Empirical study on architectural degradation in the Brazilian industry: A survey

Anderson da Silva Cossul
acossul@edu.unisinos.br
Universidade do Vale do Rio dos Sinos
São Leopoldo, Rio Grande do Sul, Brazil

Kleinner Farias
kleinnerfarias@unisinos.br
Universidade do Vale do Rio dos Sinos
São Leopoldo, Rio Grande do Sul, Brazil

ABSTRACT

The problem of architectural degradation in the software development industry refers to the gradual deterioration of a system's structure and quality over time. This deterioration can have a direct impact on the quality of software, resulting in bugs and low maintenance productivity by the development teams. In this context, this work aims to carry out an empirical study on architectural degradation in Brazilian industry, through a survey using questionnaires and interviews, aiming to understand the perception of professionals working in this area in relation to the issue. The main results obtained reveal that professionals are aware of the existence of this problem and identify the main causes of architectural degradation, with the lack of technical knowledge in software architecture and the absence of updated documentation being the most critical. In addition, the study seeks to understand the perception of professionals regarding the consequences of architectural degradation and presents recommendations to avoid it. Additionally, we seek to investigate whether these professionals have already made the conscious decision to degrade an architecture and the reasons that led them to this choice, being tight deadlines the main one. By contributing to a deeper understanding of architectural degradation and the perceptions of industry professionals, this study offers subsidies for the development of strategies and good practices that can mitigate architectural degradation, improve the quality of software and increase the efficiency in maintaining systems over time.

KEYWORDS

Architectural degradation; Software architecture; Empirical study

1 INTRODUÇÃO

Atualmente, o desenvolvimento de sistemas de *software* ocorre em ambientes de negócio cada vez mais instáveis na indústria. Neste contexto, requisitos de software complexos e voláteis, pressões por ciclos de desenvolvimento mais curtos, tecnologias e modelos arquiteturais que evoluem rapidamente são características sempre presentes [24]. A arquitetura de *software* desempenha um papel crucial no desenvolvimento e manutenção de tais sistemas. No entanto, à medida que um sistema evolui, o fenômeno da degradação arquitetural pode ocorrer. Esse fenômeno ocorre quando há uma discrepância entre a arquitetura proposta e a arquitetura efetivamente implementada de um *software*. A degradação arquitetural precisa ser identificada e tratada, uma vez que pode resultar em diversos efeitos indesejados na qualidade do sistema, inclusive encurtando sua vida útil [2], comprometendo a estabilidade arquitetura [9] e aumentando o esforço de manutenção de software [8, 10]. Os problemas de degradação arquitetural na indústria de desenvolvimento de *software* têm suas raízes na própria natureza do processo de desenvolvimento. Muitas vezes, a arquitetura originalmente definida não consegue atender às necessidades de evolução do sistema, o que acaba levando à degradação arquitetural ao longo do tempo. Essa falta de alinhamento entre a arquitetura e as necessidades em constante mudança é um desafio comum enfrentado pelos desenvolvedores [7, 29].

Este estudo abordou a problemática da degradação arquitetural na indústria brasileira de desenvolvimento de *software*, utilizando um *survey* para coletar percepções dos profissionais sobre suas causas, consequências e recomendações para evitá-la. Além disso, buscou-se compreender se os profissionais já haviam deliberadamente degradado uma arquitetura e, em caso afirmativo, quais foram os motivos que os levaram a tomar essa decisão. Também foi explorado o uso e entendimento das práticas efetivas e ideais pelos participantes em relação à representação e compreensão da arquitetura de *software*. Um questionário online foi aplicado e foram obtidas 347 respostas totais. Essas respostas foram analisadas e utilizadas como os dados quantitativos da pesquisa. Após essa análise, foram geradas perguntas a serem utilizadas em entrevistas para aprofundar possíveis lacunas encontradas. Além disso, 5 profissionais foram entrevistados e suas respostas foram utilizadas como dados qualitativos da pesquisa, sendo triangulados com os resultados quantitativos encontrados no questionário online.

Os principais resultados encontrados revelam que os profissionais estão cientes da degradação arquitetural e já tiveram que lidar com essa questão em seus ambientes profissionais. Além disso, a falta de conhecimento técnico em arquitetura de *software* e a ausência de documentação atualizada foram apontadas como as principais causas relatadas pelos participantes para a ocorrência da degradação arquitetural. Também foi possível observar que alguns participantes deliberadamente comprometeram a arquitetura, sendo o prazo a principal razão que os levou a tomar essa decisão. Assim, este trabalho estabelece uma conexão entre a literatura existente e a experiência prática de profissionais brasileiros da área de desenvolvimento de *software*, proporcionando uma visão aplicada ao cotidiano desses profissionais. Dessa forma, o estudo beneficia pesquisadores e profissionais envolvidos no desenvolvimento de sistemas, ao explorar o impacto da degradação arquitetural na qualidade e manutenção de *softwares* em desenvolvimento ativo.

O trabalho foi dividido em 5 seções. A seção 2 define a fundamentação teórica. A seção 3 define os trabalhos relacionados. A seção 4 define a metodologia utilizada. A seção 5 define os resultados de cada questão de pesquisa. Por fim, a seção 6 apresenta a conclusão e o direcionamento para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção busca explicar conceitos importantes que serão utilizados durante o trabalho de acordo com a literatura atual disponível.

2.1 Degradação Arquitetural

Degradação arquitetural acontece quando a implementação da arquitetura de algum *software* diverge da arquitetura pretendida. Essa degradação pode ocorrer por diversos motivos, como a evolução natural do *software* e a necessidade de implementação de funcionalidades novas que divergem do planejado inicial. Essa degradação pode ser considerada desvio arquitetural quando há simplesmente uma divergência entre a arquitetura documentada e a implementação. Já erosão arquitetural é um desvio arquitetural em sua essência, porém também violando regras documentadas pela arquitetura prescritiva [14]. Os efeitos da degradação arquitetural são diversos como inconsistências que acabam comprometendo diretamente a qualidade do *software* final. Também gera um impacto grande no time de desenvolvimento, uma vez que há uma dificuldade em manter e evoluir um *software* que tenha sua arquitetura degradada da arquitetura pretendida [19].

2.2 *Survey*

Um *survey* é um sistema para coletar informações de ou sobre pessoas para descrever, comparar ou explicar seus conhecimentos, atitudes e comportamentos. O principal meio de coleta de dados qualitativos ou quantitativos são entrevistas e questionários. Essas atividades são aplicadas em uma amostra representativa da população a ser estudada. Os resultados da pesquisa são então analisados para derivar informações descritivas e explicativas conclusões. Eles são então generalizados para a população da qual a amostra foi levado [12]. *Surveys* são uma das formas de conduzir um estudo empírico em formato de retrospecção e seus benefícios são observados por possibilitar coletar informações de pessoas para descrever, comparar ou explicar seus conhecimentos, atitudes e comportamentos.[33]

3 TRABALHOS RELACIONADOS

A pesquisa pelos trabalhos relacionados foi realizada em repositórios digitais como *Google Scholar* e *Scopus (Elsevier)*. Grande parte dos trabalhos selecionados foram resultados de pesquisas pelo termo "architectural degradation survey". Esse termo foi utilizado como parte da estratégia de busca na pesquisa por trabalhos relacionados devido ao seu papel fundamental em delinear o escopo e o foco da investigação, uma vez que é uma chave que encapsula o tema central que está sendo explorado, ou seja, a degradação arquitetural. A metodologia adotada para analisar os trabalhos relacionados tem sido previamente validada em estudo previamente publicados na literatura [5, 6, 15, 21, 23, 27, 31, 32].

3.1 Análise dos Trabalhos Relacionados

[2]. Apresenta um estudo voltado a explorar as causas da degradação arquitetural em aplicações *Open Source*. Foi realizada uma pesquisa literária em estudos já existentes na área até o ano de 2020 para tentar classificar os principais pontos que levam à degradação. Os resultados indicaram que diversos fatores são contribuintes, como a rápida evolução de um *software*, mudanças frequentes e a falta de percepção dos desenvolvedores. Também relata que o indicador mais prominente para os sintomas são *code smells* e *architectural smells*.

[19]. Apresenta um trabalho agregador de 73 estudos para analisar e investigar as razões, consequências e formas de detectar e tratar erosões de *software*. Os principais resultados foram que a erosão se manifesta através de violações arquiteturais e estruturais, assim como também foi observado que razões não técnicas causam problemas de erosão arquitetural e essas precisam de atenção assim como as razões técnicas. Também foi pontuado que ainda é necessário mais pesquisas na área para propor técnicas para evitar a erosão.

[4]. Foi realizado um estudo para comparar as diferenças entre uma arquitetura de *software* proposta inicialmente e a percepção dela por novos desenvolvedores. Foi observado diferenças significativas e violações da arquitetura por parte dos novos desenvolvedores, como a comunicação entre os módulos. Foi concluído que a falta de entendimento da arquitetura original por parte dos desenvolvedores é um fator muito perigoso para a qualidade do *software* e também leva à degradação arquitetural.

