

Understanding Legacy Systems in the Light of Grounded Theory

Alex Severo Chervenski
UNIPAMPA - Universidade Federal do Pampa
Alegrete, Brazil
alex.chervenski@gmail.com

ABSTRACT

Software systems developed over a long period of time require constant maintenance to remain useful to organizations. These systems, which tend to degrade and cause maintenance problems, are usually called legacy systems. The management of legacy systems involves understanding its characteristics, such as what it is, what problems it presents and which evolution strategies. From the literature perspective, there is a wide and diversified understanding of these characteristics. This article is intended to provide a more concise understanding of such features related to legacy systems. To achieve this goal, textual excerpts from the literature were collected and analyzed according to Grounded Theory (GT) procedures, resulting in a concise understanding supported by visual models. The results show that systems developed with obsolete technologies, some decades ago, that have incomplete or missing documentation, among others, are defining characteristics for a system to be considered legacy. Some characteristics cause financial problems, difficulty in maintaining the code, among others. It was also observed that there are several evolution strategies and that migration of legacies to the platform is a trend. It is hoped that this study will serve as support for the identification of elements that indicate whether a system is or is becoming a legacy, as well as the possible solutions to evolve them. In addition, the detailed data analysis process with GT procedures, which used a collaborative coding process with a tool developed for this purpose, can support researchers who need to use this type of method in their research in various areas of knowledge.

KEYWORDS

Legacy Systems, Grounded Theory, Software Evolution, Software Maintenance.

ACM Reference Format:

Alex Severo Chervenski and Andréa Sabedra Bordin. 2020. Understanding Legacy Systems in the Light of Grounded Theory. explicitly sets . In *34th Brazilian Symposium on Software Engineering (SBES '20), October 21–23, 2020, Natal, Brazil*.<https://doi.org/10.1145/3422392.3422400>

1 INTRODUÇÃO

Sistemas de software vêm sendo desenvolvidos ao longo de décadas e muitos deles ficam em operação nas organizações por muito

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SBES '20, October 21–23, 2020, Natal, Brazil

© 2020 Association for Computing Machinery.

ACM ISBN 978-1-4503-8753-8/20/09...\$15.00

<https://doi.org/10.1145/3422392.3422400>

Andréa Sabedra Bordin
UNIPAMPA - Universidade Federal do Pampa
Alegrete, Brazil
andreibordin@unipampa.edu.br

tempo, sendo reconhecidos como legados à organização. Para Perez-Castillo et al. [9], ou esses sistemas passam por vários ciclos de manutenção para continuarem sendo úteis, ou os processos organizacionais são adaptados para se trabalhar eficientemente com esses sistemas.

O excesso de manutenções em um sistema, tende a torná-lo cada vez mais degradado, envelhecido e consequentemente difícil de manter [17]. Isso ocorre por motivos diversos, como o número de mantenedores que alteram o código, cada qual com conhecimentos técnicos em níveis diferentes, a falta de documentação do sistema, dentre outros. A dificuldade de manutenção traz problemas de ordem operacional e econômica para as organizações, visto que este tipo de atividade tende a demorar mais tempo para ser realizada.

A dependência de sistemas legados provavelmente será uma realidade para sempre, uma vez que as tecnologias avançam rapidamente e a substituição desses sistemas, como uma alternativa, gera um alto custo financeiro. Assim, muitos dos novos sistemas de hoje se tornarão sistemas legados amanhã [12].

Logo, gerenciar sistemas legados é um grande desafio para as organizações. De acordo com OByrne e Wu [7] um dos desafios mais difíceis em lidar com legados é como identificar e determinar o “status legado” de um sistema, onde não existe uma definição comumente aceita. Identificar se um sistema é ou está se tornando legado, a partir das características que o definam, dos problemas que apresenta, assim como conhecer as estratégias de evolução desse tipo de sistema são importantes, de forma que as melhores decisões de evolução sejam tomadas e os efeitos negativos sejam mitigados precocemente.

A literatura que aborda sistemas legados vem se consolidando ao longos dos últimos 25 anos, apresentando definições, problemas e estratégias de evolução para esses sistemas. Contudo, observou-se uma profusão de abordagens acerca desses elementos, resultando em um corpo de conhecimentos não conciso.

Em relação às definições de sistemas legados, autores abordam aspectos diversos, tais como: “são grandes sistemas de software que não sabemos como lidar, mas que são vitais para a organização” [1], que enfatiza aspectos como o tamanho do sistema e da importância para a organização; “software legado é aquele desenvolvido com tecnologia ultrapassada, como C, C++, Cobol” [5], que destaca o aspecto tecnológico. Em relação aos problemas que costumam causar, são apontados os seguintes: “são sistemas muito caros para serem mantidos” [10]; “são sistemas difíceis de entender e manter, seja pela falta de documentação ou pela falta de profissionais especializados” [10], dentre outros. As estratégias de evolução também são diversas, desde as apresentadas inicialmente por Ransom et al. [11], como descartar o sistema completamente, reengenhar, substituir todo ou parte do sistema por um novo sistema e continuar com a manutenção regular, até algumas abordagens mais atuais como a

migração de sistemas legados para plataformas em nuvem sugerida por Zhao et al [18].

Não foram encontrados estudos onde esse conhecimento tenha sido organizado, correlacionado e sintetizado de uma forma sistemática. No estudo de Martins et al. [6] os autores se detiveram a encontrar e agrupar características de sistemas legados a partir de definições encontradas na literatura de uma forma geral, sem fazer distinções entre essas características e sem estabelecer relacionamentos entre as mesmas.

Um estudo mais completo pode permitir o entendimento único e compartilhado sobre os elementos que compõem um legado, apontando fatores causadores ou definidores 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 sua evolução.

Assim, o objetivo deste trabalho é oferecer um entendimento conciso sobre os elementos que envolvem sistemas legados. 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 conduza a um resultado, sustentado pelos dados coletados, que possa ser representado de uma forma textual e visual que proporcione o entendimento do tema. A Teoria Fundamentada em Dados, do inglês *Grounded Theory* (GT) é um método de pesquisa qualitativa que contempla essas necessidades [16].

