Publicação: Outubro de 2019

Digital Object Identifier

Identificando e Avaliando Dívidas Técnicas no Processo de Testes de Software

CLEYDIANE LIMA DE SOUSA[1], ARILO CLAUDIO DIAS NETO[2].

1 - Centro de Estudos e Sistemas Avançado do Recife (CESAR)
2- Instituto de Computação (IComp), Universidade Federal do Amazonas, Manaus, Brasil
(e-mail:cleydiane.lima@cesar.org.br, arilo@icomp.ufam.edu.br)
Autor Correspondente: Cleydiane Lima de Sousa (e-mail: cleydiane.lima@cesar.org.br).

RESUMO - Dívida Técnica (DT) está relacionada a tarefas que devem ser executadas e são acumuladas ao longo de um projeto para serem realizadas posteriormente. DTs não solucionadas tendem a criar dependências e aumentar o grau de complexidade para correção, resultando em maior esforço, custo e retrabalho em um projeto. A sua identificação e gerenciamento ao longo de um projeto é essencial para minimizar seu impacto e consequências negativas. Estudos apontam teste de software como sendo uma das principais áreas impactadas por DTs em projetos de software. Neste artigo foram mapeadas a partir da literatura técnica 22 possíveis DTs relacionadas ao processo de teste, suas causas e indicadores. Elas foram avaliadas por meio de um survey com os profissionais da área de teste de software. Como resultados do survey, são apresentados os níveis de concordância obtidos para cada DT, suas causas e indicadores sugeridos.

PALAVRAS-CHAVE - Dívida Técnica, Teste de Software, survey.

I. INTRODUÇÃO

O aumento da demanda por novos produtos de software tem desafiado cada vez mais as organizações de software a atender aos prazos e qualidade exigidos pelos seus usuários. Isto requer alguns sacrifícios diante da possível falta de tempo, recursos, profissionais capacitados, dentre outras limitações ou imprevistos em um projeto de software. De acordo com Pressman [1], em geral, nos projetos de software, quando os prazos ou recursos se tornam escassos, as organizações tendem a comprometer tarefas e práticas relacionadas à qualidade de software. Como consequência, problemas de qualidade podem ser observados no produto durante o projeto ou após sua implantação. Tais tarefas comprometidas precisam ser concluídas em algum momento ao longo do projeto, ou se não concluídas, podem gerar uma dívida, ou déficits ao projeto. Assim, surge o conceito de Dívida Técnica (DT: do inglês Technical Debt) dentro da área de Engenharia de Software, que representa um conjunto de tarefas que devem ser executadas e são acumuladas ao longo de um projeto para serem realizadas posteriormente.

O conceito de DT foi criado por Cunningham [2], que definiu como: "A primeira vez que a qualidade do código é comprometida é como se estivesse incorrendo em débito. Um pequeno débito acelera o desenvolvimento até que seja pago por meio da reescrita do código. O perigo ocorre quando o débito não é pago. Cada minuto em que o código é mantido em inconformidade, juros são acrescidos na forma de reescrita". Assim, a DT que não é solucionada com o decorrer do tempo tende a criar dependências e aumentar o grau de complexidade para correção, ou até mesmo ocorrer sua despriorização pelo acúmulo de novas dívidas. Este tema vem ganhando importância dentro das diversas áreas da Engenharia de Software. Li et al. [3] apresentam um levantamento das áreas que podem ocasionar DT. Dentre as áreas apresentadas neste estudo, a área de teste de software possui um grande destaque como causadora de DTs. Segundo Bertolino [4], teste de software abrange diversas atividades no decorrer do ciclo de desenvolvimento de software,

nas quais são realizadas observações para verificar o comportamento do que está sendo desenvolvido e validar se está como o esperado. De acordo com a norma ISO/IEC/IEEE 29119 [5], um processo de testes é formado por diversas atividades, incluindo: planejamento, monitoramento e controle, projeto, execução e análise dos resultados dos testes. Ao longo deste processo, a equipe de testes pode deixar de realizar atividades ou deixá-las para outro momento, gerando DT que precisam ser identificadas, controladas e solucionadas ao longo do projeto.

No entanto, a identificação e monitoramento de DTs relacionadas ao processo de teste de software, suas eventuais causas e os indicadores que permitem observá-las em um projeto de software ainda são pouco explorados em pesquisas acadêmicas. Em geral, elas estão dispersas em diversas fontes (artigos científicos e relatos da indústria). Baseado neste cenário, este artigo apresenta um conjunto de 22 DTs relacionadas ao processo de testes de software (reportadas na literatura técnica), as eventuais causas para cada DT e indicadores que possibilitam a sua identificação em um projeto de software. Tais DTs foram organizadas em 2 grupos: DTs relacionadas a atividades gerenciais de teste (planejamento, monitoramento e controle) e DTs relacionadas a atividades técnicas de teste (projeto e execução dos testes). Este conjunto foi avaliado por meio de um survey com a participação de 64 especialistas em teste de software, que responderam o quanto concordam com cada DT, suas causas e indicadores. Tais DTs foram ordenadas de acordo com o grau de concordância obtido ao longo do survey e uma análise quantitativa e qualitativa dos resultados é apresentada. O artigo segue a seguinte estrutura: Seção II apresenta o referencial teórico, descrevendo os conceitos de DT e do processo de TS e apresenta as DTs em TS identificadas a partir da literatura técnica. Seção III descreve o planejamento e execução do survey, com objetivo de avaliar a concordância dos profissionais de TS quanto as DTs identificadas na literatura. Seção IV apresenta os resultados obtidos a partir da análise dos dados coletados ao longo do survey. Finalmente, a seção V apresenta as conclusões e trabalhos futuros.