[20]. Foi realizado um estudo sobre a possibilidade de detectar e corrigir problemas de erosão de *software* no processo de *code review*. Para isso foram analisados diversos *reviews* de dois projetos mais ativos no comunidade *Open Stack: Nova e Neutron*. O resultado indica que discussões sobre erosão de *software* foram pequenas, porém ainda presentes, sendo as mais frequentes violações arquiteturais, funcionalidades duplicadas e dependência cíclica, concluindo que é possível detectar e corrigir esses problemas antecipadamente.

[3]. O estudo seguiu o aprendizado que a maneira mais eficaz de observar problemas em erosão de *software* é via métricas. Contudo, não havia uma abordagem sistêmica para mapear essas métricas. Foram elencadas em torno de 100 métricas e foi proposto uma categorização dessas. Foi concluído que apesar de ter sido possível metrificar as causas da erosão, muitas métricas são ambíguas e ainda precisam ser mais exploradas.

[26]. O trabalho identificou que apesar de já existirem estudos para identificar quando degradações arquiteturais aconteceram, ainda havia oportunidade para entender mais as causas que levam à degradação. Para isso, foi utilizado uma análise recursiva de literatura (*recursive literature review*) e técnicas de *Bootstrapping*. Foi observado que os participantes tomam diversas atitudes para manter a consistência de dados e que uma das principais causas que levam a degradação arquitetural é não revisitar a arquitetura de um *software* constantemente.

[7]. O estudo abordou técnicas para metrificar problemas na erosão de *software*, uma vez que foi observado na indústria que a dinâmica de desenvolvimento muitas vezes acaba não priorizando débitos técnicos, uma vez que é difícil categorizá-los e priorizá-los. Foram analisadas diversas ferramentas na área e foi observado que cerca da metade delas focam em análise de código, sem focar em análise de *design* e arquiteturais.

3.2 Análise Comparativa dos Trabalhos Relacionados

Critério de Comparação. Foram definidos dez Critérios de Comparação (CC) para realizar a comparação de similaridades e diferenças entre o trabalho proposto e os artigos selecionados. Sua comparação é crucial para auxiliar na identificação de oportunidades de pesquisa utilizando critérios objetivos, ao invés de subjetivos. Os critérios são descritos a seguir:

- **Estudo empírico (CC1):** pode ser entendida como aquela em que é necessária comprovação prática de algo, especialmente por meio de experimentos ou observação de determinado contexto para coleta de dados em campo;
- **Contexto (CC2):** este critério busca avaliar se o trabalho realizado é focado em profissionais da indústria de desenvolvimento de *software*;
- **Perfil do participante (CC3):** caracteriza se foram coletados dados dos participantes para caracterização de perfil;
- **Entrevistas (CC4):** indica se foram realizadas entrevistas para a condução do trabalho;
- **Região geográfica específica (CC5):** se o trabalho considera uma região específica, entendendo que as conclusões podem ser relativas àquela região;
- **Causas da degradação arquitetural (CC6):** caso o trabalho tenha focado em compreender mais das causas que levam à degradação arquitetural;
- **Recomendações para evitar a degradação arquitetural (CC7):** indica se o estudo propõe recomendações a fim de combater a degradação arquitetural;
- **Intenção de degradação arquitetural (CC8):** caso o trabalho se propõe a analisar as intenções dos profissionais em conscientemente degradar uma arquitetura;
- **Representação da arquitetura (CC9):** busca entender como os profissionais representariam uma arquitetura de *software*;
- **Ciência da arquitetura (CC10):** caso o trabalho visa entender mais a fundo as práticas e considerações envolvidas numa arquitetura de *software*;

Oportunidades de pesquisa. A Tabela 1 apresenta a comparação dos estudos selecionados, evidenciando o não atendimento de todos os critérios. Foram observadas as seguintes lacunas a partir das comparações: (1) nenhum trabalho contemplou todos os critérios observados, exceto o trabalho proposto; (2) a literatura ainda carece de estudos que exploram as intenções de degradação arquitetural em uma região geográfica específica e (3) poucos estudos exploram como a arquitetura é representada e como os desenvolvedores têm consciência dos componentes arquiteturais. Portanto, este trabalho busca entender através de um *survey* como a arquitetura de *software* é tratada na prática na indústria, principalmente explorando as causas e intenções que levam à degradação arquitetural.

4 METODOLOGIA

Esta seção descreve a metodologia seguida para executar o estudo. A Seção 4.1 apresenta o objetivo do estudo e a questão de pesquisa investigada. A Seção 4.2 detalha o processo experimental seguido, descrevendo as fases do estudo e as atividades executadas. Por fim, a seção 4.3 apresenta mais informações sobre o questionário e as entrevistas. A metodologia adotada foi baseada em orientações empíricas amplamente conhecidas [33] e em estudos anteriores publicados [16, 17].

4.1 Objetivo e Questão de Pesquisa

O objetivo deste estudo é essencialmente avaliar as causas da degradação de *software* na indústria. Em particular, busca-se investigar se os profissionais têm ciência deste efeito e caso sim, se fazem ou já fizeram uma degradação arquitetural conscientemente. Também são investigados possíveis soluções para esse problema de acordo com os resultados da pesquisa, assim como é explorado entender como uma arquitetura é representada na prática.

4.2 Processo Experimental

Fase 1: Seleção de participantes. Participantes foram selecionados baseados nos seguintes critérios: nível de conhecimento, experiência prática com arquitetura de *software* e programação em projetos na indústria de *software*. Usando esses critérios, foram selecionados participantes com histórico acadêmico e experiência prática na indústria brasileira. Este conjunto de participantes representaram o público alvo. Mais especificamente, o público alvo se dá por participantes trabalhando no Brasil, incluindo desenvolvedores (diferentes níveis de senioridade), arquitetos de *software*, todos com histórico acadêmico de universidades brasileiras. Este público representa quem está apto a responder o questionário e a quem os resultados se aplicam. No total, 347 participantes responderam o questionário.

Fase 2: Aplicação do questionário e entrevistas. Esta fase foca na aplicação do questionário e condução das entrevistas. Foram conduzidas entrevistas para coletar dados qualitativos adicionais relacionados às perguntas da pesquisa. Tais dados são essenciais para triangular os resultados obtidos. O questionário foi disponibilizado ao público alvo através de distribuição online através de bancos de e-mails, *Linkedin* e indicação entre participantes.

Fase 3: Análise dos dados. Esta fase procura analisar os dados coletados através dos questionários e entrevistas de maneira cuidadosa. Para isso, primeiro foi analisado os dados coletados (entrevistas e questionário) separadamente e então os comparar (triangulação). Inicialmente, os dados do questionário foram analisados e então tabulados através da técnica de *Ground Theory* [13]. Então, estes dados foram utilizados para formular as questões das entrevistas. Desta maneira, os entrevistados responderam questões que focaram explorar os resultados obtidos através do questionário de maneira mais profundo, buscando consistência na análise de dados. A investigação proporcionou interação por meio de um processo dialético, com interação e reflexão entre o pesquisador e os participantes. A análise dos dados das entrevistas foram feitas de maneira manual e foi objetivada uma troca de uma visão ampla para uma visão mais focal, sem divergências. Isso ajudou a obter evidências complementares para explicar os resultados quantitativos e então derivar conclusões concretas a partir de uma cadeia de evidências formada a partir do alinhamento sistemático de dados quantitativos e qualitativos.

Table 1: Análise comparativa dos Trabalhos Relacionados selecionados

Trabalho Relacionado	Critério de Comparação									
	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10
Trabalho Proposto	●	●	●	●	●	●	●	●	●	●
[2]	●	○	○	○	○	●	○	○	○	○
[19]	●	●	○	●	○	●	○	●	○	●
[4]	◐	●	●	○	○	●	●	○	○	●
[20]	◐	○	●	●	○	●	○	○	●	●
[3]	●	○	●	○	○	●	◐	○	○	○
[26]	●	○	●	●	○	●	●	●	●	○
[7]	●	●	●	○	○	●	◐	○	●	○

Legenda: ● Atende Completamente ◐ Atende Parcialmente ○ Não Atende

Table 2: Questões de pesquisa investigados nesse artigo

Questão de pesquisa	Motivação	Variável
QP1: Baseado na sua experiência, quais são as possíveis causas de degradação arquitetural?	Revelar as causas da degradação baseado em opções pré definidas anteriormente.	Causas de degradação
QP2: Baseado na sua experiência, quais são as 5 causas mais críticas de degradação arquitetural?	Identificar quais causas afetam levam à degradação arquitetural de maneira mais consistente.	Causas mais críticas
QP3: As recomendações abaixo melhorariam significativamente os problemas de degradação arquitetural?	Apresenta algumas opções pré determinadas para reduzir os impactos da degradação com graus de escolha diferentes.	Recomendações de melhoria
QP4: Baseado na sua experiência, quais seriam as 5 ações mais efetivas para evitar degradação arquitetural?	Fornece algumas opções pré determinadas para buscar maneiras de evitar uma degradação arquitetural.	Recomendações para evitar
QP5: Você já "degradou" a arquitetura de um *software* de forma consciente?	Busca entender se os participantes já degradaram uma arquitetura de *software*.	Degradação consciente
QP6: Quais foram os motivos que lhe levaram a degradar a arquitetura de forma consciente?	Procura entender melhor os motivos que levaram os participantes a degradarem uma arquitetura de forma consciente.	Motivos da degradação consciente
QP7: Como arquitetura de *software* é representada na prática?	Visa entender como os participantes têm representado a arquitetura de *software*.	Representação da arquitetura
QP8: Como o desenvolvedor tem consciência da arquitetura do *software*?	Explora conhecer como os desenvolvedores conhecem a arquitetura de um sistema ou de uma funcionalidade até então desconhecida.	Conhecimento da arquitetura
QP9: Como a arquitetura é avaliada?	Entender como a arquitetura de software é avaliada na prática por profissionais.	Forma de avaliação
QP10: Baseado na sua experiência, qual seria a melhor forma de representar a arquitetura de um *software*?	Pretende identificar maneiras utilizadas pela indústria para representar uma arquitetura de *software*.	Representação da arquitetura