A principal contribuição deste trabalho reside na apresentação de um entendimento conciso sobre o que é um sistema legado, quais os problemas que apresenta e quais as estratégias de evolução possíveis, na forma de uma teoria e de representações visuais. Entende-se também que outra contribuição é o detalhamento da utilização dos procedimentos da GT, visto que a maioria dos estudos não a aborda com detalhes, especialmente em problemas da Engenharia de Software. Por fim, o processo de codificação aberta colaborativa e a ferramenta desenvolvida para este fim, também são entendidas como contribuições.

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta alguns trabalhos relacionados, a Seção 3 apresenta a metodologia utilizada neste estudo, etapas e atividades necessárias para a obtenção dos resultados pretendidos, a Seção 4 explica os resultados obtidos neste estudo, a Seção 5 descreve as ameaças a validade deste estudo e, finalmente, a Seção 6 apresenta as considerações finais e os trabalhos futuros.

2 TRABALHOS RELACIONADOS

O trabalho que mais se aproxima da presente pesquisa, na motivação e escolha do método, é o de Martins et al [6]. Nesse trabalho os autores argumentam que não existe um consenso nas definições e entendimento acerca de sistemas legados e que mapear suas características é importante para identificar se um sistema é ou está se tornando legado. Foi realizada uma busca sistemática na literatura a fim de coletar definições de sistemas legados, na qual obteve-se um total de 125 definições, capturadas entre os anos de 1995 à 2015 (com um intervalo de 5). Os autores escolheram um método de pesquisa qualitativo, a análise de conteúdo, para identificar códigos (chamados de características) nas definições e agrupá-los em categorias (chamados de aspectos). Foram extraídas 62 características

distintas e as respectivas frequências, as quais foram agrupadas nos aspectos tempo, organizacional, econômico técnico, dentre outros.

Contudo, o trabalho apresenta os elementos de forma genérica, como características (traço, propriedade ou qualidade distintiva fundamental), não explorando outras facetas ou categorias que são, de fato, problemas ou estratégias de evolução, como por exemplo, difíceis e caros de manter, inexiste ncia de pessoal especializado para a manutenção, evoluem com novas tecnologias emergentes, dentre outros. Além disso, muitos códigos possuem relacionamento diversos, por exemplo, de causa e consequência, como no caso da documentação ausente ou desatualizada (causa) que resulta em dificuldade de entendimento pelos desenvolvedores (consequência).

Dessa forma, entende-se que a presente pesquisa compartilha da motivação deste trabalho relacionado, mas vai além, preenchendo as lacunas destacadas ao analisar outros elementos importantes para o entendimento de sistemas legados. Além disso, também utiliza a literatura como fonte de dados e um método qualitativo de análise, mas neste caso, um método que fornece procedimentos de análise mais adequados a geração de um entendimento mais consistente.

3 METODOLOGIA

Para a 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 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. 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 [16].

Ao longo dos anos, a GT acabou tendo algumas versões diferentes, sendo adaptada de acordo com os estudos em que eram utilizada. De acordo com [14] é reconhecido que existem pelo menos três fluxos, são eles: GT versão Clássica ou Glaseriana [4], a GT versão Straussiana [15] e a GT versão Construtivista [2].

No estudo de [13], é afirmado que a linha Straussiana é melhor indicada para pesquisadores iniciantes no método da GT, pois apresenta um sistema de análise de dados mais sistemático em relação as demais vertentes da GT. Isso acabou sendo determinante para a escolha da utilização da GT versão Straussiana neste trabalho.

A GT Straussiana é constituída por 3 fases distintas, são elas:

- Codificação Aberta: consiste em geração de códigos e categorias e como são variadas dimensionalmente podem ser analisadas linha por linha, frase, parágrafo ou documentos completos.
- Codificação Axial: tem como objetivo especificar as propriedades e as dimensões de uma categoria e consiste em um processo de reagrupamento dos dados. Também é possível identificar relações entre os agrupamentos encontrados.
- Codificação Seletiva: serve para identificar uma categoria central, sendo a fase mais abstrata do processo, que subsidiará a formação de um esquema organizacional maior e os resultados que compõem a teoria. Ainda nesta fase, conceitos e agrupamentos são revistos a fim de alterar algo caso seja necessário.

A GT foi utilizada neste estudo por fornecer uma forma de análise de dados bem definida e criteriosa, que proporcionou um maior discernimento dos elementos que envolvem um sistema legado. A Figura 1 demonstra o processo geral dos procedimentos metodológicos necessários para cumprir os objetivos deste trabalho.

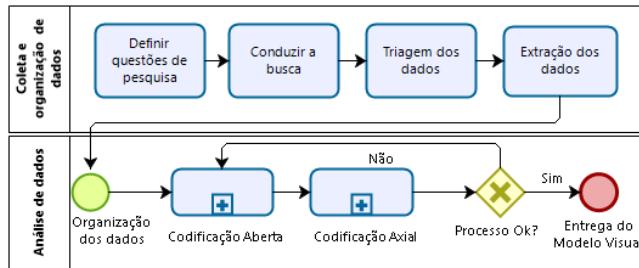


Figure 1: Processo dos procedimentos metodológicos

3.1 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 metodológico. Essa fase foi realizada conforme algumas atividades definidas para um protocolo de Mapeamento Sistemático na Literatura (MSL) como o proposto por [8].

3.1.1 Definir questões de pesquisa. Com o objetivo de encontrar dados relacionados à definições, problemas ou estratégias de evolução de sistemas legados, as seguintes questões de pesquisa foram elaboradas:

- Questão de pesquisa 1: O que é um sistema legado?
- Questão de pesquisa 2: Quais os problemas de um sistema legado?
- Questão de pesquisa 3: Quais estratégias existem para a evolução de um sistema legado?

3.1.2 Conduzir a busca. A busca foi conduzida em 2 passos, sendo eles: definir as bases de dados e construir a *string* de busca para encontrar estudos relevantes à pesquisa.

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 permitem busca em texto completo, além de serem bases conhecidas e utilizadas no âmbito acadêmico.

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*. As strings podem ser visualizadas na Tabela 1.

Table 1: Strings de Busca

Base de Dados	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"))))
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"))))
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"))))

3.1.3 Triagem dos dados. De acordo com as atividades do processo de mapeamento, os critérios de inclusão (CI) e critérios de exclusão (CE) serviram para realizar uma triagem dos estudos que foram retornados pelas bases de dados escolhidas, removendo estudos que não eram relevantes para a pesquisa. Os critérios adotados por este trabalho são os listados na Tabela 2.