II. REVISÃO DA LITERATURA

A. DÍVIDA TÉCNICA EM ENGENHARIA DE SOFTWARE

Cunningham [2] é considerado o pioneiro em DT em Engenharia de Software. Para ele, em curto prazo, adquirir uma DT pode se apresentar como um fator positivo, desde que se tenha a visão que no futuro é necessário pagar a dívida para não acumular. Este cenário se apresenta, por exemplo, em situações em que é necessário gerar uma DT de código mal escrito para contornar problema de prazo curto para entrega de um módulo.

Em contrapartida à visão de DT centrada em código, Klinger [6] enfatiza o fato que uma DT não vem somente da parte técnica, de código. Ela vai além disso. Outros fatores interferem em seu surgimento, como diversas funções técnicas e não-técnicas, tais como testes de produtos, estratégia de marca, questões jurídicas, marketing, dentre outros. Observar estes fatores e o que mais envolve todo o processo de um produto ajudam a entender melhor porque uma DT veio a ocorrer e foi ou não paga.

Dentre outras definições, Guo [7] considera DT como "uma metáfora para artefatos imaturos, incompletos ou inadequados no ciclo de vida de desenvolvimento de software que causam maiores custos e menor qualidade". A criação destes artefatos pode acelerar o desenvolvimento em curto prazo. Porém, em longo prazo, a má qualidade dos artefatos tende a gerar um maior custo, devido esforços de manutenção e força de trabalho para correções. Torna-se perceptível os riscos no acúmulo de dívidas técnicas, as dependências entre as dívidas e outros artefatos aumentam gradativamente a complexidade de pagamento destas dívidas.

Para Shull [8], a DT varia de equipe para equipe, havendo a necessidade da equipe refletir sobre qual tipo de dívida se deve considerar com uma maior preocupação e estar sempre monitorando. O monitoramento da DT é necessário para não se perder o controle e não adquirir nova DT sem perceber, acumulando-a ao ponto de quando for percebida ser muito caro para ser paga.

Quanto a relevância do tema para comunidade de pesquisa, Alves et al. [9] apresentam uma revisão sistemática que apontou que o tema DT obteve trabalhos com publicações principalmente a partir de 2010. Porém, ainda é um tema novo e seus diferentes tipos e indicadores ainda não estão organizados. Já Li et al. [3] complementam estes dados por meio de sua pesquisa que identificou trabalhos publicados a partir do ano de 1992, ano em que o conceito de DT foi apresentado, até 2013.

B. PROCESSO DE TESTE DE SOFTWARE

Segundo Mettle e Hass [10], o processo de testes consiste nas atividades do ciclo de vida, tanto estática e dinâmica, preocupadas com planejamento, preparação e avaliação de produtos de software e produtos de trabalho relacionados para determi-

nar que satisfaçam os requisitos especificados, para demonstrar que estão aptos para o efeito e para detectar defeitos. Um processo de teste pode definir melhor as atividades a serem realizadas dentro da área de testes, além de prover melhorias para que as atividades sejam feitas de maneira organizada.

A norma ISO/IEC/IEEE 29119 [5] sugere um processo de testes de software com etapas e artefatos para serem gerados. O processo de testes é dividido em três subprocessos: organizacional, gerenciamento e dinâmico. O processo de teste organizacional deve compreender atividades para a criação, revisão e manutenção de especificações de teste organizacionais. Ele também deve abranger o monitoramento da conformidade organizacional [5]. O objetivo destas atividades dentro das organizações está em ajudar a se ter um processo e controlar melhor o que está sendo realizado para que os testes ocorram com sucesso. Os outros dois subprocessos são formados pelas seguintes atividades [5]:

- Processo de Gerenciamento dos Testes: Planejamento do Processo de Teste, Monitoramento e Controle de Teste e Conclusão de Teste.
- Processo de Teste Dinâmico: Projeto e Implementação dos Testes, Configuração e Manutenção do Ambiente de Teste, Execução dos Testes e Reporte dos Incidentes de Teste.

Tais atividades serão usadas neste trabalho para organização das DTs relacionadas ao processo de testes, indicando o momento em que elas surgem em um projeto.

C. TRABALHOS RELACIONADOS SOBRE DÍVIDA TÉCNICA EM TESTE DE SOFTWARE

A maioria dos trabalhos que abordam DT apresenta seu foco em DTs identificadas na etapa de codificação. Nesta seção, serão apresentados trabalhos centrados em DT na área de teste de software com o objetivo de identificar as principais contribuições.

Shah et al. [11] relatam sobre a DT estar relacionada a diversas atividades, em que, uma delas é a de teste de software. Os autores citam DTs dentro de teste de software, focando em DTs que podem vir a ocorrer quando há execução de somente testes exploratórios. Os autores ainda destacam que a falta de planejamento e documentação tendem a criar uma grande dificuldade para realizar testes de regressão de forma adequada, o que tende a gerar retrabalho e também uma cobertura de defeitos inadequada. Logo, adquire-se uma DT maior a cada vez que se cria mais funcionalidades para o produto, e a regressão não identifica os defeitos.

Bavani [12] descreve uma entrevista com Johanna Rothman e Lisa Crispin (importantes pesquisadores da área de testes ágeis) realizada em 2012 que busca identificar a opinião deles sobre DT em equipes ágeis e distribuídas. Os entrevistados se preocupam em sanar as DTs que aparecem no processo de teste, e uma sugestão é automatizar, mas sem perder tempo com a manutenção. Além disso, para sanar problemas em equipes distribuídas, Crispin e Rothman concordam que as equipes de teste devem ser integradas com o restante da equipe, e que é necessário ter um ritmo sustentável para não surgirem novas DTs.