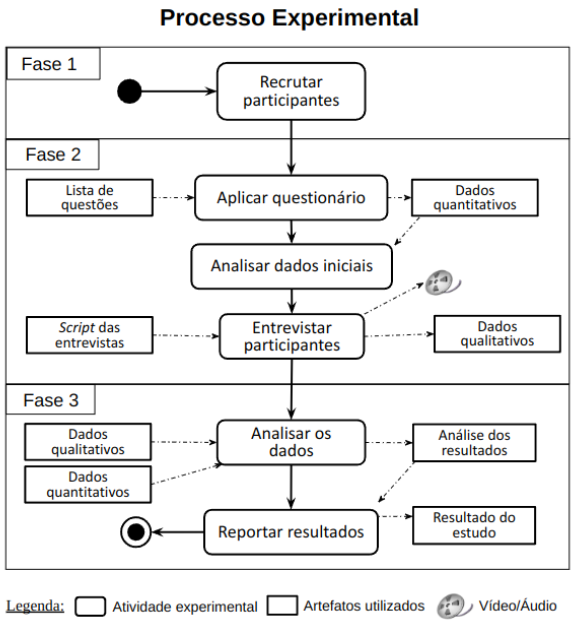

Figure 1: Processo experimental

4.3 Questionário e Entrevistas

Os dados foram coletados de entrevistas e de um questionário online[1] (criado no *Google Forms*). Os participantes refletiram sobre sua experiência na indústria de desenvolvimento de *software* através das nossas entrevistas semi-estruturadas [33]. A tabela **??** apresenta a lista de questões utilizadas nas entrevistas. Essas entrevistas buscaram gerar dados qualitativos. Novas perguntas foram formuladas a partir das respostas dos participantes. Um questionário online foi escolhido em função da sua facilidade de resposta para os participantes e a rápida forma de distribuição, alcançando assim um maior número de profissionais em localizações geográficas distintas sem custo adicional.

Para a realização das entrevistas, foi adotada a técnica de entrevista semiestruturada. Essa abordagem é amplamente utilizada na coleta de dados qualitativos, na qual o entrevistador possui uma lista de tópicos ou questões predefinidas, oferecendo uma estrutura básica para a conversa. No entanto, a técnica permite uma certa flexibilidade, possibilitando explorar ideias adicionais que possam surgir durante a entrevista. Isso permite uma abordagem mais aberta e aprofundada, capturando ideias e perspectivas únicas dos participantes [18].

A fim de promover um ambiente propício para a obtenção de informações valiosas, durante as entrevistas foi mantido um clima de

[1]Questionário: https://forms.gle/et2KdiRuVdpRf9NM8

abertura e empatia. Os participantes foram encorajados a compartilharem suas experiências e opiniões de maneira franca e genuína. Além de seguir os tópicos e questões pré-estabelecidos, os participantes ficaram livres para explorar informações adicionais que emergissem durante a conversa, permitindo que os entrevistados expressassem suas perspectivas de forma mais abrangente. Acredita-se que essa abordagem facilitou a obtenção de ideias mais profundos e enriquecedores, proporcionando uma compreensão mais completa dos temas abordados durante as entrevistas.

A escolha de não utilizar a entrevista não estruturada foi baseada na falta de estrutura dessa abordagem, o que poderia levar a respostas inconsistentes e dificultar a comparação entre os participantes. Por outro lado, a entrevista completamente estruturada não foi adotada, pois poderia restringir a profundidade das respostas e resultar em respostas padronizadas que não abordassem de maneira abrangente os tópicos de interesse. A opção pela entrevista semiestruturada foi fundamentada na consideração de que essa abordagem seria a mais adequada para garantir a consistência nas respostas, possibilitar uma exploração mais aprofundada e capturar a diversidade de perspectivas dos entrevistados [28].

5 RESULTADOS

Esta seção apresenta os resultados obtidos para as questões de pesquisa (descritas na Seção 4.1). Foram usados histogramas para prover uma visão geral dos dados coletados dos 347 participantes da pesquisa.

5.1 Análise de Perfil dos Participantes

A Tabela 4 resume o perfil dos participantes, reportando diversas informações como faixa etária, formação acadêmica, atividade profissional, experiência geral e experiência com arquitetura de *software*. Como algumas questões não eram obrigatórias, a soma das respostas não necessariamente se equivale ao número total de participantes (347).

Table 4: Os dados de perfil dos participantes.

Característica	Resposta	#	%
Faixa etária	18 - 20 anos	21	6,2%
	21 - 23 anos	117	34,5%
	24 - 26 anos	86	25,4%
	27 - 29 anos	54	15,9%
	30 - 32 anos	30	8,8%
	33 - 35 anos	16	4,7%
	36 - 38 anos	6	1,8%
	39 - 41 anos	6	1,8%
	42 - 44 anos	2	0,6%
	45 anos	1	0,3%
Graduação	Ciência da Computação	111	32,4%
	Análise de Sistemas	174	50,7%
	Sistemas de Informação	43	12,5%
	Outros	15	4,4%
Maior grau de escolaridade	Graduação	133	38,9%
	Técnico	71	20,7%
	Graduando	117	34,0%
	Mestrado	10	2,9%
	Outros	13	3,8%
Tempo estudado em universidade	< 2 anos	17	4,9%
	2-4 anos	132	38,3%
	5-6 anos	104	15.2%
	7-8 anos	33	9,6%
	> 8 anos	22	6,4%
Cargo/posição	Programador	192	56,3%
	Analista	67	19,6%
	Arquiteto	9	2,6%
	Gerente	5	1,5%
	Outros	68	19,9%
Quanto tempo nesse cargo/posição	< 2 anos	96	28,2%
	2-3 anos	30	8,8%
	3-4 anos	121	35,6%
	5-6 anos	51	15,0%
	7-8 anos	17	5,0%
	> 8 anos	25	7,36%
Experiência com desenvolvimento de *software*	< 2 anos	76	22,2%
	2-3 anos	34	9,9%
	3-4 anos	120	35,0%
	5-6 anos	56	16,3%
	7-8 anos	24	7,0%
	> 8 anos	33	9,6%
Experiência com arquitetura de *software*	< 2 anos	180	52,9%
	2-3 anos	54	15,9%
	3-4 anos	67	19,7%
	5-6 anos	19	5,6%
	7-8 anos	7	2,1%
	> 8 anos	13	3,8%

Faixa etária. A maioria (63,5%) dos participantes estão na faixa de 21 a 26 anos, representando um público relativamente jovem. Essa informação é importante de ser observada, porém é necessário também analisar a experiência desses participantes, pois muitos possuem anos de experiência, sendo possível concluir que iniciaram a sua carreira profissional ainda bastante jovens. Além disso, 18% dos participantes reportaram ter mais de 30 anos, trazendo assim uma variedade muito interessante para a pesquisa. Essa variedade se dá uma vez que esse público presenciou diversas fases do desenvolvimento de *software*, desde uma programação mais clássica até a modernidade de hoje com o avanço rápido da tecnologia.

Graduação. A maioria (63,2%) dos participantes cursam ou cursaram tecnólogos na área de TI, sendo 50,7% do público em análise de sistemas e 12,5% em sistemas de informação. Levando em conta a proposta dos cursos ser voltado para o mercado de trabalho, os participantes agregam muito conhecimento prático para a pesquisa, contribuindo muito com o objetivo de procurar entender a realidade do mercado de trabalho no Brasil. Além disso, 32,4% informaram estar ou ter cursado Ciência da Computação, trazendo também uma visão um pouco mais acadêmica do assunto, uma vez que esse curso é de bacharelado. Esse contraste é muito bem vindo pois traz diferentes percepções para as questões estudadas e contribuem muito para a qualidade dos resultados encontrados.

Maior grau de escolaridade. A maioria (59,6%) dos participantes já haviam finalizado o seu curso na época da pesquisa, sendo 38,9% formados em ensino superior e 20,7% formados em estudo técnico. Além disso, 34,0% informaram ainda estarem estudando no ensino superior. Esses resultados condizem com a faixa etária dos participantes, que revelou a maioria ser um público mais jovem. Também é importante ressaltar que 10 participantes haviam concluído o mestrado, contribuindo assim novamente para diferentes perspectivas do assunto e trazendo um espectro mais amplo para as questões estudadas.

