

GIARDINO, C. et al. Software development in startup companies: The greenfield startup model. **IEEE Transactions on Software Engineering**, v. 42, p. 585–604, 2016. Citado na página 41.

GIBBS, G. **Análise de Dados Qualitativos**. [S.l.]: Porto Alegre: Artmed, 2009. Citado 3 vezes nas páginas 35, 53 e 56.

GLASER, B.; STRAUSS, A. **The Discovery of Grounded Theory: Strategies for Qualitative Research**. [S.l.]: New York: Aldine, 1967. Citado 3 vezes nas páginas 30, 31 e 34.

GRALHA, C.; DAMIAN, D.; WASSERMAN, A. The evolution of requirements practices in software startups. **2018 ACM/IEEE 40th International Conference on Software Engineering**, 2018. Citado na página 42.

GUO, H. et al. Wrapping client-server application to web services for internet computing. **Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05)**, 2005. Citado na página 20.

HANDIGUND, S.; KULKARNI, R. An ameliorated methodology for the design of object structures from legacy 'c' program. **International Journal of Computer Applications**, 2010. Citado na página 20.

HODA, R. et al. Socio-cultural challenges in global software engineering education. **IEEE Transactions on Education**, v. 60, p. 173–182, 2017. Citado na página 41.

HODA, R.; NOBLE, J. Becoming agile: A grounded theory of agile transitions in practice. **2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)**, 2017. Citado na página 41.

HOLMES, R.; ALLEN, M.; CRAIG, M. Dimensions of experientialism for software engineering education. **2018 IEEE/ACM 40th International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)**, 2018. Citado na página 41.

JATAIN, A.; GAUR, D. Reengineering techniques for object oriented legacy systems. **International Journal of Software Engineering and Its Applications**, 2015. Citado na página 20.

JURIC, M.; ROZMAN, I.; HERICKO, M. Performance comparison of corba and rmi. **Information and Software Technology**, v. 42, p. 915–933, 2000. Citado na página 20.

JúNIOR, N. Toward a theory of communication in distributed software development teams: A research proposal. **2018 IEEE/ACM 13th International Conference on Global Software Engineering (ICGSE)**, 2018. Citado na página 43.

KHADKA, R. et al. How do professionals perceive legacy systems and software modernization? **ICSE 2014 Proceedings of the 36th International Conference on Software Engineering**, p. 36–47, 2014. Citado na página 26.

KHANYE, T.; OPHOFF, J.; JOHNSTON, K. Issues in migrating legacy systems to the cloud. **8th International Conference on Cloud Computing, Data Science Engineering (Confluence)**, 2018. Citado na página 60.

KITCHENHAM, B. Procedures for performing systematic reviews. **Joint Technical Report, TR/SE-0401 and NICTA 0400011T.1**, 2004. Citado 2 vezes nas páginas 39 e 50.

KNOL, A.; TAN, Y. H. The cultivation of information infrastructures for international trade: stakeholder challenges and engagement reasons. **Journal of Theoretical and Applied Electronic Commerce Research archive**, v. 13, p. 106–117, 2018. Citado na página 60.

KUGELE, S. et al. On service-orientation for automotive software. **2017 IEEE International Conference on Software Architecture (ICSA)**, 2017. Citado na página 44.

KURTANOVIC, Z.; MAALEJ, W. Mining user rationale from software reviews. **2017 IEEE 25th International Requirements Engineering Conference (RE)**, 2017. Citado na página 43.

MALKI, M.; BENSLIMANE, S. M.; ABDELKADER, M. A heuristic approach to locate candidate web service in legacy software. **International Journal of Computer Applications in Technology**, v. 47, 2013. Citado na página 26.

MARTINS, D.; CHERVENSKI, A.; BORDIN, A. S. Identificação de características de sistemas legados a partir da análise de conteúdo da literatura. 1 Escola Regional Engenharia de Software (ERES), 2017. Citado 2 vezes nas páginas 19 e 21.

MATSUBARA, P.; SILVA, C. D. Game elements in a software engineering study group: A case study. **2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)**, p. 160–169, 2017. Citado na página 41.

MERTZ, J.; NUNES, I. A qualitative study of application-level caching. **IEEE Transactions on Software Engineering**, v. 43, p. 798–816, 2017. Citado na página 44.