Table 2: Critérios de Inclusão (CI) e Exclusão (CE)

Tipo de critério	Critério
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.
Exclusão	CE1. O estudo não é fornecido por completo pela base de dados.
	CE2. O estudo é duplicado.
	CE3. O estudo não atende algum dos CI.

A Figura 2 mostra o número de estudos retornados após a execução da *string* de busca e a aplicação dos CI e CE. Um total de 7.562

artigos foram retornados na primeira pesquisa nas bases digitais. Após as aplicações do CI1, obteve-se como resultado um número de 375 artigos e após a aplicação do CI2, foi obtido o resultado final de 87 estudos.

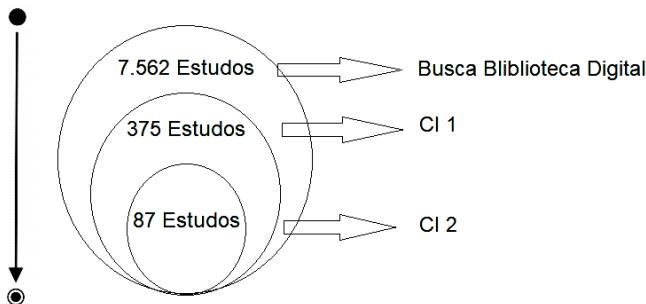


Figure 2: Ciclos de Extração de dados

3.1.4 Extração dos dados. O processo de extração de dados foi realizado por dois especialistas em 2 momentos, com um dos especialistas realizando as extrações de trechos textuais relacionados à definições, problemas e estratégias de evolução de sistemas legados manualmente, após a leitura do artigo completo e o outro especialista analisando os trechos, para verificar se estavam adequados. A Tabela 3 demonstra alguns exemplos de trechos textuais extraídos.

Table 3: Exemplos de trechos textuais

ID	Trecho textual em Inglês
89	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.
90	Indeed, legacy systems are the most important ones for the day-to-day business operations. The old-fashioned COBOL language is predominant for such systems since it is very efficient to process millions of batch transactions per day, which is the core activity of any financial organization. Such systems are truly business critical. They are reliable and have been running for decades, have been well tested in time, and run virtually with no errors.

Dos 87 estudos selecionados no mapeamento, obteve-se um total de 27 trechos de definições sobre sistemas legados, 25 trechos com problemas e 59 trechos com possíveis estratégias, totalizando 111 trechos textuais¹. Após esta extração, os trechos foram analisados através de procedimentos da GT.

3.2 Análise de Dados

Neste trabalho foram utilizadas 2 fases da GT versão Straussiana, a codificação aberta e a codificação axial. A primeira fase serve para a identificação dos códigos e agrupamentos deles em níveis mais amplos, diminuindo assim a quantidade de códigos que denotam o mesmo sentido. Já a segunda fase serve para a identificação dos tipos de códigos extraídos anteriormente (definições, problemas ou estratégias) e para a análise das relações existentes entre os grupos

¹Planilha com trechos textuais: <https://docs.google.com/spreadsheets/d/1i5y7QdcTDSa68RBipGS3ep2OH8g6ii59Yg6rjQn14WQ/edit?usp=sharing>

criados. A fase de Codificação Seletiva não foi considerada relevante para os objetivos desta pesquisa pois contempla atividades como a identificação de categoria central e a revisão do modelo visual que foram desenvolvidas ao longo do processo.

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 [3], ou seja, é uma habilidade de identificar categorias ou agrupamentos contendo códigos semelhantes que exemplificam características sobre o tema estudado.

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

3.2.1 Codificação Aberta. Nesta atividade buscou-se analisar cada trecho coletado a fim de identificar conceitos (códigos), ou seja, buscou-se através da leitura do trecho descobrir uma ou mais 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 possuíam 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 3 e explicados nas subseções seguintes.



Figure 3: Processo de Codificação Aberta

A codificação aberta foi realizada de uma forma colaborativa com o intuito de torná-la mais ágil, consensual e consequentemente confiável. Assim, não foram apenas 2 pesquisadores analisando os trechos e extraíndo códigos, e sim um número maior de pessoas (codificadores) envolvidas no processo.

Identificar Códigos. A identificação dos códigos foi realizada através de um processo colaborativo, com o objetivo de torná-la mais confiável e factível em um determinado 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. A Figura 4 demonstra esse processo.

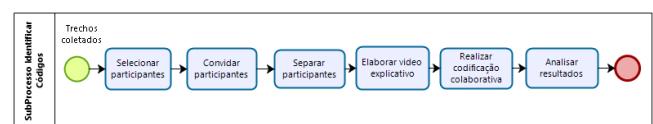


Figure 4: SubProcesso Identificar Códigos

Selecionar participantes: O processo iniciou-se com a seleção dos participantes que fariam as análises dos trechos textuais e as criações dos códigos relevantes a cada trecho. A escolha levou em consideração a existência de algum conhecimento prévio sobre sistemas legados. Logo, optou-se por escolher graduandos do curso de Engenharia de Software que já tinham cursado a disciplina de Evolução de Software e graduandos do curso de Ciência da Computação que já tinham cursado a disciplina de Engenharia de Software II. Partiu-se do princípio que, por já terem cursado tais disciplinas e já estarem nos semestres finais, possuiriam maior senso crítico para a avaliação dos trechos. Assim, foram selecionadas 15 pessoas que atendiam a esses critérios.

Convidar participantes: Com a seleção dos participantes concluída, foram realizados os convites formalmente, sendo obtidas 14 aceitações de colaboração.

Separar Participantes: Após o recebimento das aceitações, cada participante foi alocado a um grupo e a cada grupo foi atribuído um conjunto de trechos textuais, conforme descrito na Tabela 4. O número de grupos foi definido em função da quantidade e do tamanho dos trechos textuais. Definiu-se que cada grupo deveria ser composto por 3 participantes, de forma que as decisões de codificação não ficassem polarizadas e que cada grupo teria condições de codificar entre 25 a 30 trechos. Como alguns trechos eram bem maiores, optou-se por alocar uma quantidade menor nos grupos D e E.

Table 4: Relação de grupos para codificação colaborativa.