Wiklund [13] descreve uma pesquisa que visou identificar os contribuintes comuns a DTs acumuladas em testes automatizados, para avaliar a consciência desta dívida em organizações que usam o teste automatizado. A pesquisa foi realizada a partir de uma entrevista semiestruturada com participantes de uma equipe de software em um projeto de telecomunicações. Em suas conclusões, o autor relatou que não foi possível detectar como planejar, monitorar e deixar a DT de automação em um valor aceitável, destacando problemas existente que a automação apresenta, sem visualizar soluções.

Snipes et al. [14] apresentam um estudo que identificou os fatores na tomada de decisão para corrigir defeitos e os custos que envolvem isto, sob a perspectiva de DTs. Os autores identificaram vários custos que são encadeados com a manutenção de defeitos. Foi ressaltado que na tomada de decisão para corrigir os defeitos, para incluir DT é necessário incluir custo-benefício, acrescentando-os junto com os fatores identificados.

Deve-se levar em consideração que os trabalhos relacionados à DT em teste de software, além de serem poucos, apresentam DTs em tipos específicos de teste (ex: testes exploratórios ou automação de testes). Assim, realizar um levantamento de DTs no processo de testes de software se mostra uma importante contribuição para comunidade de engenharia de software, com objetivo de cada vez mais ser cumprido o que for planejado dentro de um processo de testes de software, sem deixar em débito o que possa prejudicar a saúde de um projeto.

III. METODOLOGIA

A. DÍVIDAS TÉCNICAS EM TESTE DE SOFTWARE

Para identificar DTs na literatura técnica, foram realizadas pesquisas manuais em diversas bibliotecas digitais, tais como: ACM Digital Library, IEEExplore, Scopus, ScienceDirect e Google Scholar.

Os trabalhos que foram identificados abordam DT em teste de software, além de trabalhos em teste de software que não lidam com termo DT, porém, apresentam menção de problemas (dificuldades) na área.

Depois da identificação das DT, foram apontados causas e indicadores de cada DT. Neste contexto, Causa é o acontecimento que é feito ou deixa de ser feito para gerar a DT; e os Indicadores são indícios para perceber a DT em um projeto de software. Nas Tabelas 1 e 2 são apresentadas as DTs que foram encontradas na literatura técnica, separadas por DTs associadas ao Gerenciamento dos Testes (Tabela 1) e Projeto e Execução dos Testes (Tabela 2). Ao lado do título de cada DT, estão apresentadas as referências de onde elas foram extraídas. Também são apresentados as causas e indicadores associados a cada DT em análise.

Em seguida foi realizado o survey para a validação do conjunto de DT junto a especialistas em teste de software, a ser descrito na próxima seção.

B. SURVEY SOBRE DTS, CAUSAS E INDICADORES EM TESTE DE SOFTWARE

Nesta seção serão apresentados o planejamento e projeto de um survey que visou investigar um conjunto de dívidas técnicas (DTs) relacionadas ao processo de testes de software, suas causas e indicadores, sob o ponto de vista de profissionais da área.

O survey busca caracterizar como testadores avaliam as DTs identificadas na literatura técnica como relacionadas à área de teste de software, e ainda se há variações de percepção dentro da população analisada quanto ao nível de relevância.

1) Questão de Pesquisa

A questão de pesquisa que norteou este estudo foi a seguinte: Qual o nível de concordância de 22 DTs (suas causas e indicadores) presentes na literatura técnica sob o ponto de vista de profissionais de Teste de Software? A Variável Independente é o Conjunto Inicial de DTs, causas e indicadores (Tabelas 1 e 2). As Variáveis Dependentes são: O nível de concordância para cada DT, causa e indicador, pertencentes ao conjunto inicial e mantidos no conjunto final de acordo com a opinião dos profissionais.

2) Definição do Instrumento

O instrumento adotado neste survey é formado por quatro etapas sequenciais, descritas a seguir:

Convite. A partir do link presente no e-mail do convite, o participante era redirecionado à página de apresentação do survey, que dá acesso ao questionário. Apresentação. Foi desenvolvida uma página web contendo informações sobre o survey, e ao final desta página a solicitação do consentimento para participarem do survey.

Caracterização do Participante. Após a tela de apresentação o participante era direcionado para um formulário de caracterização, que apresenta o objetivo da pesquisa e perguntas que visam caracterizar o grau de conhecimento/experiência dos participantes a respeito do tópico do estudo.

Avaliação de DTs, Causas e Indicadores. Devido ao grande número de DTs e suas classificações dentro do processo de testes, optou-se pela separação da avaliação das DTs (suas causas e indicadores) em dois questionários, que deveriam ser respondidos por grupos diferentes de participantes, de acordo com o perfil de cada participante, definido no passo anterior. Os questionários são:

- O questionário “Planejar e Concluir”: apresenta 9 Dívidas Técnicas, causas e indicadores nos processos gerenciais de Planejamento, Monitoramento e Controle de Teste de Software e Análise/Conclusão de Resultados (conforme a ISO 29119).
- O questionário ¨Projetar e Executar¨: apresenta 13 Dívidas Técnicas, causas e indicadores dos processos de Projetar e Executar Teste de Software (conforme ISO 29119).

Por fim, ao final de cada um dos dois questionários foi oferecido um campo para comentários gerais, que poderia ser preenchido como campo opcional. Em ambos os questionários, para cada DT (suas causas e indicadores), foram apresentadas seis opções de respostas (obrigatório), além de um campo para comentários (opcional). As opções de respostas são apresentadas na Tabela 3.