MUNARETTO, L. F.; CORRêA, H. L.; CUNHA, J. A. C. D. Um estudo sobre as caracterÍsticas do mÉtodo delphi e de grupo focal, como tÉcnicas na obtenÇÃo de dados em pesquisas exploratÓrias. **Rev. Adm. UFSM, Santa Maria, v. 6, n. 1, p. 09-24, JAN./MAR. 2013**, 2013. Citado na página 55.

NICOLáS, J. et al. On the risks and safeguards for requirements engineering in global software development: Systematic literature review and quantitative assessment. **IEEE Access**, v. 6, p. 59628–59656, 2018. Citado na página 44.

NILSSON, S. Application modernization: Approaches, problems and evaluation. Dissertation, 2015. Citado na página 26.

O'BYRNE, P.; WU, B. Lace frameworks and technique - identifying the legacy status of an information system from the perspectives of its causes and effects. **In Proceedings International Symposium on Principles of Software Evolution**, 2000. Citado na página 21.

OLIVEIRA, J. A.; VARGAS, M.; RODRIGUES, R. Soa reuse: Systematic literature review updating and research directions. **SBSI'18 Proceedings of the XIV Brazilian Symposium on Information Systems Article No. 71**, 2018. Citado na página 60.

PETERSEN, K. et al. Systematic mapping studies in software engineering. **EASE'08 Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering**, p. 68–77, 2008. Citado na página 50.

PINTO, H. L. M.; BRAGA, J. L. Sistemas legados e as novas tecnologias: técnicas de integração e estudo de caso. **Informática Pública, Belo Horizonte**, p. 48–69, 2004. Citado na página 27.

PRECHELT, L.; SCHMEISKY, H.; ZIERIS, F. Quality experience: A grounded theory of successful agile projects without dedicated testers. **2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)**, 2016. Citado na página 42.

PRESSMAN, R.; MAXIM, B. **Engenharia de Software - Uma Abordagem Profissional**. [S.l.]: Bookman, 2006. Citado na página 27.

RAHMAN, A.; STALLINGS, J.; WILLIAMS, L. Poster: Defect prediction metrics for infrastructure as code scripts in devops. **2018 ACM/IEEE 40th International Conference on Software Engineering: Companion Proceedings**, 2018. Citado na página 43.

RAHMAN, A.; WILLIAMS, L. Characterizing defective configuration scripts used for continuous deployment. **2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)**, 2018. Citado na página 44.

RAJAVAT, A.; TOKEKAR, V. Investigation of quality and functional risk issues in reengineering process of legacy software system. **International Journal of Programming Languages and Applications (IJPLA)**, v. 4, 2014. Citado na página 21.

RANSOM, J.; SOMMERVILLE, I.; WARREN, I. **A Method for Assessing Legacy Systems for Evolution**. [S.l.]: IEEE, 1998. Citado na página 28.

RODEGHERO, P. Behavior-informed algorithms for automatic documentation generation. **2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)**, 2017. Citado na página 43.

RODRIGUEZ, P.; EMILIA, M.; TURHAN, B. Key stakeholders' value propositions for feature selection in software-intensive products: An industrial case study. **IEEE Transactions on Software Engineering**, 2018. Citado na página 42.

SANDBORN, P.; PRABHAKAR, V. The forecasting and impact of the loss of critical human skills necessary for supporting legacy systems. **IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT**, 2015. Citado na página 20.

SANTOS, J. et al. Análise de dados: comparação entre as diferentes perspectivas metodológicas da teoria fundamentada nos dados. **Revista da Escola de Enfermagem da USP**, 2018. Citado na página 49.

SANTOS, W. et al. Towards a theory of simplicity in agile software development: A qualitative study. **2017 43rd Euromicro Conference on Software Engineering and Advanced Applications**, 2017. Citado na página 43.

SCHRAMM, M.; DANEVA, M. Implementations of service oriented architecture and agile software development: What works and what are the challenges? **2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS)**, 2016. Citado na página 44.

SEACORD, R. C.; PLAKOSH, D.; LEWIS, G. A. **Modernizing Legacy Systems: Software Technologies, Engineering Process and Business Practices**. [S.l.]: Boston : Addison-Wesley, 2003. - 332 p, 2003. Citado na página 27.