Tempo estudado em universidade. A tabela apresenta resultados diversos, sendo a maior concentração de 2 a 4 anos de estudo, totalizando 38,3% dos resultados. Esse resultado condiz com os resultados anteriores, sendo o público em sua maioria mais jovem e também ter cursado cursos tecnólogos, que tem uma duração menor que bacharelados. Além disso, é possível observar que 31,2% dos participantes estão ou estudaram na universidade por mais de 5 anos, tendo 22 participantes com mais de 8 anos de estudo em

universidade. Esse valor não necessariamente se traduz em experiência pois alguns participantes podem ter focado na carreira em paralelo ao estudo e isso acaba atrasando a graduação em virtude de agregar mais experiência profissional.

Cargo/posição. Os resultados apresentados indicam que 80% dos participantes estão ligados à atividade de desenvolvimento de *software*, tendo os demais indicado profissões não diretamente ligadas a esse fim. Dos que estão ligados à área, a maioria ocupava cargo de programador. Esse resultado é importante pois o trabalho do programador é um elo crucial para a manutenção da arquitetura, a causa e a prevenção da degradação arquitetural. Além disso, 67 participantes atuavam como analistas de *software* e podem trazer *insights* importantes sobre efeitos da degradação arquitetural fora do código.

Quanto tempo nesse cargo/posição. Os participantes informaram ter uma experiência bastante diversificada com a área, sendo a faixa mais escolhida de 3 a 4 anos, sendo apontada por 35,6% dos participantes. Essa faixa comumente é conhecida no mercado como profissionais plenos, o que indica um bom nível de conhecimento e experiência para assumir cargos na área de desenvolvimento de *software*. Além disso, 27,4% dos participantes informaram ter mais de 5 anos de experiência no cargo/posição. Essa informação é muito útil pois nesse caso já é possível encontrar perfis mais experientes que podem trazer mais valor ainda às respostas encontradas.

Experiência com desenvolvimento de *software*. Os dados coletados nessa questão condizem com a questão anterior sobre tempo no cargo/posição. Essa observação pode indicar pouco movimento de troca de cargos pelos participantes dentro da área de desenvolvimento de *software*. Essa informação contribui para a análise de senioridade dos participantes feita na questão anterior, uma vez que aponta que os participantes tiveram a maior parte da experiência na área diretamente no cargo informado.

Experiência com arquitetura de *software*. Apesar dos participantes terem indicado uma boa experiência com desenvolvimento de *software*, 52,9% dos participantes informaram ter menos de 2 anos de experiência com arquitetura de fato. Essa informação é importante pois demonstra uma falta de experiência dos participantes com a área. Também indica que o mercado de trabalho que esses participantes atuam não tratam a arquitetura como algo intrínseco ao desenvolvimento de *software*, criando um distanciamento entre eles. Apesar disso, 35,6% dos participantes disseram ter até 4 anos de experiência com o assunto e 11,5% informaram ter mais de 5 anos de experiência.

Perfil dos participantes entrevistados. 5 participantes foram entrevistados para os resultados qualitativos. A faixa etária dos participantes foi de 19 a 25 anos e todos possuem ou estão envolvidos em ensino superior na área de TI. Desses, 60% cursam ou cursaram Ciência da Computação, 20% Sistemas para Internet e os demais 20% Análise de Sistemas. Além disso, todos possuem experiência profissional na área de desenvolvimento de *software*, variando suas experiências de 3 a 6 anos. Esses participantes serão referenciados no trabalho como participantes "A", "B", "C", "D", e "E".

5.2 QP1: Possíveis causas de degradação arquitetural

A Figura 2 apresenta os dados coletados referentes à variável causas de degradação (QP1). Cada participante indicou o grau de concordância com as afirmações propostas, de acordo com a escala Likert. Em geral, os resultados indicam que os participantes concordaram (de 45% a 92%) com as afirmações feitas, produzindo resultados homogêneos. A falta de conhecimento da arquitetura foi apontada como a principal causa de degradação, tendo 92,8% de concordância entre os participantes. Além disso, a falta de padronização, com 55,5% de concordância completa, destacou-se como a segunda causa mais severa de degradação. Por outro lado, os participantes indicaram que o desconhecimento da filosofia/cultura da empresa, limitações das tecnologias utilizadas e políticas de redução de custos da empresa são fatores menos relevantes para a degradação arquitetural.

Os resultados corroboram com estudos empíricos anteriormente reportados na literatura. [25] aponta que apesar da documentação ser um artefato muito valioso para conhecer a arquitetura do software muitas vezes ela não está disponível ou atualizada. Já [2] aponta que, entre outras causas, a falta de conhecimento dos desenvolvedores levam a degradação arquitetural. Isso indica que a indústria brasileira de desenvolvimento de *software* enfrenta problemas similares da indústria global, indicando assim uma sincronia entre tecnologias, práticas e eventuais problemas entre os dois mundos.

Além disso, os resultados foram aprofundados nas entrevistas qualitativas, onde foi possível estudar mais sobre as possíveis causas da degradação arquitetural. Fatores ligados a competência técnica foram mencionados por 80% dos participantes. O participante "B" reportou que "*a falta de conhecimento da arquitetura na hora de dar manutenção é um fator predominante na geração de degradação arquitetural*", enquanto o participante "D" reportou que "*a falta de experiência dos profissionais na área contribui diretamente para o problema, tendo sua origem no ingresso muito rápido desse profissional na área e a evolução rápida das tecnologias que não é acompanhado num ritmo condizente pelos profissionais que já atuam há alguns anos*". Os demais participantes que também pontuaram a competência técnica como uma causa raiz da degradação arquitetural trouxeram pontos similares aos mencionados acima.

Conclusão da QP1: A QP1 investigou 23 causas de degradação arquitetural de forma empírica. Os principais resultados observados foram que 92,8% dos participantes concordaram que a principal causa da degradação arquitetural é a falta de conhecimento e 55,5% a falta de padronização. Isso implica que há uma necessidade de inclusão de assuntos relacionados à arquitetura de *software* no processo de formação dos profissionais envolvidos na área. A QP1 também auxilia gestores a escalarem suas equipes de desenvolvimento, uma vez que demonstra a necessidade de ter profissionais ligados à áreas de arquitetura de *software* além dos demais profissionais necessários.

Figure 2: Possíveis causas de degradação arquitetural (QP1)

5.3 QP2: As 5 causas mais críticas de degradação arquitetural

A Figura **??** apresenta os dados coletados referentes à variável causas mais críticas (QP2). Essa questão era objetiva e foram coletadas 346 respostas que foram ordenadas de maneira decrescente. De maneira geral, é possível observar que a falta de conhecimento da arquitetura e a falta de documentação foram escolhidos pela maioria (mais de 50%) dos participantes. A Figura **??** também facilita a visualização das causas menos críticas (menos de 20% dos votos), que incluem diversos fatores que, apesar de serem importantes, não são os principais complicadores para a degradação arquitetural.

Os resultados corroboram os resultados encontrados na QP1 e com a literatura já existente. [19] aponta 13 causas para a degradação arquitetural que possuem relação com as causas estudadas pela QP2. Apesar disso, o estudo não ordena os resultados apresentados, dificultando assim a tarefa de priorização para resolver o problema. Essa priorização é útil pois norteia gestores e demais profissionais ligados ao desenvolvimento de *software* na atuação da resolução do problema de degradação arquitetural.

Além disso, os resultados foram aprofundados nas entrevistas qualitativas, onde foi possível observar uma sintonia entre ambas entrevistas. Novamente fatores ligados à falta de conhecimento e documentação foram apontados pelos participantes. O participante "E" reportou que "*por notar uma inexperiência de alguns profissionais na área, as vezes eles não tem a real dimensão do problema e acabam desenvolvendo sem entender muito bem a arquitetura e os requisitos do sistema*". Esse ponto foi também mencionado pelo participante "D", que reportou que "*alguns desenvolvedores não têm senso de responsabilidade*".

Conclusão da QP2: A QP2 procurou ordenar as causas mais críticas para a degradação arquitetural. Com isso, é possível entender de maneira mais pontual os problemas encontrados na QP1, uma vez que os resultados da QP2 corroboram, apontando mais uma vez problemas ligados à competências técnicas e padronização foram encontradas. Essa informação auxilia gestores na priorização dos problemas para atuar de maneira mais produtiva e buscar mitigar a degradação arquitetural em seus projetos.

5.4 QP3: As recomendações abaixo melhorariam significativamente os problemas de degradação arquitetural?

A Figura **??** apresenta os dados coletados referentes à variável recomendações de melhoria (QP3). Cada participante indicou o grau de concordância com as afirmações propostas, de acordo com a escala Likert. Em geral, os resultados indicam que os participantes concordaram (de 49,1% a 98,8%) com as afirmações feitas, produzindo assim resultados homogêneos. A documentação clara, objetiva e detalhada foi apontada como a recomendação principal para melhorar os problemas de degradação arquitetural, tendo 98,8% de concordância entre os participantes. Por outro lado, a metodologia de desenvolvimento comum em todos os projetos da empresa, a gerência de dependência, o gerenciamento de dependência entre os módulos da arquitetura e adequação da arquitetura às reais necessidades da empresa foram as menos aceitas pelos participantes, com porcentual de discordância igual ou maior a 10%.