Através da combinação entre as opções de resposta, é possível identificar a concordância dos participantes quanto aos itens de uma DT (descrição da DT, causas e indicadores). Por exemplo, as respostas R1, R2 e R4 (Tabela 3) são opções que representam a concordância dos participantes quanto as causas da DT analisada. Desta forma, para cada questão, a partir do somatório dos participantes que concordaram com determinado item (dívidas técnicas, causas ou indicadores) é realizada a divisão pelo total de participantes que responderam ao questionário, resultando assim em uma porcentagem que representa o nível de concordância dos participantes quanto o item analisado.

Na próxima seção são apresentados os resultados obtidos a partir da análise das respostas coletadas

Tab. 1: Dívidas Técnicas, Causas e Indicadores associados ao Gerenciamento dos Testes.

ID	Dívidas Técnicas	Causas	Indicadores
DT01	Cronograma de teste não definido/ inadequado [15] [16] [17]	- Falta de experiência da equipe de teste	- Cronograma não foi criado
		- Tempo limitado para os testes como premissa do projeto	- Cronograma sem considerar atividades nos processos de testes
		- Não saber quando iniciar os testes	
		- Não saber quando terminar os testes	
DT02	Ferramentas de teste não utilizadas/ aplicáveis no projeto [4] [15] [16] [18]	- Falta de experiência da equipe de teste	- Quantidade de recurso disponível menor que o valor das ferramentas
		- Falta de recurso	- Diversidade de ferramentas disponíveis menor do que o esperado para a cobertura dos testes
DT03	Riscos de teste não definidos/inadequados [15] [19]	- Ausência de critérios para seleção dos testes	- Não ter os riscos mapeados
			- Premissas do projeto que impactam nos testes não consideradas nos levantamentos de riscos
DT04	Planejamento de testes não adequados para o projeto [11] [16] [18] [20]	- Não entendimento dos requisitos do projeto	- Tipos de testes não cobrem os requisitos
		- Não buscar entender melhor o projeto	
DT05	Equipe de teste não definida/inadequada para projetar e executar os testes [11] [15] [18]	- Falta de recurso	- Planejar uma quantidade X de pessoas e ter disponível menos pessoas
		- Falta de comunicação dos contratantes com o gerente de testes	- Não ter todos os perfis mapeados
		- Falta de planejamento e estratégia do projeto	
		- Ausência de critérios para seleção de equipe	
DT06	Fornecer um entregável com bugs [20] [21]	- Falta de regressão	- % bugs sendo encontrados em funcionalidades já testadas
		- Falta de tempo para corrigir	- % bugs conhecidos
DT07	Bugs sem correções [21]	- Falta de relatório dos bugs	- % Bugs não corrigidos
		- Falta de tempo para correção	
DT08	Falta de controle de quais tipos de testes estão sendo executados [13] [15]	- Falta de relatório de execução	- A quantidade de tipos de testes executados pela equipe não está presente em relatórios
		- Falta de tempo de fazer relatórios	
DT09	Problemas no software não aceito pelo cliente [16] [22]	- Cliente não ter conhecimento da estratégia (critérios de aceitação e saída)	- Quantidade de criticidades dos bugs para aquela entrega não aceita pelo cliente
		- Equipe de teste não checando a documentação do projeto (ex: caso de uso, ux-guide)	- % de cobertura dos testes realizados, não aceitos pelo cliente
			- Cliente esperando que sejam feitos testes que não estão sendo aplicados
DT10	Scripts de teste não implementados [3]	- Tempo limitado para os testes	- % de scripts de teste implementados (esperado / realizado)
		- Equipe de teste inexperiente	
		- Entrega do desenvolvedor atrasada	
		- Cronograma de teste não definido/inadequado	
DT11	Criação de cenários de testes não finalizados [3][10] [16]	- Tempo limitado para os testes	- % de cenários de testes implementados (esperado / realizado)
		- Equipe de teste inexperiente	
		- Falta de definição dos requisitos	
		- Cronograma de teste não definido/inadequado	
DT12	Falta de documentação dos testes que serão executados [11] [13] [15] [16]	- Equipe de teste inexperiente	- Quantidade de documentos
		- Tempo limitado para os testes	
		- Cronograma de teste não definido/inadequado	

Tab. 2: Dívidas Técnicas, Causas e Indicadores associados ao Projeto e Execução dos Testes.