SEDANO, T.; PAUL, R.; PéRAIRE, C. Lessons learned from an extended participant observation grounded theory study. **2017 IEEE/ACM 5th International Workshop on Conducting Empirical Studies in Industry (CESI)**, 2017. Citado na página 41.

SOMMERVILLE, I. **ENGENHARIA DE SOFTWARE 8. ed.** [S.l.]: São Paulo: Pearson Addison-Wesley, 2007. Citado na página 27.

SOMMERVILLE, I. **ENGENHARIA DE SOFTWARE 9. ed.** [S.l.]: Pearson Prentice Hall, 2011. Citado 3 vezes nas páginas 28, 29 e 83.

SONG, H. et al. How to support customisation on saas: A grounded theory from customisation consultants. **2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)**, 2017. Citado na página 41.

SOUSA, L. et al. Identifying design problems in the source code: A grounded theory. **2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)**, 2018. Citado na página 41.

STOL, K.-J.; RALPH, P.; FITZGERALD, B. Grounded theory in software engineering research: A critical review and guidelines. **IEEE International Conference on Software Engineering**, 2016. Citado 2 vezes nas páginas 31 e 32.

STRAUSS, A.; CORBIN, J. **Basics of Qualitative Research : Techniques and Procedures for Developing Grounded Theory 2 ed.** [S.l.]: Londres: Sage Publications, 1998. Citado 11 vezes nas páginas 21, 31, 33, 34, 36, 49, 50, 53, 54, 56 e 57.

STRAUSS, A.; CORBIN, J. **Pesquisa qualitativa: técnicas e procedimentos para o desenvolvimento de teoria fundamentada**. [S.l.]: Porto Alegre: Artmed, 2008. Citado 6 vezes nas páginas 21, 30, 33, 35, 36 e 49.

TWEED, A.; CHARMAZ, K. Grounded theory methods for mental health practitioners. **Qualitative Research Methods in Mental Health and Psychotherapy: A Guide for Students and Practitioners, First Edition.**, 2011. Citado na página 31.

VALENTIM, N.; SILVA, W.; CONTE, T. The students' perspectives on applying design thinking for the design of mobile applications. **2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET)**, 2017. Citado na página 42.

VISAGGIO, G. Ageing of a data-intensive legacy system: symptoms and remedies. **Journal of Software: Evolution and Process**, v. 13, p. 281–308, 2001. Citado na página 20.

WARREN, I. **The Renaissance of Legacy Systems**. [S.l.]: Springer, London, 1999. Citado 4 vezes nas páginas 27, 29, 30 e 83.

ZAKARIA, N.; IBRAHIM, S.; MAHRIN, M. Using grounded theory approach to identify value-based factors in software development. **2016 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M)**, 2016. Citado na página 41.

ZANATTA, A.; MACHADO, L.; STEINMACHER, I. Competence, collaboration, and time management: Barriers and recommendations for crowdworkers. **2018 IEEE/ACM 5th International Workshop on Crowd Sourcing in Software Engineering (CSI-SE)**, 2018. Citado na página 44.

ZHANG, Z. et al. Securing fpga-based obsolete component replacement for legacy systems. **International Symposium on Quality Electronic Design (ISQED)**, 2018. Citado na página 60.

ZHANG, Z.; YU, Q. Exploiting principle of moving target defense to secure fpga systems. **IEEE Computer Society Annual Symposium on VLSI (ISVLSI)**, 2018. Citado na página 60.

ZIERIS, F.; PRECHELT, L. Observations on knowledge transfer of professional software developers during pair programming. **2016 IEEE/ACM 38th IEEE International Conference on Software Engineering Companion**, 2016. Citado na página 42.

ZOU, Y. Quality driven software migration of procedural code to object-oriented design. **Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05)**, 2005. Citado na página 20.

ÍNDICE

CE, 13, 40, 52
CI, 13, 40, 47, 52

ES, 39–41, 45, 47

GT, 11, 13, 17, 21–23, 30–36, 39–47, 49,
54, 72, 73, 79, 83

MSL, 50, 53, 72

QP, 31, 39, 45, 47, 51

RSL, 39–41, 43–47