Grupo	Número de participantes	Quantidade de Trechos
A	3	30
B	3	29
C	2	25
D	3	13
E	3	14

Elaborar vídeo explicativo: Foi elaborado um material explicativo com os procedimentos que deviam ser seguidos para a realização do processo de codificação colaborativa, sendo disponibilizado para cada participante através de um *link* na plataforma *Youtube*. No vídeo foram dadas explicações teóricas sobre o processo de codificação, tais como: o que são trechos textuais, o que são códigos, exemplos de códigos, bem como uma breve demonstração do uso da ferramenta que seria utilizada.

Realizar codificação colaborativa: A codificação colaborativa foi realizada através da ferramenta *Open Code Tool*², desenvolvida especificamente para esta atividade. Para acessar a aplicação cada participante recebeu um *login* e senha. Após realizado o acesso, cada participante teve acesso a um grupo (Tabela 4), ao qual já havia sido alocado um conjunto de trechos textuais. Cada participante codificador, a partir de uma análise individual, podia criar um ou mais códigos relacionados a um determinado trecho textual. Cada código criado deveria ser salvo.

Cabe destacar que a ferramenta possibilitou aos participantes o acesso e a realização do processo de codificação de qualquer lugar e a qualquer momento que desejasse. Por exemplo, caso o participante estivesse codificando um trecho “x” e precisasse fechar a ferramenta, quando retornasse a acessá-la, seu trabalho estaria no

²Ferramenta Open Code Tool: <http://200.132.136.14/legacysystemreport/public/>

ponto que o participante parou. Ao término das análises textuais e criações de códigos, o participante deveria finalizar seu processo de codificação.

Analizar resultados: Após cada finalização de codificação dos participantes, os códigos criados ficavam disponíveis na ferramenta para serem avaliados e chancelados por avaliadores especialistas no domínio. Os avaliadores tinham a opção de aceitar ou não cada código criado. Eles também podiam criar códigos de acordo com suas análises, bem como editar códigos já criados pelos participantes.

Através desse processo foi possível obter um total de 237 códigos avaliados e compilados³. Desse total cerca de 23% foram códigos novos criados pelos avaliadores. Percebe-se, portanto, que 77% dos códigos foram criados pelo grupo de codificadores.

Identificar agrupamentos. Após a identificação dos conceitos (códigos), iniciou-se uma fase para agrupar os mesmos em grupos considerados mais amplos, identificando características comuns entre os conceitos capturados. O objetivo principal desta fase é diminuir numericamente o número de conceitos obtidos. Segundo [15] a categorização permite reduzir o número de unidades de análise trabalhadas, objetivando a clareza.

Foram definidos dois níveis de agrupamentos aninhados, são eles: Subgrupo Nível 2, que são os agrupamentos formados por códigos semelhantes e cuja frequência é maior que 1, se ligando diretamente ao subgrupo nível 1; Subgrupo Nível 1, que são os agrupamentos que podem ser formados por subgrupos nível 2 ou por apenas códigos isolados que não puderam ser agrupados, possuindo ligação direta com um grupo de nível mais alto (explicitado na seção seguinte).

A Tabela 5 ilustra como os códigos foram agrupados nos vários níveis. Observa-se que os códigos “typically written in assembly”, “written in a third-generation language” e “COBOL language in predominant” foram agrupados primeiramente em “Obsolete language” (subgrupo nível 2), que por sua vez foi agrupado em “Obsolete Technology” (subgrupo nível 1). Os códigos “vital to our organization” e “most important ones for the day-to-day business operations” foram agrupados diretamente ao subgrupo nível 1 “Vital to the organization” porque não foram detectados agrupamentos diferentes.

Table 5: Agrupamento de códigos

ID	Código	Subgrupo Nível 2	Subgrupo Nível 1
89	“large software systems”		Large system
	“vital to our organization”		Vital to the Organization
	“typically written in assembly”	Obsolete language	Obsolete Technology
	“written in a third-generation language”	Obsolete language	Obsolete Technology
90	“most important ones for the day-to-day business operations”		Vital to the Organization
	“COBOL language is predominant”	Obsolete language	Obsolete Technology

³Planilha de códigos extraídos: <https://docs.google.com/spreadsheets/d/1KN3bgTs5zF9c3KbUFpzTMzJmAYLkaeskCl2vXyuLkU/edit?usp=sharing>

Em cada nível foi possível encontrar vários agrupamentos distintos, totalizando 20 subgrupos. Aproximadamente 48% dos códigos não foram agrupados em algum dos subgrupos, pois sua frequência não foi maior que 1.

3.2.2 Codificação axial. Nesta atividade os grupos identificados anteriormente foram novamente analisados a fim de identificar a que tipo pertenciam as relações entre eles.

Na codificação axial, [15] sugerem que os códigos sejam estruturados através da criação de modelos. [3] relata que “modelos são estruturas que tentam explicar o que foi identificado como aspectos fundamentais de um fenômeno 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 5, pode-se visualizar como foi realizado o processo de análise dos dados desta etapa.

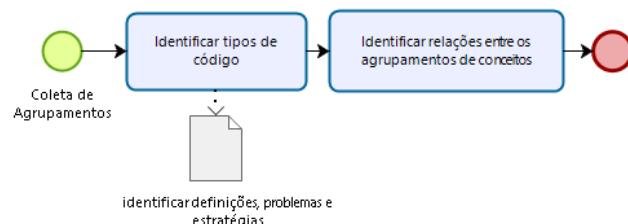


Figure 5: Processo de Codificação Axial

Identificar tipos de código. Nesta atividade ocorreu a identificação dos tipos de códigos. [15] sugerem a existência 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 cada subgrupo e código identificados na codificação aberta, foi realizada uma análise para determinar o seu tipo de código (aqui denominado de grupo). Neste estudo, o tipo Condições causais foi denominado *Definition*, o tipo Consequências de *Problem* e o tipo Estratégias de *Strategy*, tendo sido detectados somente esses três tipos. Assim, todos os subgrupos e códigos pertencem a algum desses três tipos.

Identificação de relações entre os agrupamentos de conceitos. Esta atividade contempla a identificação de relações entre os grupos, subgrupos e códigos. As relações receberam uma nomenclatura de acordo com a percepção dos avaliadores. Foram criadas as seguintes nomenclaturas para as relações:

- **Consequence:** relação que caracteriza que um elemento (código ou subgrupo) é consequência de outro elemento.
- **Benefits:** relação que identifica que um elemento é benefício de outro elemento.