ID	Dívidas Técnicas	Causas	Indicadores
DT13	Necessidade de refatorar os scripts de testes [15] [16]	- Equipe de teste inexperiente	- Falta de atributos no código para scripts de testes
		- Falta de conhecimento técnico	
		- Falta de organização na hora de iniciar os scritps	
		- Alteração na funcionalidade	
DT14	Cenários/scripts de testes que não representam o estado atual da funcionalidade [16] [23] [24]	- Não realizar manutenção nos casos de testes e scripts de testes	- % de cenários/scripts de testes não coerente com a atual funcionalidade
		- Equipe de teste inexperiente	- Quantidade de tempo disponível para manutenção menor que o planejado
DT15	Casos/scripts de testes desatualizados na ferramenta de gestão [15] [16] [17]	- Tempo limite para os testes	- % de testes sendo projetados sem incluir/atualizar os casos de testes / scripts das ferramentas de gestão
		- Cronograma de teste não definido/inadequado	- Quantidade de tempo disponível para manutenção menor que o planejado
		- Falta de manutenção	
DT16	Casos de testes não executados [3] [25]	- Atraso na liberação da versão para teste	- Quantidade de casos de testes não executados (ou %)
		- Tempo curto para execução dos testes	- Quantidade de tempo disponível para execução menor que o planejado
		- Execução apenas de testes automatizados	
		- Cronograma de teste não definido/inadequado	
		- Regressão não executada	
DT17	Scripts de Teste não executados [3]	- Atraso na liberação da versão de teste	- Quantidade de scripts de testes não executados (ou %)
		- Tempo curto para execução dos testes	- % Scripts falhando
		- Cronograma de teste não definido/inadequado	- Quantidade de tempo disponível para execução menor que o planejado
		- Regressão não executada	
DT18	Sobrecarga de atividades em determinados períodos do projeto [16] [27]	- Cronograma de teste não definido/inadequado	- Quantidade de tempo para trabalho menor que o tempo planejado para as atividades no período
		- Falta de divisão de atividades	- % de versões lançadas em determinado período do projeto
		- Atraso na liberação da versão para teste	
DT19	Bugs encontrados tardiamente [15] [20]	- Falta de Cronograma	- % de tipos de testes não sedo executados há tempo ou não sendo executados
		- Falta de integração contínua	- % de bugs encontrados em momentos errados (sem término de desenvolvimento)
		- Desenvolvimento atrasado	
		- Testes iniciados tardiamente	
DT20	Acúmulo de bugs não corrigidos [21]	- Não corrigir os bugs por criticidade	- % funcionalidades importantes com erros graves
		- Falta de tempo para correções	- % de bugs com criticidades erradas
		- Equipe de desenvolvimento não disponibiliza tempo para corrigir bugs	
DT21	Falta de cobertura dos requisitos do sistema (funcionais e não funcionais) [3] [16] [20]	- Inexperiência dos testadores	- Quantidade de bugs sendo encontrados por falta de cobertura de certos tipos de testes
		- Execução de somente um tipo de teste	- % de requisitos não cobertos pelos tipos de testes aplicados
DT22	Funcionalidades já finalizadas presentando bugs [15] [20]	- Pouco tempo para executar os testes	- % bugs sendo encontrados em funcionalidades já testadas
		- Testes não fazem mais sentido (funcionalidade alterada)	- % de bugs encontrados nos testes de regressão
		- Não executar testes de regressão	

Tab. 3: Opções de Resposta

Opções	Respostas
R1	Concordo com a Dívida Técnica, causas e indicadores
R2	Concordo com a Dívida Técnica e com as causas, mas não concordo com os indicadores
R3	Concordo com a Dívida Técnica e com os indicadores, mas não concordo com as causas
R4	Concordo com as causas e os indicadores, mas não concordo com a Dívida Técnica
R5	Não concordo com a Dívida Técnica, causas e indicadores
R6	Concordo com a Dívida Técnica, mas não concordo com as causas e indicadores

$$n = \frac{N.\frac{1}{E_0^2}}{N + \frac{1}{E_0^2}} \rightarrow 64 = \frac{248.\frac{1}{E_0^2}}{248 + \frac{1}{E_0^2}} \rightarrow E_0 = \sqrt{\frac{248 - 64}{248 * 64}} \rightarrow E_0 = 0{,}107 \rightarrow Confiança = 89{,}3\%$$

Onde: N = tamanho da população | n = tamanho da amostra | E_0 = nível de confiança (ex: 0,05 → 95%)

Fig. 1: Cálculo do Nível de Confiança de uma amostra baseado em [28]

com o survey.

IV. RESULTADOS

A. PERÍODO E TOTAL DE PARTICIPANTES DO SURVEY

Quanto ao período de realização da coleta de dados, o survey deste artigo é de corte transversal, pois os dados foram coletados em apenas um momento, do dia 01/06/2015 a 22/06/2015, no endereço http://icomp.ufam.edu.br/experts/survey/. A população de estudo foi focada em profissionais da área de Teste de Software que trabalham em empresas brasileiras centradas em desenvolvimento de software.

Foram enviados um total de 248 e-mails, destes convites um total de 64 participantes finalizaram a pesquisa, em que 19 responderam o questionário Planejar e Concluir e 45 responderam o questionário Projetar e Executar. Para calcular o nível de confiança da amostra deste estudo, utilizou-se a fórmula descrita na Figura 1. Sendo o tamanho da população de 248 pessoas e tamanho da amostra de 64 participantes, isso resulta em 89,3% de nível confiança para a amostra obtida.

B. ANÁLISE DA CARACTERIZAÇÃO DOS PARTICIPANTES

Na Figura 2a, o número de participantes com experiência acima de 5 anos em teste de software foi de 51,6%. A pesquisa contou ainda com um número

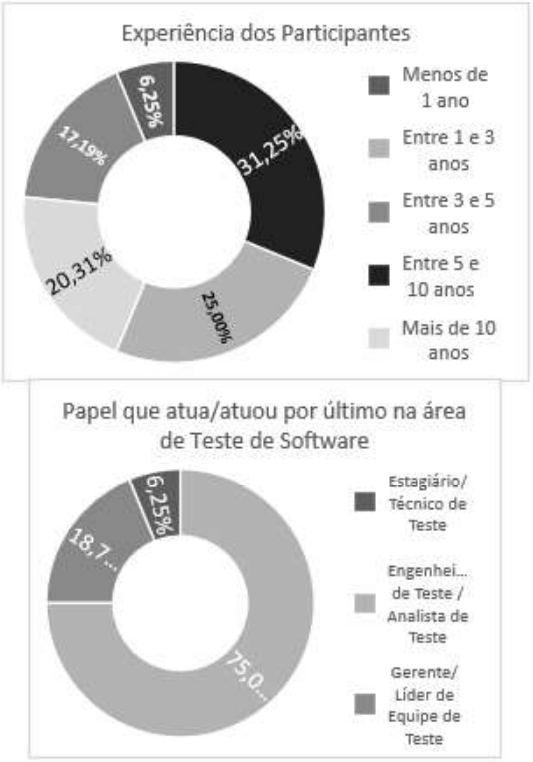

Fig. 2: (a) Distribuição de Experiência dos Participantes. (b) Papel da Área de Testes.

relevante de participantes com experiência entre 3 e 5 anos (17,2%). Por fim, o fato de 31,3% dos participantes terem menos 3 anos de experiência ajudou para que a pesquisa apresente uma variação de experiência nas respostas.