Esse resultado condiz com o estudo de [25], que explicou que um artefato de documentação eficaz e atualizado ajudaria muito em evitar uma degradação arquitetural, mas infelizmente muitas vezes ele não está disponível ou então não está atualizado. Outro ponto que foi amplamente (84,9%) concordado pelos participantes foi a refatoração de código. Isso corrobora com o estudo realizado por [11] que explica que refatorações de código existem para melhorar estruturas de código pois algumas vezes a qualidade da solução não atende ao nível esperado ou então viola regras propostas pela arquitetura empregada. Por fim, também é possível visualizar que fatores externos ao código influenciam diretamente na qualidade do trabalho e também na degradação arquitetural. Esses fatores incluem a gestão do projeto, a análise de requisitos, o treinamento para novos desenvolvedores e o empenho da equipe em geral.

Além disso, foi possível abordar mais a fundo a questão com as entrevistas qualitativas, onde pontos importantes foram mencionados e acrescentam aos resultados encontrados. O participante "A" reportou que "*é necessário ter um bom tech lead que entenda bem do negócio e possa conduzir e incentivar o time a manter e melhorar a qualidade do código produzido*". Também reportou que "*é necessário a gerência dar liberdade para o time todo opinar, já que são eles que estão em contato com o produto diariamente e sabem dos possíveis problemas arquiteturais existentes*". Por fim, comentou que "*é necessário ter espaço para focar nas melhorias de código e arquiteturais necessárias junto com o desenvolvimento de novas funcionalidades*". Já o participante "B" reportou que algumas práticas como refatorações e *code reviews* são técnicas eficazes para combater a degradação arquitetural. Por fim, o participante "E" focou na necessidade de entender o problema antes de focar na solução, assim como também não subestimá-lo, sendo necessário dedicar um tempo suficiente para o refinamento da solução antes do desenvolvimento.

Conclusão da QP3: A QP3 investigou 15 recomendações de forma empírica, sendo os principais resultados encontrados a necessidade de ter uma documentação adequada a recomendação principal para melhorar os problemas de degradação arquitetural. Além disso, a refatoração de código também foi amplamente aceita pelos participantes, tendo 84,9% concordado com a opção. Esses resultados beneficiam gestores e desenvolvedores pois apontam medidas específicas a serem observadas e trabalhadas afim de atuar na problemática de degradação arquitetural.

5.5 QP4: Baseado na sua experiência, quais seriam as 5 ações mais efetivas para evitar degradação arquitetural?

A Figura 5 apresenta os dados coletados referentes à variável recomendações para evitar (QP4). Essa questão era objetiva e foram coletadas 342 respostas que foram ordenadas de maneira decrescente. De modo geral, é possível ver porcentagens bastante equilibradas entre as ações propostas. As 5 principais ações corroboram com a linha de raciocínio apresentada pelos participantes ao longo da pesquisa, corroborando com as questões de pesquisa anteriores. Além disso, os resultados expandem a literatura existente. [19] explica que as ferramentas para prevenir a degradação arquitetural se dão em análise e avaliação da arquitetura principalmente. Já a QP4 expande essa análise trazendo artefatos mais pontuais para auxiliar na tarefa, como é possível observar na Figura 5. Essas ações incluem tanto o trabalho prático no código como refatorações e uso de padrões, como fatores externos ligados à gestão da equipe, análise de requisitos e documentação. Também é possível notar alguns fatores humanos como o empenho da equipe, que foi escolhido dentro as 10 principais ações, tendo sido votado por 31% dos participantes.

Também foi possível triangular os resultados encontrados com as entrevistas qualitativas, onde os participantes opinaram sobre pontos semelhantes aos encontrados nas entrevistas quantitativas. O participante "C" apontou que "*é necessário manter uma documentação atualizada para poder evitar a degradação arquitetural*".

Ao mesmo tempo, também explicou que "*é mais simples quando há um time menor de desenvolvimento, porém mesmo assim é difícil de manter a documentação atualizada*". Por fim, reportou que "*nos projetos que participei não era normal essa tarefa de atualização da documentação, apesar de acreditar ser importante*". Já o participante "D" reportou que "*é necessário o time estar engajado e decidido a manter a qualidade da arquitetura do software*". Para isso, explicou que "*é necessário algum puxar a frente e que o time técnico precisa "vender" a necessidade de ter uma documentação para a gerência*".

Conclusão da QP4: A QP4 investigou 16 ações mais efetivas para evitar a degradação arquitetural. Os resultados encontrados foram bastante equilibrados, tendo novamente questões relacionadas documentação e padronização ocupando as 5 primeiras ações. Além disso, a QP4 traz uma ordenação das ações estudadas. Isso beneficia e auxilia gestores e desenvolvedores para identificar práticas e ações mais efetivas para evitar a degradação arquitetural e poder montar uma força de trabalho adequada para atuar na questão.

5.6 QP5: Você já "degradou" a arquitetura de um software de forma consciente?

A Figura **??** apresenta os dados coletados referentes à variável degradação consciente (QP5). Essa questão era objetiva e foram coletadas 344 respostas. De maneira geral, a Figura **??** apresenta de maneira bastante objetiva os resultados obtidos, tendo a maioria dos participantes negado ter degradado a arquitetura de maneira consciente. Por outro lado, é importante entender que 35,5% informaram já ter praticado a degradação arquitetural de maneira consciente.

Esse resultado expande a literatura já existente. [1] estudou o comportamento de *code smells* e degradação arquitetural em projetos *open source*. O trabalho revelou que a inserção de degradação arquitetural é algo frequente e que muitos *commits* têm a finalidade de remover a degradação arquitetural existente. Também observou uma divisão entre membros principais do projeto e demais colaboradores, onde os colaboradores tendem a gerar mais degradação arquitetural e os membros principais mais correções para essas degradações do que o inverso. Apesar disso, não cita se os colaboradores degradam a arquitetura de forma intencional ou não intencional.

Além disso, as entrevistas qualitativas apresentaram resultados condizentes com os apresentados acima. Alguns participantes afirmaram já terem feito um desenvolvimento que acabou gerando degradação arquitetural, mesmo estando conscientes disso. Para esses casos, os participantes informaram que era comum criar um item de débito técnico no *backlog* do projeto para poder refatorar o código futuramente. O participante "D" reportou que "*é necessário ter uma conversa franca do que foi feito e analisar um histórico para entender o que aconteceu*".

Conclusão da QP5: A QP5 procura entender precisamente se os participantes já cometerem uma degradação arquitetural de forma consciente, buscando entender mais a fundo a origem do problema e como ele evolui ao longo do tempo. A maioria dos participantes negou ter degradado a arquitetura de maneira consciente, porém 35,5% informaram já terem realizado a prática. Essa informação é muito útil para gestores entenderem um pouco mais das práticas realizadas dentro dos times de desenvolvimento e poder realizar um trabalho preventivo com seus times para evitar que esse cenário aconteça.

5.7 QP6: Quais foram os motivos que lhe levaram a degradar a arquitetura de forma consciente?

A Figura 7 apresenta os dados coletados referentes à variável motivos da degradação consciente (QP6). Essa questão era discursiva e foram obtidas 138 respostas, que foram categorizadas em 9 categorias utilizando a técnica de *Ground Theory* [13]. De maneira geral, essa questão responde à questão de pesquisa anterior, que apontou que 35,5% dos participantes já degradaram a arquitetura de maneira consciente. É possível observar que o prazo foi apontado como o motivo principal (51%) e também teve a diferença de porcentagem mais expressiva dos outros motivos.