3.3 Ferramentas metodológicas

Neste trabalho foi utilizada uma ferramenta desenvolvida especificamente para a atividade colaborativa de codificação aberta. A *Open Code Tool* possui as seguintes funcionalidades: Importação dos Trechos textuais; Visualização dos trechos textuais; Criação,

edição e exclusão de códigos; Chancela da codificação e Geração de relatórios.

4 RESULTADOS

Após a codificação aberta, onde os códigos identificados foram agrupados primeiramente em um nível denominado de subgrupo 2 e posteriormente reagrupados em um nível superior, chegou-se aos seguintes subgrupos nível 1⁴:

Obsolete Technology: agrupa subgrupos compostos por códigos que denotam características relacionadas ao uso ou construção de sistema com tecnologia obsoleta, este subgrupo possui cerca de 8% dos códigos analisados. Ligados a este subgrupo foram identificados dois subgrupos nível 2, com códigos que denotam características mais específicas, são eles: *Obsolete language*, que contém códigos relacionados ao uso de linguagens de programação obsoletas em sistemas legados; *Obsolete paradigm*, que contém códigos relacionados a paradigmas de projeto e desenvolvimento.

Old system: relacionado a códigos que enfatizam o tempo como fator de identificação de sistemas legados. Cerca de 4% dos códigos analisados possuem características relacionadas a este subgrupo.

Documentation: Contém códigos relacionados à documentação inexistente, incompleta ou desorganizada. Cerca de 3% dos códigos analisados estão neste subgrupo.

Vital to the Organization: Agrega códigos que denotam a importância de sistemas legados à organização. Cerca de 2% dos códigos analisados estão neste subgrupo.

Currently Used: relacionado a códigos que caracterizam sistemas legados como sendo utilizado atualmente. Aqui estão 2% dos códigos analisados.

Large system: agrupa códigos que caracterizam sistemas legados como sistemas grandes, com uma quantidade excessiva de linhas de código, de componentes, etc. Cerca de 1% dos códigos possuem características relacionadas a este subgrupo.

Costs: códigos que caracterizam os custos de investimentos que sistemas legados estão aqui agrupados. Cerca de 2% dos códigos analisados estão neste subgrupo.

Maintenance: formado por códigos que denotam diversos problemas de manutenção de legados. Cerca de 4% dos códigos analisados possuem características relacionadas a este subgrupo.

Developers: formado por códigos que evidenciam a falta de habilidades e conhecimentos de desenvolvedores para lidar com a manutenção de legados. Cerca de 3% dos códigos analisados possuem essas características.

Old hardware: Agrupa códigos relacionados a problemas decorrentes da utilização de hardware antigo para rodar sistemas legados. Cerca de 1% dos códigos estão neste subgrupo.

Migration: relacionado a códigos que evidenciam a migração de sistema legado como uma estratégia que deve ser levada em consideração para evoluir tais sistemas. Aproximadamente 12% dos códigos possuem características relacionadas a este subgrupo. *Cloud* foi o tipo de migração mais citada entre esta estratégia, com cerca de 53% dos códigos, assim se tornando um subgrupo nível 2 ligado a este.

⁴Planilha agrupamentos de códigos: https://docs.google.com/spreadsheets/d/1p3jjYp45681fPaUcWBZdPj9kRfN13l_McAAL_0hxKQ/edit?usp=sharing

Replacement: Agrega códigos relacionados à estratégia do tipo substituição. Cerca de 3% dos códigos analisados possuem essas características. A substituição de partes ou componentes de um legado foi o código mais encontrado mais de uma vez, sendo agrupado subgrupo nível 2 *Obsolete Component*. O mesmo aconteceu com os códigos que indicam começar um novo desenvolvimento, sendo agrupados em *Prefer to start fresh*.

Integration: Agrupa códigos relacionados a estratégias de integração de legados à novos sistemas e tecnologias. Cerca de 2% dos códigos possuem essas características.

Conversion: Contém códigos relacionados a estratégias de conversão de legados a novos paradigmas e tecnologias. Cerca de 1% dos códigos possuem essas características.

Reengineering: agrupa códigos relacionados a estratégias de reengenharia de sistemas legados. Cerca de 1% dos códigos possuem essas características.

Disposal process: relaciona códigos que denotam o descarte de todo ou de parte do sistemas. Cerca de 1% dos códigos possuem essas características.

Como resultado da codificação axial, obteve-se a tipificação dos agrupamentos e códigos isolados e os relacionamentos entre eles⁵. Todos os subgrupos e códigos isolados foram considerado “tipo” de algum dos seguintes três tipos (grupos): *definition*, *problem* ou *strategy*. Seis (6) subgrupos foram classificados como *definition*, quatro (4) subgrupos como *problem* e seis (6) como *strategy*.

Com os subgrupos e códigos tipificados, diversos relacionamentos entre códigos emergiram, tais como, códigos sendo consequência (*consequence*) de outro, códigos sendo benefícios (*benefits*) de outro. Os subgrupos relacionados e tipificados em grupos com suas respectivas frequências, e os relacionamentos foram representados na forma de modelos visuais individuais para cada grupo e de um modelo visual geral. Nos modelos, a cor verde especifica os subgrupos de nível 2 imediatamente relacionados ao grupo (tipo) principal, destacado com a cor verde. As subseções seguintes detalham como a tipificação e os relacionamentos foram estabelecidos e apresenta os respectivos modelos visuais.

4.1 Tipo *Definition*

Entendeu-se que os agrupamentos tecnologia obsoleta, sistema antigo (velho), vital para a organização, sistema grande, sistema em uso atualmente e documentação são características que definem um sistema legado. Tecnologias obsoletas aparecem com maior frequência e são do tipo linguagens (Cobol, Fortran, C, C++, Assembly), paradigmas (procedimentos e funções) ou hardware obsoleto. Este último, relaciona-se com problemas como lentidão e custo. Sistemas antigos são aqueles desenvolvidos desde a década de 70, ainda em uso pelas organizações, e que são considerados vitais as mesmas. O tamanho do sistema, seja pelo número de linhas de código ou pela quantidade de componentes, também é evidenciado como uma característica, assim como a documentação inexistente ou desatualizada. Alguns aspectos de documentação geram consequências ou problemas como a dificuldade de compreender e manter o sistema e a própria documentação.