A Figura 2b apresenta a distribuição dos papéis que os participantes atuam/atuaram por último na área de teste de software. É possível observar que a maioria dos participantes atua/atuou como Engenheiros/Analistas de Testes (75%), seguido por Gerentes/Líderes de testes (18,8%) e, finalmente, estagiários/técnicos em teste (6,3%).

C. ANÁLISE DO RESULTADO DO SURVEY

Para análise dos dados gerados a partir da aplicação do survey, foi realizada uma análise quantitativa a partir de estatística descritiva.

A Tabela 4 apresenta os resultados das DTs presentes nos questionários, divididos por questionário: DT01 a DT09 refere-se ao questionário de Planejar e Concluir, no qual foram obtidas 19 respostas, enquanto que DT10 a DT22 são as respostas do questionário de Projetar e Executar, que contou com 45 participantes. Nessa tabela, cada linha representa um grupo de DT, causas e indicadores analisado

no estudo. As colunas de R1 à R6 representam a contagem das possíveis opções de resposta que os usuários escolheram para cada uma das DT. As três últimas colunas condizem com o número de participantes que concordaram com os seguintes itens:

- Dívidas: número participantes que concordaram com o conceito que define a DT analisada. Essa opção consiste na soma das colunas R1+R2+R3, pois todas estas opções estão associadas à concordância com a DT.
- Causas: número participantes que concordaram com causas relacionadas à DT que está sendo analisada. Essa opção consiste na soma das colunas R1+R2+R4, pois todas estas opções estão associadas à concordância com as causas.
- Indicadores: número participantes que concordaram com os indicadores relacionados à DT que está sendo analisada. Essa opção consiste na soma das colunas R1+R3+R4, pois todas estas opções estão associadas à concordância com os indicadores que relacionados à DT.

Por exemplo, para a DT1 (Cronograma de teste não definido/inadequado), a opção de resposta R1 (Concorda com a Dívida Técnica causas e indicadores) foi escolhida 8 vezes. Neste exemplo, 15 pessoas, do total de 19, concordaram com os indicadores da primeira Dívida Técnica.

Nas próximas seções serão apresentadas as análises referentes às DTs, causas e indicadores, separando-as por tipo de questionário aplicado ("Planejar e Concluir" e "Projetar e Executar").

D. DÍVIDA TÉCNICA, CAUSAS E INDICADORES DO QUESTIONÁRIO PLANEJAR E CONCLUIR

A Figura 3 exibe o percentual de concordância referente a DTs, causas e indicadores do questionário Planejar e Concluir, pela ordem decrescente de concordância que cada item obteve. A DT que obteve maior concordância foi "Riscos de teste não definidos/inadequados" (DT3) e a que teve menor grau de concordância foi a DT7 "Bugs sem correções", com uma diferença considerável para a DT com maior concordância. As demais DTs, apresentaram uma variação de concordância entre 70% e 90%. Entre todos os comentários realizados pelos participantes do questionário Planejar e Concluir, a maioria foi direcionada a opiniões referentes às causas e indicadores. Os poucos comentários referentes às DTs apresentam discordância da DT em algum contexto. Por exemplo, um participante comentou sua discordância quanto à DT1 (Cronograma de teste não definido/inadequado) no contexto de

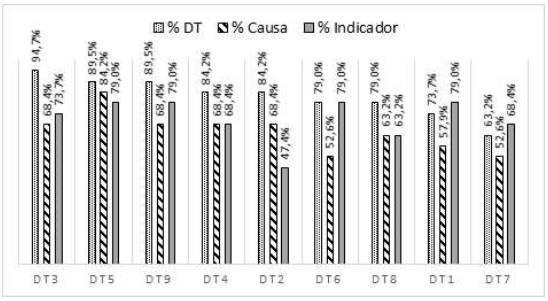

Fig. 3: Análise de DTs, Causas e Indicadores do Questionário "Planejar e Concluir Testes".

sua aplicação em projetos ágeis: "No processo que testes é considerado no final, posso concordar, mas em uma metodologia ágil não faz sentido um cronograma de testes e sim um cronograma do time como um todo". A Figura 3 ainda apresenta a análise de concordância das causas de cada DT. As causas, diferentemente das DTs, apresentaram porcentagens de grau de concordância menores, ficando a maioria, com exceção de uma, abaixo de 69%.

O fato de as causas analisadas terem apresentado um menor grau de concordância não significa que os participantes tenham discordado completamente delas. Pode-se observar por alguns comentários dos participantes que em determinados casos a discordância das causas ocorreu por acharem que mais de uma causa da mesma DT tinham o mesmo significado e deveriam ser unificadas. Em outros comentários, os participantes sugeriram que o conjunto das causas da DT estavam incompletas e era necessário acrescentar novas causas. Por exemplo: um participante sugeriu acrescentar uma causa: "Outra causa: Falta de automação, teste unitário na build, etc.", comentário referente à DT6 (Fornecer um entregável com bugs).