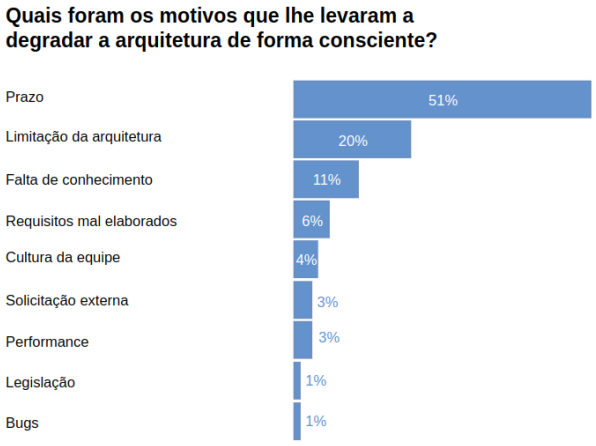

Figure 7: Motivos da degradação consciente (QP6)

Esse resultado condiz com o estudo de [7], que aponta que prazos apertados contribuem diretamente para débitos técnicos. Além disso, podem provocar uma pressão no time de desenvolvimento e essa pode levar à degradação arquitetural. Também é possível observar que 31% dos participantes falaram sobre limitações tanto da arquitetura quanto dos próprios desenvolvedores pela falta de conhecimento impactam diretamente a questão. Essa informação corrobora e amplia o estudo de [30], que buscou entender a diferença entre arquitetura de *software* e o código fonte na prática. Foi observado que por vezes há uma dificuldade dos desenvolvedores em implementar a arquitetura desejada. A QP6 estende essa análise,

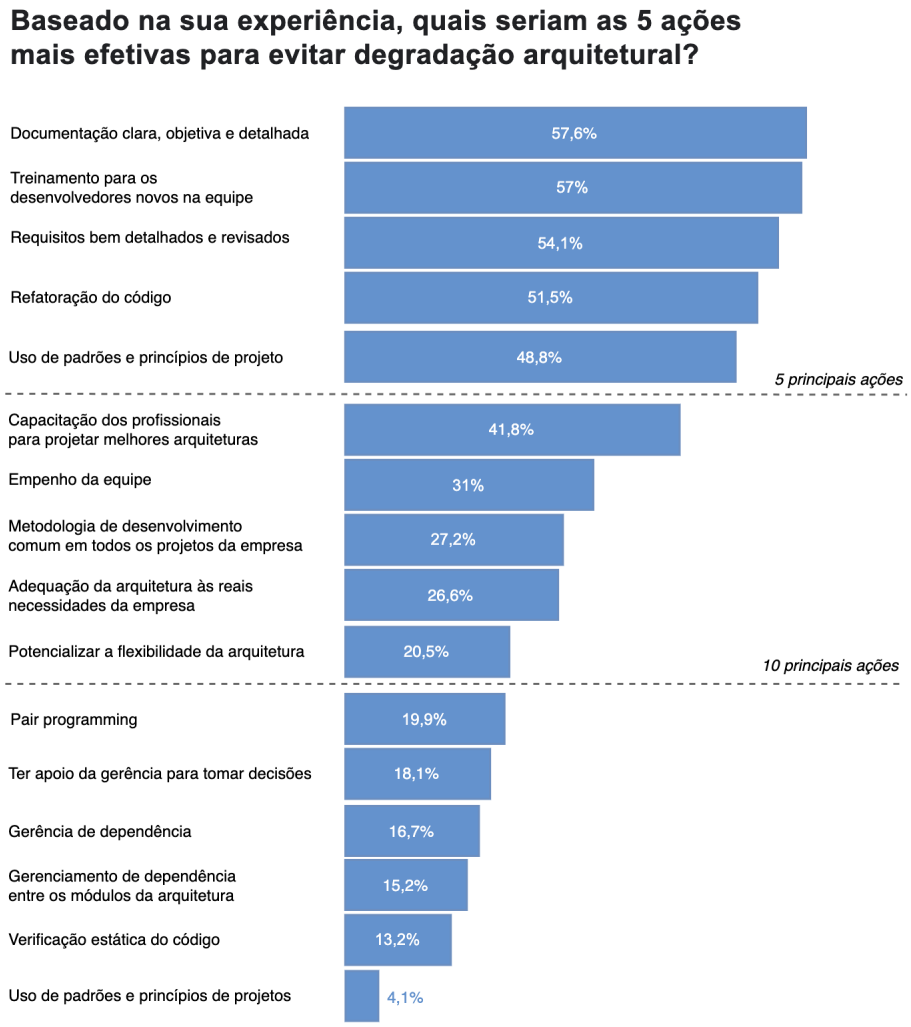

Figure 5: Recomendações para evitar (QP4)

uma vez que a limitação pode também ser em função da própria arquitetura não suportar a funcionalidade desejada, além da limitação dos desenvolvedores.

Além disso, as entrevistas qualitativas observaram alguns outros pontos. O participante "A" reportou que "*as vezes fico desmotivado em fazer a melhor implementação quando parece que a própria gerência não está interessada na qualidade do código e da arquitetura implementada*". Esse depoimento ligado a fatores humanos foi pontuado por todos os participantes, onde questões como motivação, reconhecimento e disciplina foi apontado como sendo necessários para a manutenção da saúde do projeto. Inclusive, um participante com mais de 5 anos de experiente na área relatou que "*hoje o desenvolvedor precisa desenvolver muito bem as suas soft skills, uma vez que o esteriótipo do desenvolvedor quieto programando o que lhe é dado não é suficiente. É necessário um perfil mais comunicativo e colaborativo, que incentive os outros a evoluírem e procurar evoluir também*".

Conclusão da QP6: A QP6 investigou os motivos que levaram os participantes a terem degradado a arquitetura de software de maneira intencional. Os principais pontos relatados pelos participantes foram o prazo, a limitação da arquitetura e a falta de conhecimento. Essa informação auxilia gestores e desenvolvedores a identificar certas tendências, práticas e perigos relacionados ao desenvolvimento de *software*, os possibilitando assim trabalhar em práticas que evitem esses cenários e por consequência a degradação arquitetural.

5.8 QP7: Como arquitetura de *software* é representada na prática?

A Figura 8 apresenta os dados coletados referentes à variável representação da arquitetura (QP7). Essa questão era discursiva e foram obtidas 233 respostas, que foram categorizadas em 7 categorias utilizando a técnica de *Ground Theory* [13]. Alguns participantes apontaram mais de um fator em suas respostas e em função disso

a porcentagem obtida na soma das categorias ultrapassa 100%. O ponto principal a ser observado é que 46% dos participantes disseram não representar a arquitetura na prática.

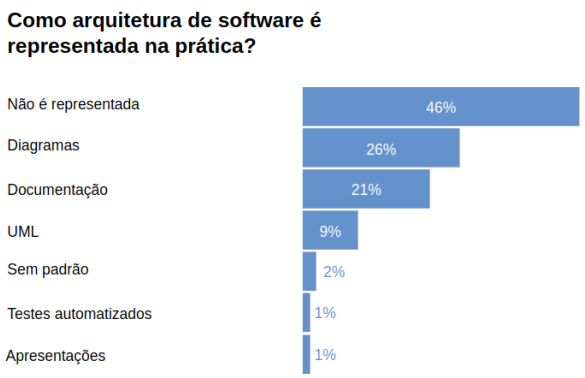

Figure 8: Representação da arquitetura (QP7)

Esse resultado condiz com o estudo de [22], que mostra que um dos cinco principais débitos técnicos levantados é a falta de documentação nos projetos de desenvolvimento de *software*. Outro fator possível de ser observado é que apesar de muitos participantes terem apontado diagramas como o item principal para representar a arquitetura, apenas 9% apontaram a UML como o modelo a ser utilizado. Esse dado condiz com o estudo de [17], que mostrou que 74% dos participantes não utilizam a UML numa aplicação prática de um projeto real na indústria brasileira.

As entrevistas qualitativas corroboram com os resultados apresentados acima. Todos os participantes revelaram que normalmente essa arquitetura não é representada e quando é, não segue padrões específicos como a UML. O participante "D" revelou que "*a liberdade de ferramentas como o Miro*[2] *é boa pois a UML é muito limitada e traz dificuldades para profissionais que não têm um mundo acadêmico, porém quando é muito livre pode ficar bagunçado e é necessário um tempo considerável para reorganizar*". Já o participante "E" apontou que "*diagramas mais gerais descrevendo funcionamento, até diagramas mais específicos falando das camadas e o que cada é responsável são ferramentas excelentes para descrever a arquitetura do software*".

Conclusão da QP7: A QP7 buscou entender como os participantes têm representado a arquitetura de *software* na prática. Foi observado que muitos participantes não a representam. Além disso, apesar de 26% utilizarem diagramas, apenas 9% citaram utilizar padrões já existentes como a UML. Esses resultados são muito úteis para desenvolvedores entenderem que essa prática acaba sendo comum e não um possível problema isolado, os permitindo assim falar mais sobre o assunto e procurar resolver esse problema em seus times.

[2]Miro: https://miro.com/

5.9 QP8: Como o desenvolvedor tem consciência da arquitetura do *software*?

A Figura 9 apresenta os dados coletados referentes à variável conhecimento da arquitetura (QP8). Essa questão era discursiva e foram obtidas 246 respostas, que foram categorizadas em 7 categorias utilizando a técnica de *Ground Theory* [13]. Alguns participantes apontaram mais de um fator em suas respostas e em função disso a porcentagem obtida na soma das categorias ultrapassa 100%. Os resultados foram bastante equilibrados, porém com práticas muito opostas liderando os resultados. Apesar de 34% dos participantes apontarem para a prática de entender a documentação, 32% apontaram ir diretamente ao código e empiricamente tentar entender a arquitetura. Isso demonstra que ou a documentação não é suficiente ou inexistente para conseguir entender a arquitetura do projeto.

Esses resultados condizem com o estudo de [25], que revela que a documentação é um artefato muito valioso para entender a arquitetura de um *software*. Também revela que apesar de ser muito valioso, muitas vezes esse artefato não está disponível. Essa informação também condiz com o estudo de [22], que explora que a falta de documentação aumenta a carga de trabalho do desenvolvedor para ter que entender o código de maneira empírica e pode levar à degradação arquitetural.