⁵ Planilha de agrupamentos/tipos de códigos: https://docs.google.com/spreadsheets/d/1vAOuZkDpAkgjNZF3q_gyO7-VXlskhZAGaX4elB2bbCU/edit?usp=sharing

Os códigos e subgrupos vinculados ao tipo *Definition* foram representados em um modelo visual⁶, cuja versão resumida é apresentada na Figura 6.

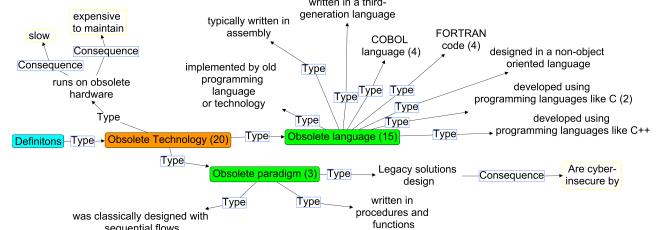


Figure 6: Modelo visual *Definition*

4.2 Tipo *Problem*

Identificou-se que manutenção, custos, hardware antigo e desenvolvedores são problemas decorrentes de sistemas legados. O problema de manutenção se destaca principalmente pela dificuldades de compreensão e atualização de um sistema legado. Atualizações geram consequências como inserção de mais erros no código, necessidade de mais tempo, complexidade e necessidade de investimentos de alto custo. Custos também é tipo de problema expressivo, sendo consequência de vários outros problemas, como a falta de entendimento do sistema por parte dos desenvolvedores.

Problemas relacionados a desenvolvedores referem-se a falta de habilidades para a manutenção de legados, que causam consequências como a dificuldade de compreensão, modificação e manutenção do código-fonte (problema de manutenção); a flutuação ou rotatividade de desenvolvedores que causam consequências como o objetivo original do sistema se perder e a manutenção se tornar problemática (problema de manutenção); a falta de entendimento geral, que traz com consequência o aumento dos custos da manutenção (problema de custo); o conhecimento incompleto sobre o sistema faz com que novos erros introduzidos precisem ser corrigidos e isso torna a manutenção problemática (problema de manutenção).

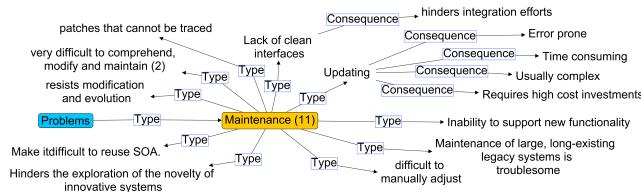
Problemas específicos decorrentes do hardware antigo onde os legados rodam são a performance lenta, que causa como consequência a redução de produtividade; a careza para manter e falta de componentes de hardware que não estão mais em produção.

Outros elementos foram identificados como problemas decorrentes de sistemas legados, como por exemplo, a inflexibilidade que tais sistemas possuem trazendo como consequência arquiteturas rígidas, possuírem códigos que vão se deteriorando ao longo do ciclo de vida tornando a manutenção problemática, entre outros.

Os códigos e subgrupos vinculados ao tipo *Problem* foram representados em um modelo visual⁷ cuja versão resumida é apresentada na Figura 7.

⁶ Modelo visual *Definition*: <https://drive.google.com/file/d/1NjdypTKg0gt0XGJcp0cmS4RgxIw0Ky1-/view?usp=sharing>

⁷ Modelo visual *Problems*: https://drive.google.com/file/d/14H7O34_FHMtswKwF3K0_h7QHNxaOjhLC/view?usp=sharing

Figure 7: Modelo visual *Problems*

4.3 Tipo *Strategy*

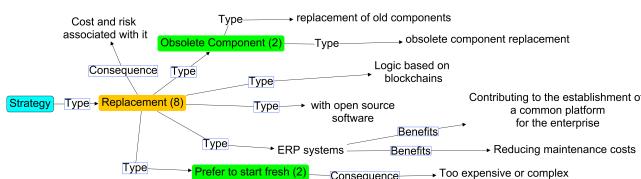
Integração, conversão, migração, substituição, desenvolver novo sistema e reengenharia foram consideradas estratégias de evolução de sistemas legados. A migração é a estratégia mais frequente e se manifesta de várias formas, sendo a migração de legados para a nuvem a mais relatada nos estudos analisados. Identificou-se também 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, como o custo, assim como os desafios do gerenciamento de aspectos técnicos e de negócios deste tipo de migração.

Outra estratégia também relatada com 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 participar um legado migrando o mesmo para recursos reutilizáveis. Substituir legados por software de código aberto, substituir legado por sistemas que integram todos os dados e processos da organização em um único sistema, conhecidos como sistemas *ERP - Enterprise Resource Planning*. Esta estratégia tem como consequências o alto risco e custos associados.

A integração de legados para algum elemento específico também obteve uma frequência considerável nesse estudo. Alguns exemplos são a integração de legados com novas tecnologias, como *IoT - Internet of Things*, com sistemas *ERP*, dentre outras.

Converter um legado também foi identificado como estratégia de evolução, seja para um paradigma orientado a objetos ou no caso de uma conversão de dados para um protocolo *IIoT - Industrial Internet of Things*. Alguns tipos de estratégias foram pouco citadas, mas são relevantes ao estudo, como: reimplementação, construir um sistema do zero, refatorar, descartar, entre outros.

Os códigos e subgrupos vinculados ao tipo *Estrategy* foram representados em um modelo visual⁸ cuja versão resumida é apresentada na Figura 8.

Figure 8: Modelo visual *Strategy*

⁸Modelo visual *Strategy*: https://drive.google.com/file/d/1w0jBalM4pijlwgAfMq0B4cWfd-3LDoz_/view?usp=sharing

4.4 Entendimento sobre Sistemas Legados

Com os tipos *Definition*, *Problem* e *Strategy* definidos, restou estabelecer a ligação com o fômeno de estudo principal, *Legacy System*. Assim, percebeu-se por quais características sistemas legados são definidos, quais os problemas que apresenta e quais as estratégias de evolução, que subsidiaram a proposição de uma teoria acerca de sistemas legados.