Por fim, pode observar a partir da Figura 3 que os indicadores, assim como as causas, tiveram graus de concordância menores que as DTs. O indicador que obteve o grau de concordância mais baixo (47,4%) apresenta o menor grau de concordância de toda a análise. Este indicador está relacionado à DT2 (Ferramentas de teste não utilizadas/aplicáveis no projeto), com os indicadores: quantidade de recurso disponível menor que os valores das ferramentas; diversidade de ferramentas disponíveis menor do que o esperado para a cobertura dos testes. Alguns dos comentários referentes à DT2 indicaram que existem muitas ferramentas sem custo e o fator financeiro não seria um indicador. Isto pode refletir a baixa concordância para os indicadores da DT2.

Como ocorreu com as causas, os participantes

Tab. 4: Resumos das Respostas Obtidas por Dívida Técnica Avaliada.

	Itens Analisados	R1	R2	R3	R4	R5	R6	Dívidas	Causas	Indicadores
Planejar e Concluir	DT01	8	1	5	2	3	0	14	11	15
	DT02	7	6	2	0	3	1	16	13	9
	DT03	11	2	3	0	1	2	18	13	14
	DT04	10	2	2	1	2	2	16	13	13
	DT05	14	2	1	0	2	0	17	16	15
	DT06	8	1	6	1	3	0	15	10	15
	DT07	7	1	4	2	5	0	12	10	13
	DT08	9	2	2	1	3	2	15	12	12
	DT09	11	2	4	0	2	0	17	13	15
Projetar e Executar	DT10	29	8	6	1	1	0	43	38	36
	DT11	27	6	9	1	0	2	44	34	37
	DT12	16	10	6	2	9	1	33	28	24
	DT13	29	9	4	1	1	1	43	39	34
	DT14	29	4	7	0	3	2	42	33	36
	DT15	33	5	1	1	3	1	40	39	35
	DT16	33	2	7	1	1	0	42	36	41
	DT17	32	2	7	2	2	0	41	36	41
	DT18	30	4	5	1	0	1	40	35	36
	DT19	35	2	6	0	0	1	44	37	41
	DT20	35	6	1	0	1	1	43	41	36
	DT21	32	1	9	0	1	1	43	33	41
	DT22	34	1	7	0	1	2	44	35	41

não necessariamente discordavam completamente dos indicadores. Eles concordavam em partes, pediam para acrescentar mais indicadores ou retirar algum deles. Dentre os comentários, um participante acrescentou na DT8 (Falta de controle de quais tipos de testes estão sendo executados): "Acredito que temos muitos indicadores aí: progressão de testes, planejados X executados, critérios de saída de testes, testes por tipo, entre outros para a DT".

E. DÍVIDA TÉCNICA, CAUSAS E INDICADORES DO QUESTIONÁRIO PROJETAR E EXECUTAR

A Figura 4 exibe o percentual de concordância referente a DTs, causas e indicadores do questionário Projetar e Executar, pela ordem de concordância que cada item obteve. A partir da Figura 4 pode-se observar que as DTs tiveram em sua maioria mais 90% de aprovação. Isto demostra a alto nível de concordância adquirida para a maioria das DTs deste questionário.

Assim como no questionário anterior, alguns participantes realizaram comentários quanto as DTs sugerindo, por exemplo, agrupar uma DT com outra (sugerindo que elas apresentam mesmo significado). Também houve sugestões para alterar a nomenclatura de determinada DT. Isto ocorreu, por exemplo, com a DT13 (Necessidade de refatorar os scripts de teste), em que um participante comentou: "Dívida Técnica: creio que pode ser alterado para Scripts de Teste Obsoletos".

A Figura 4 apresenta ainda a análise das Causas. As causas apresentaram uma maior variação de concordância, entre 62% e 91%, diferente das DTs que

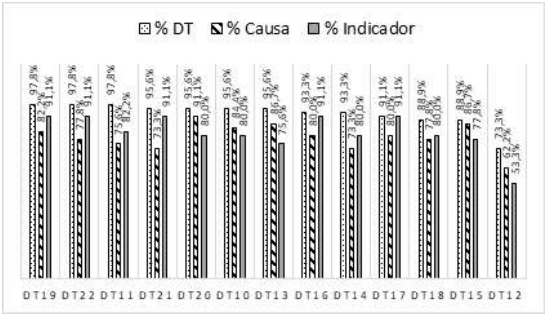

Fig. 4: Análise de DTs, Causas e Indicadores do Questionário "Projetar e Executar Testes".

apresentaram níveis de concordância com menor variação e maiores porcentagens. As causas deste questionário, assim como as do questionário anterior, apresentaram discordâncias que refletem à necessidade de acrescentar novas causas, ou ainda, retirar uma ou mais causas sugeridas. Por exemplo, referente as causas da DT18 (Sobrecarga de atividades em determinados períodos do projeto) um participante sugeriu a inclusão de uma nova causa: "Causas: Inexperiência deve ser incluído".

A Figura 4 apresenta também a concordância dos indicadores das DTs do questionário Projetar e Executar. Pode-se observar que os indicadores não apresentaram muita variação na porcentagem de concordância e mantiveram a maioria dos grupos de indicadores com aceite acima de 80%.

O grupo de indicadores que tiveram menor aprovação (53,33%, mais de 20% de distância do penúltimo grupo de indicadores) é associado à

DT12 (Falta de documentação dos testes que serão executados). O indicador da DT12 é: Quantidade de documentos. Alguns dos comentários referentes à DT12 indicaram que a quantidade de documentos pode não ser uma métrica adequada como indicador, vários participantes citam que a qualidade dos documentos reflete diretamente na eficiência dos mesmos, e seria um indicador mais adequado, isto pode refletir a baixa concordância para o indicador da DT12.