As entrevistas qualitativas apresentaram respostas condizentes com os resultados apresentados acima. Alguns participantes deram uma resposta um pouco mais ampla, como foi o caso do participante "E", que dividiu sua resposta em uma análise para projetos novos e projetos em andamento. Para projetos novos apontou que "*é necessário entender bem o problema, conversar com outros profissionais experientes para ajudar na modelagem da arquitetura e estudar produtos similares no mercado*". Já para projetos em andamento, apontou que "*é necessário conversar com os membros que trabalharam no projeto e novamente entender bem o problema a ser solucionado*". Quando questionado se tem a prática de procurar por documentação disse que não pois em sua experiência não é algo comum de existir. Apesar disso, os demais participantes fizeram menção à documentação do *software* como fonte primária de consulta.

Conclusão da QP8: A QP8 procurou entender e categorizar as formas que os desenvolvedores têm consciência da arquitetura do *software*. É possível observar que apesar da documentação ser algo muito útil e apontado por 34% dos participantes, 32% informaram que foi necessário um trabalho mais manual diretamente no código em função da falta dessa documentação atualizada. Essa informação é relevante para os desenvolvedores que estão trabalhando em algum projeto em andamento perceberem a necessidade de ter essa representação para o caso de futuras manutenções ou mudanças no time.

5.10 QP9: Como a arquitetura é avaliada?

A Figura 10 apresenta os dados coletados referentes à variável avaliação da arquitetura (QP9). Essa questão era discursiva e foram obtidas 320 respostas, que foram categorizadas em 12 categorias utilizando a técnica de *Ground Theory* [13]. Alguns participantes apontaram mais de um fator em suas respostas e em função disso a porcentagem obtida na soma das categorias ultrapassa 100%. De maneira geral, os participantes concordaram em maioria com as

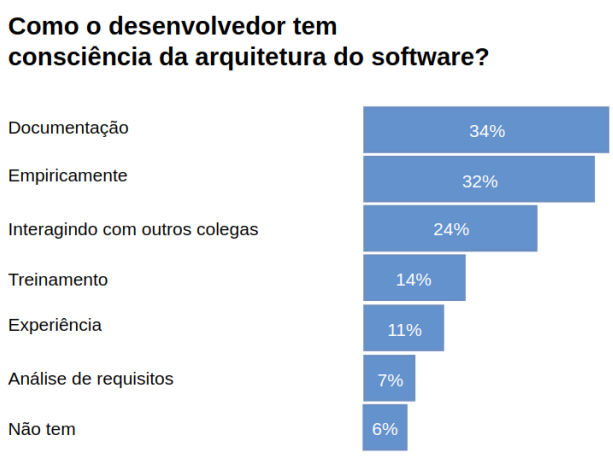

Figure 9: Conhecimento da arquitetura (QP8)

duas opções já pré-existentes para a pergunta: opinião de experts e métricas.

Esse resultado encontrado condiz com os resultados encontrados no estudo de [3], que aponta que é possível identificar se houve erosão arquitetural baseado em diversas métricas, com nove classificações principais. Além dos resultados mais escolhidos também é possível notar alguns extremos, como a informação que a arquitetura não é avaliada, tendo sido apontada por 2% dos participantes e também a avaliação se dar por "instinto", sem ter um padrão oficial. Esse fator foi apontado por 0,3% dos participantes.

Além disso, as entrevistas qualitativas corroboram com os resultados observados. O participante "A" citou que "*é necessário a presença de tech leads para auxiliar o time no desenvolvimento e na manutenção da arquitetura, assim como incentivar os demais membros a manter a qualidade do código e o estudo sobre arquitetura em dia*". Também mencionou que apesar de ter essa figura de liderança, é muito importante que os membros do time procurem entender bem a arquitetura e estarem atualizados quanto as tecnologias e práticas do mercado. Da mesma forma analisou o papel do arquitetura de *software*, uma vez que sua presença é importante em momentos específicos do projeto, mas é necessário o time ter a competência técnica suficiente para trabalhar nos momentos que sua presença não é necessitada.

Conclusão da QP9: A QP9 procurou entender como a arquitetura de *software* é avaliada pelos participantes. Os resultados encontrados revela que os participantes concordaram com as opções já pré-existentes e acrescentaram outras opções de acordo com sua experiência. A avaliação baseada em opinião de *experts* e métricas foram as mais escolhidas pelos participantes. Isso é útil para os gestores identificarem a necessidade de ter essa figura *expert* nos times, afim de construir as métricas necessárias e fazer a avaliação da arquitetura de maneira recorrente.

5.11 QP10: Baseado na sua experiência, qual seria a melhor forma de representar a arquitetura de um *software*?

A Figura 11 apresenta os dados coletados referentes à variável Representação da arquitetura (QP10). Essa questão era discursiva e foram obtidas 235 respostas, que foram categorizadas em 7 categorias utilizando a técnica de *Ground Theory* [13]. Alguns participantes apontaram mais de um fator em suas respostas e em função disso a porcentagem obtida na soma das categorias ultrapassa 100%. De maneira geral, é possível observar que diagramas e documentação foram apontados pela maioria dos candidatos. Por outro lado, apesar de 66% dos participantes terem apontado o uso de diagramas como uma forma para representar, apenas 17% fizeram a menção para a UML especificamente.

Este resultado condiz com o estudo realizado por [17], que mostra que a maior parte dos participantes da pesquisa na indústria brasileira não utiliza a UML para representar a arquitetura em projetos reais. Também é possível observar que 1,2% dos participantes apontaram ideias de inovação para novas ferramentas na IDE e de monitoramento, com auxílio de inteligência artificial poderiam ser recursos muito úteis para representar a arquitetura de um *software* e que poderiam ser objetos de estudo para futuras pesquisas.

Da mesma maneira, os participantes das entrevistas qualitativas indicaram a documentação e diagramas como fontes principais de representação da arquitetura. O participante "B" reportou também não utiliza UML, porem acrescentou que "*apesar de não utilizar uma metodologia específica, é necessário um desenvolvimento consciente, isto é, fazer bem feito e de maneira consistente*". Já o participante "E" relatou que "*diagramas são uma ótima ferramenta para representar a arquitetura, mesmo sem utilizar uma metodologia específica, como a UML*".

Conclusão da QP10: A QP10 buscou entender qual seria a melhor forma de representar a arquitetura de um *software* na opinião dos participantes. O principal resultado encontrado foi a utilização de diagramas e também a não utilização da UML especificamente. Esse resultado é útil para pesquisadores desenvolverem técnicas mais alinhadas com as necessidades do mercado para documentar a arquitetura de uma maneira mais dinâmica, mas que ainda seja possível padronizar e garantir a consistência da documentação.

6 CONCLUSÃO E TRABALHOS FUTUROS

O trabalho apresentou um *survey* exploratório em como os participantes lidam com a degradação arquitetural na indústria de desenvolvimento de *software* brasileira. Ao todo, 347 participantes de diversas empresas participaram de um questionário online onde fatores ligados às causas, consequências e recomendações para evitar a degradação arquitetural foram abordados. Além disso, também foram conduzidas 5 entrevistas seguindo o protocolo de entrevistas semi-estruturadas com profissionais da área de empresas diferentes para investigar e triangular os resultados encontrados no questionário quantitativo. Em suma, os resultados apontam que a falta de documentação, padronização e competência técnica são fatores fundamentais na geração de degradação arquitetural. Concomitantemente, fatores ligados a pressões externas, como prazos ou limitações de sistemas de terceiros também foram apontados como

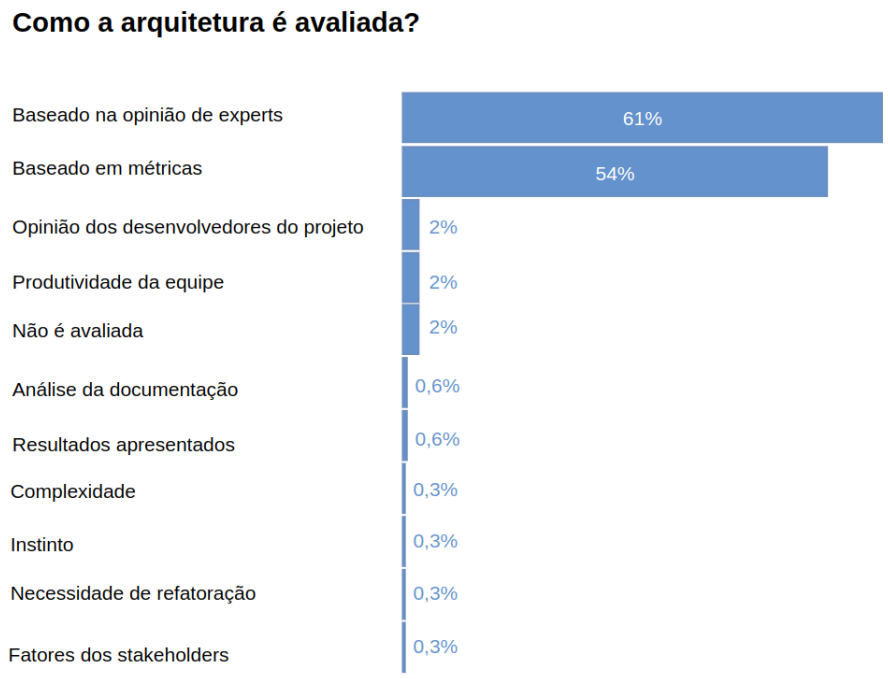

Figure 10: Avaliação da arquitetura (QP9)

Figure 11: Representação da arquitetura (QP10)

fatores contribuintes para a geração da degradação. Além disso o estudo também revelou que alguns fatores humanos têm implicação direta da degradação arquitetural. Esses fatores envolvem motivação, disciplina e estratégias de acompanhamento de atualizações do mercado de trabalho.