Um sistema pode ser considerado legado através da tecnologia obsoleta utilizada para o seu desenvolvimento, pelo tamanho do sistema, pelo tempo em que foi desenvolvido e continua em operação, pelo status da documentação e pela importância que representa para a organização.

Sistemas legados ocasionam problemas, como a dificuldade de dar manutenção, em muito decorrente da falta de conhecimento especializado dos desenvolvedores, e isso gera como consequência um alto custo financeiro.

Problemas podem ser mitigados com estratégias de evolução de sistemas legados. Existem diversas possibilidades como a migração, a reengenharia, a substituição, a aposentadoria, as quais apresentam riscos e benefícios. A migração para a nuvem é atualmente uma estratégia em evidência, que vem sendo adotada pelas organizações.

Para apoiar o entendimento geral, mais conciso, sobre sistemas legados, foi criado um modelo visual geral⁹, como mostra apêndice A. Este modelo contém todos os agrupamentos e relacionamentos já descritos, mas também permite a visualização das relações entre os grandes agrupamentos (tipos).

5 AMEAÇAS À VALIDADE

No decorrer do desenvolvimento deste trabalho foi possível identificar algumas ameaças à sua validade, são elas:

- Strings de busca: Na base digital *Science Direct* houve alterações na string pois há limitações no mecanismo de busca desta base digital. Assim, alguns trabalhos podem não ter sido recuperados.
- Critério de Inclusão: Foi definido que os estudos fossem publicados no ano de 2018. Essa decisão foi tomada para tornar o trabalho de leitura, extração dos trechos textuais e codificação factível no período de tempo de 12 meses. Assim, entende-se que os resultados do trabalho expressem um recorte de todo o conhecimento disponível acerca do objeto de estudo desta pesquisa.
- Procedimentos da GT: A GT é um método qualitativo, até então pouco explorado no âmbito da Engenharia de Software. Foi realizado um estudo exaustivo sobre sua origem, vertentes e aplicações. Contudo, poucos estudos demonstram com clareza como aplicar o método. Dessa forma, entende-se que algumas interpretações podem, de alguma forma, impactar os resultados. Além disso, a própria natureza deste método qualitativo traz uma subjetividade grande, principalmente nos procedimentos de codificação, onde a experiência dos codificadores é essencial.

⁹Modelo visual geral: <https://drive.google.com/file/d/1xBr7PQ6UefNJ8fu2PQTZfd-zEFUL4RqF/view?usp=sharing>

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho tem como objetivo oferecer um entendimento mais conciso sobre as características que definem sistemas legados, assim como os problemas que causam e as possíveis alternativas de evolução. 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 uma teoria que auxilia nesse entendimento de forma compartilhada e única.

As lacunas observadas no trabalho relacionado foram supridas abordando-se o problema de entendimento de sistemas legados através de uma perspectiva metodológica diferente. Na coleta de dados, buscou-se extraír trechos textuais que fizessem menção não só a características gerais, mas também a problemas e estratégias de solução de sistemas legados. Na análise dos dados qualitativa, o uso da GT propiciou que as categorias encontradas não ficasse isoladas e sim vinculadas através de relações de causa e efeito.

Entende-se que tanto o modelo visual como a teoria são artefatos que podem auxiliar um processo de identificação de sistemas que são ou estão se tornando legados. Atualmente este é um processo complexo, uma vez que não existe um consenso sobre os elementos que caracterizam esse tipo de sistema. Essa identificação pode se dar, tanto pelas características que o definem como pelos problemas que ocasionam. Ao reconhecer que os sistemas de uma organização já apresentam alguns elementos apresentados no modelo, o tomador de decisão pode agir de forma a mitigá-los a tempo. O modelo também pode auxiliar proporcionando uma visão geral das alternativas de solução ou evolução de sistemas legados.

Os desafios encontrados durante a condução desta pesquisa relacionam-se ao uso da GT como abordagem qualitativa de análise de dados. Percebeu-se que há uma carência de trabalhos que mostrem a aplicação do desenvolvimento desta metodologia, mais especificamente, não foram encontrados muitos trabalhos que utilizem a GT na investigação de problemas do domínio da Engenharia de Software.

Neste sentido, entende-se que além do modelo visual e da teoria gerada, este trabalho tem como contribuição a apresentação de um processo para análise de dados qualitativos que utiliza procedimentos da GT. O processo, tal como apresentado, é genérico e suficiente para guiar outros pesquisadores que necessitem adotar esses procedimentos em seus problemas de pesquisa.

Os resultados deste trabalho são baseados em um recorte específico da literatura, referente ao ano de 2018. Entende-se que com o passar do tempo as visões sobre sistemas legados podem sofrer alterações. Assim, como trabalho futuro, pretende-se ampliar a coleta e extração de dados para outros anos, o que tornará a teoria construída mais robusta e o entendimento acerca de sistemas legados mais confiável. Adicionalmente, para viabilizar uma extração de trechos textuais mais ampla, propõe-se o desenvolvimento de um módulo na ferramenta *Open Code Tool* que permita a extração de trechos textuais de forma automática ou semi-automática.

Por fim, é possível vislumbrar o desenvolvimento 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 deve ser alimentada com dados reais, de acordo com os elementos definidos no modelo, e deve emitir um alerta quando um determinado número ou percentual relacionado

à proximidade do sistema ser ou se tornar legado for atingido. A aplicação também pode indicar o que pode ser feito para evoluir esse sistema.

AGRADECIMENTOS

Agradecemos aos voluntários pela colaboração neste trabalho e ao Dr. Igor Fabio Steinmacher pelas valiosas contribuições no desenvolvimento deste artigo.

REFERENCES

- [1] Bennett. 1995. Legacy systems: coping with success. *IEEE Software* 12, 1 (1995), 19–23.
- [2] Kath Charmaz. 2009. *A construção da teoria fundamentada - Guia Prático para Análise Qualitativa*. Porto Alegre: Artmed.
- [3] Graham Gibbs. 2009. *Análise de Dados Qualitativos*. Porto Alegre: Artmed.
- [4] B.G. Glaser and A.L. Strauss. 1967. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. New York: Aldine.
- [5] M Malki, S M Benslimane, and M Abdelkader. 2013. A heuristic approach to locate candidate web service in legacy software. *International Journal of Computer Applications in Technology* 47 (2013).
- [6] Daniele Martins, Alex Chervenski, and Andréa Bordin. 2017. Identificação de características de sistemas legados a partir da análise de conteúdo da literatura. In *Anais da I Escola Regional de Engenharia de Software*. SBC, 81–88.
- [7] P O'Byrne and B Wu. 2000. 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).
- [8] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic mapping studies in software engineering. *EASE'08 Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering* (2008), 68–77.
- [9] R Pérez-Castillo, B. Mas, and M. Pizka. 2015. Understanding legacy architecture patterns. In *2015 International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*. 282–288.
- [10] Anand Rajavat and Vrinda Tokekkar. 2014. Investigation of quality and functional risk issues in reengineering process of legacy software system. *International Journal of Programming Languages and Applications (IJPLA)* 4 (2014).
- [11] J. Ransom, I. Somerville, and I. Warren. 1998. A method for assessing legacy systems for evolution. In *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering*. 128–134.
- [12] P. A. Sandborn and V. J. Prabhakar. 2015. The Forecasting and Impact of the Loss of Critical Human Skills Necessary for Supporting Legacy Systems. *IEEE Transactions on Engineering Management* 62, 3 (2015), 361–371.
- [13] JLG Santos, KS Cunha, EK Adamy, MTS Backes, JL Leite, and FGM Sousa. 2018. Análise de dados: comparação entre as diferentes perspectivas metodológicas da Teoria Fundamentada nos Dados. *Revista da Escola de Enfermagem da USP* (2018).
- [14] Klaas-Jan Stol, Paul Ralph, and Brian Fitzgerald. 2016. Grounded Theory in Software Engineering Research: A Critical Review and Guidelines. *IEEE International Conference on Software Engineering* (2016).
- [15] Anselm Strauss and Juliet Corbin. 1998. *Basics of Qualitative Research : Techniques and Procedures for Developing Grounded Theory* 2nd ed. Londres: Sage Publications.
- [16] Anselm Strauss and Juliet Corbin. 2008. *Pesquisa qualitativa: técnicas e procedimentos para o desenvolvimento de teoria fundamentada*. Porto Alegre: Artmed.
- [17] G. Wang, R. Valerdi, and J. Fortune. 2010. Reuse in Systems Engineering. *IEEE Systems Journal* 4, 3 (2010), 376–384.
- [18] J. Zhao, Z. Zhao, and H. Yang. 2018. Distributed Parallelizability Analysis of Legacy Code. In *2018 IEEE Int'l Conf on Parallel Distributed Processing with Applications, Ubiquitous Computing Communications, Big Data Cloud Computing, Social Computing Networking, Sustainable Computing Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*. 103–110.

A MODELO VISUAL GERAL

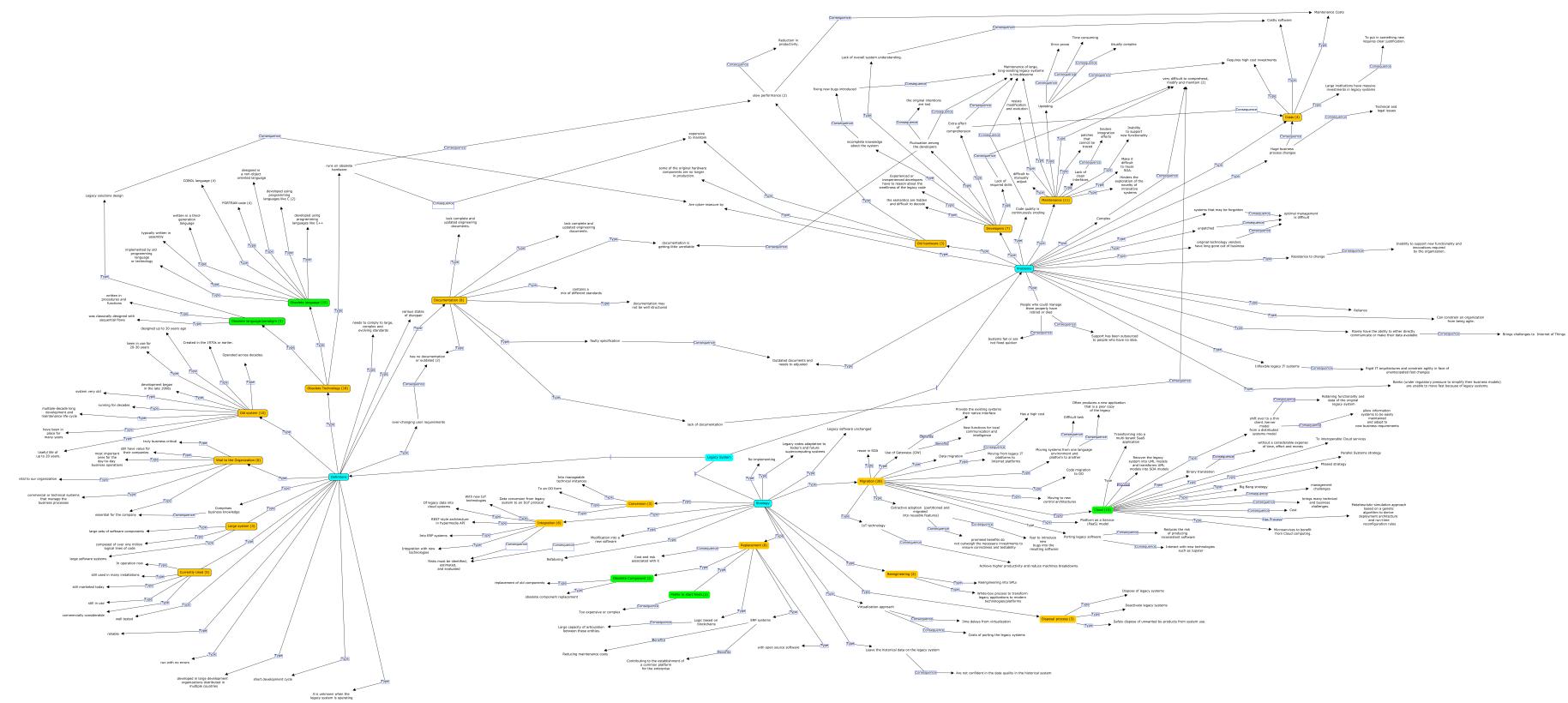


Figure 9: Modelo visual geral de conceitos