Os resultados encontrados nessa pesquisa corroboraram com a literatura já existente relacionada a degradação arquitetural [2, 25]. Em geral, muitos desenvolvedores não costumam estudar arquitetura de *software*, apesar de terem anos de experiência com desenvolvimento em geral. Essa falta de conhecimento técnico em arquitetura é um dos pilares que levam à degradação arquitetural. A falta de documentação atualizada dificulta bastante a tarefa de garantir a consistência da arquitetura prescritiva com a implementação real do sistema. Caso os times de desenvolvimento incluíssem mais práticas relacionados à arquitetura os problemas de degradação

arquitetural seriam amplamente reduzidos. Por último, a pesquisa procurou mostrar pontos mais práticos a serem observados pelos profissionais em seus times para mitigar a degradação arquitetural de maneira eficaz através de ações e estudo de causas ordenadas pelas respostas dos participantes. Além disso, o trabalho incentiva a pesquisa de novas metodologias de documentação de *software* que sejam mais dinâmicas que as existentes, uma vez que os participantes reportaram não fazer o uso de técnicas já estabelecidas, como a UML, apesar de reconhecerem a importância da documentação e de diagramas para representar a arquitetura de *software*.

REFERENCES

[1] Iftekhar Ahmed, Umme Ayda Mannan, Rahul Gopinath, and Carlos Jensen. 2015. An empirical study of design degradation: How software projects get worse over time. In *2015 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 1–10.

[2] Ahmed Baabad, Hazura Binti Zulzalil, Salmi Binti Baharom, et al. 2020. Software architecture degradation in open source software: A systematic literature review. *IEEE Access* 8 (2020), 173681–173709.

[3] Ahmed Baabad, Hazura Binti Zulzalil, Salmi Binti Baharom, et al. 2022. Characterizing the architectural erosion metrics: A systematic mapping study. *IEEE Access* (2022).

[4] David Benjaminsson and Edvin Bengtsson. 2021. Observing software architectural gaps in industry: A case study. (2021).

[5] Hebert Cabane and Kleinner Farias. 2024. On the impact of event-driven architecture on performance: An exploratory study. *Future Generation Computer Systems* 153 (2024), 52–69.

[6] Carlos Carbonera, Kleinner Farias, CéSar Graeff, and Robson Silva. 2023. Software Merge: A Two-Decade Systematic Mapping Study. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*. 99–108.

[7] Dipta Das, Abdullah Al Maruf, Rofiqul Islam, Noah Lambaria, Samuel Kim, Amr S Abdelfattah, Tomas Cerny, Karel Frajtak, Miroslav Bures, and Pavel Tisnovsky. 2022. Technical debt resulting from architectural degradation and code smells: a systematic mapping study. *ACM SIGAPP Applied Computing Review* 21, 4 (2022), 20–36.

[8] Carlos Eduardo Carbonera, Kleinner Farias, and Vinicius Bischoff. 2020. Software development effort estimation: A systematic mapping study. *IET Software* 14, 4 (2020), 328–344.

[9] Kleinner Farias, Alessandro Garcia, and Carlos Lucena. 2014. Effects of stability on model composition effort: an exploratory study. *Software & Systems Modeling* 13 (2014), 1473–1494.

[10] Kleinner Farias, Alessandro Garcia, Jon Whittle, Christina von Flach Garcia Chavez, and Carlos Lucena. 2015. Evaluating the effort of composing design models: a controlled experiment. *Software & Systems Modeling* 14 (2015), 1349–1365.

[11] Eduardo Fernandes, Alexander Chávez, Alessandro Garcia, Isabella Ferreira, Diego Cedrim, Leonardo Sousa, and Willian Oizumi. 2020. Refactoring effect on internal quality attributes: What haven't they told you yet? *Information and Software Technology* 126 (2020), 106347.

[12] Arlene Fink. 2003. *The survey handbook*. sage.

[13] Barney G Glaser and Anselm L Strauss. 2017. *Discovery of grounded theory: Strategies for qualitative research*. Routledge.

[14] Sebastian Herold, Christoph Knieke, Mirco Schindler, and Andreas Rausch. 2020. Towards Improving Software Architecture Degradation Mitigation by Machine Learning. In *The Twelfth International Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2020), Nice, France, October, 26-29 2020*.

[15] Ed Júnior and Kleinner Farias. 2021. Modelgame: A quality model for gamified software modeling learning. In *15th Brazilian Symposium on Software Components, Architectures, and Reuse*. 100–109.

[16] Ed Júnior, Kleinner Farias, and Bruno Silva. 2021. A Survey on the Use of UML in the Brazilian Industry. In *XXXV Brazilian Symposium on Software Engineering*. 275–284.

[17] Ed Wilson Júnior, Kleinner Farias, and Bruno da Silva. 2022. On the use of UML in the Brazilian industry: A survey. *Journal of Software Engineering Research and Development* 10 (2022), 10–1.

[18] Hanna Kallio, Anna-Maija Pietilä, Martin Johnson, and Mari Kangasniemi. 2016. Systematic methodological review: developing a framework for a qualitative semi-structured interview guide. *Journal of advanced nursing* 72, 12 (2016), 2954–2965.

[19] Ruiyin Li, Peng Liang, Mohamed Soliman, and Paris Avgeriou. 2022. Understanding software architecture erosion: A systematic mapping study. *Journal of Software: Evolution and Process* 34, 3 (2022), e2423.

[20] Ruiyin Li, Mohamed Soliman, Peng Liang, and Paris Avgeriou. 2022. Symptoms of architecture erosion in code reviews: a study of two OpenStack projects. In *2022 IEEE 19th International Conference on Software Architecture (ICSA)*. IEEE, 24–35.

[21] Maicon Azevedo da Luz and Kleinner Farias. 2020. The use of blockchain in financial area: A systematic mapping study. In *Proceedings of the XVI Brazilian Symposium on Information Systems*. 1–8.

[22] Antonio Martini, Viktoria Stray, and Nils Brede Moe. 2019. Technical-, social-and process debt in large-scale agile: an exploratory case-study. In *Agile Processes in Software Engineering and Extreme Programming–Workshops: XP 2019 Workshops, Montréal, QC, Canada, May 21–25, 2019, Proceedings 20*. Springer, 112–119.

[23] Juliano Paulo Menzen, Kleinner Farias, and Vinicius Bischoff. 2021. Using biometric data in software engineering: a systematic mapping study. *Behaviour & Information Technology* 40, 9 (2021), 880–902.

[24] Anderson Oliveira, Vinicius Bischoff, Lucian José Gonçales, Kleinner Farias, and Matheus Segalotto. 2018. BRCode: An interpretive model-driven engineering approach for enterprise applications. *Computers in Industry* 96 (2018), 86–97.

[25] Anderson Oliveira, Willian Oizumi, Leonardo Sousa, Wesley KG Assunção, Alessandro Garcia, Carlos Lucena, and Diego Cedrim. 2022. Smell Patterns as Indicators of Design Degradation: Do Developers Agree?. In *Proceedings of the XXXVI Brazilian Symposium on Software Engineering*. 311–320.

[26] Tajmilur Rahman, Joshua Nwokeji, and Tejas Veeraganti Manjunath. 2022. Analysis of Current Trends in Software Aging: A Literature Survey. *Computer and Information Science* 15, 4 (2022).

[27] Maluane Rubert and Kleinner Farias. 2022. On the effects of continuous delivery on code quality: A case study in industry. *Computer Standards & Interfaces* 81 (2022), 103588.

[28] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering* 14 (2009), 131–164.

[29] Robson Keemps Silva, Kleinner Farias, Rafael Kunst, and Jovani Dalzochio. 2023. An Approach Based on Machine Learning for Predicting Software Design Problems. In *Proceedings of the XIX Brazilian Symposium on Information Systems*. 53–60.

[30] Fangchao Tian, Peng Liang, and Muhammad Ali Babar. 2022. Relationships between software architecture and source code in practice: An exploratory survey and interview. *Information and Software Technology* 141 (2022), 106705.

[31] Roger Gonçalves Urdangarin, Kleinner Farias, and Jorge Barbosa. 2021. Mon4Aware: A multi-objective and context-aware approach to decompose monolithic applications. In *XVII Brazilian Symposium on Information Systems*. 1–9.

[32] Roger Denis Vieira and Kleinner Farias. 2020. Usage of psychophysiological data as an improvement in the context of software engineering: A systematic mapping study. In *XVI Brazilian Symposium on Information Systems*. 1–8.

[33] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in software engineering*. Springer Science & Business Media.