Os comentários sobre os indicadores foram em menor número, porém, manteve o mesmo padrão observado nos comentários referentes as causas, isto é, acrescentando novos indicadores, refutando alguns, ou ainda discordando de um ou outro indicador. Na próxima seção são apresentadas as conclusões e trabalhos futuros gerados a partir da execução desta pesquisa.

V. CONCLUSÕES

DT é um tema recente em Engenharia de Software, e apresenta a maioria dos trabalhos com ênfase na fase de codificação. Assim, são poucos os trabalhos relacionados diretamente a DT na área de Teste de Software. Este artigo buscou mapear possíveis DTs no processo de testes, com objetivo de servir como instrumento para facilitar a identificação das DTs e ajudar no entendimento de suas possíveis origens. Para isto, no mapeamento proposto foram relacionados indicadores e causas à cada DT. Os indicadores se apresentaram como recursos interessantes para que o gestor possa assumir medidas preditivas ao identificar um indicador eminente em uma DT. Já as causas ajudam no mapeamento da origem da DT facilitando a identificação de possíveis soluções. O artigo apresentou um survey no qual profissionais da área concordaram com as DTs, que foram retiradas de problemas que acontecem em teste de software. O trabalho apresentou 22 DTs com suas causas e indicadores. Como limitação, vale ressaltar que podem existir outros itens (DT, causa ou identificador) que não foram identificados para serem avaliados. Como trabalhos futuros pretende-se identificar possíveis soluções (a partir da literatura técnica) para cada uma das 22 DTs reportadas neste trabalho, avaliando-as com profissionais da área de teste de software, e, por fim, propor um mapa de apoio a gestão de DTs no processo de teste de software.

REFERENCES

[1] 1. R. S. Pressman. Software Engineering: a Practitioner's Approach, McGraw-Hill, 7 ed. 2011.

[2] W. Cunningham. The Wycash Portfolio Management System, In: ACM SIGPLAN OOPS Messenger (Vol. 4, No. 2). ACM. December 1992, p. 29-30.

[3] Z. Li, P. Avgeriou, P. Liang. A Systematic Mapping Stydy on Technical Debt and its Management. In: Journal Of Systems and Software. v. 101. Março de 2015. p. 1 – 272.

[4] A. Bertolino. Software Testing Research: Achievements, Challenges, Dreams. In: Future of Software Engineering, 2007. FOSE '07. 2007, Minneapolis, MN. 23 - 25 de mai. de 2007. p. 85–103.

[5] ISO/IEC/IEEE 29119-2. Software and Systems Engineering Software Testing - Part 2: Test Processes, International Organization of Standardization, 2013.

[6] T. Klinger. An Enterprise Perspective on Technical Debt. In: International Conference on Software Engineering. 2011, New York, NY, USA. ACM, 2011. p. 35–38.

[7] Y. Guo. Measuring and Monitoring Technical Debt. 4th International Doctoral Symposium on Empirical Software Engineering. [S.I.]:[s.n.].2009 p. 25-46.

[8] F. Shull. Perfectionists in a World of Finite Resources: IEEE Software, 2011, 28, p. 4 - 6

[9] N.S.R. Alves, L. F. Ribeiro, V. Caires, T. S. Mendes, R .O. Spínola. Towards an Ontology of Terms on Technical Debt. In: International Workshop on Managing Technical Debt, 6, 2014. p. 1-7

[10] A. Mettle, J. Hass. Testing Processes, In: Software Testing Verification and Validation Workshop, 2008. ICSTW '08. IEEE International Conference on, 2008 Lillehammer, Norway, Anais, Lillehammer, Norway, 9-11 de Abril de 2008, p. 321 - 327

[11] S. Shah; Torchiano, M.; Vetro, A.; Morisio, M.. Exploratory Testing as a Source of Testing Technical Debt. IT Professional, v. 16, ed. 3. p. 44 - 51. 7 de mar. de 2014

[12] R. Bavani. Distributed Agile: Agile Testing and Technical Debt. IEEE Software 29. Jun. de 2012. p. 28 – 33.

[13] K. Wiklund; Eldh, S.; Sundmark, Daniel; Lundqvist, K. et al. Tehnical Debt in Test Automation. In: Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on. 2012. p. 887 – 892.

[14] W. Snipes, B. Robinson, Y. Guo. C. Seaman. Defining the Decision Factors for Managing Defects: A Technical Debt Perspective, In: Proceedings of the 3rd International Workshop on Managing Technical Debt (MTD'12), IEEE, Zurich, Switzerland, 2012, p. 54 – 60.

[15] E. Dustin. Effective Software Testing: 50 Ways to Improve Your Software Testing. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. 2003.

[16] K. Naik, P. Tripathy. Software Testing and Quality Assurance: Theory and Practice. Hoboken, New Jersey: John Wiley Sons, Inc. 2008. 616 p.

[17] W. E. Perry. Effective Methods for Software Testing: Includes Complete Guidelines, Checklists, and Templates. 3 ed. Indianapolis, Indiana: Wiley Publishing, Inc. 2006. 973 p.

[18] J. Kasurinen, O. Taipale, K. Smolander. Analysis of Problems in Testing Practices, In: Software Engineering Conference, 2009. APSEC '09. Asia-Pacific 2009, Batu Ferringhi, Penang, Malaysia. 1 - 3 de Dezembro de 2009, p. 309 – 315.

[19] S. Amland. Risk-based testing: risk analysis fundamentals and metrics for software testing including a financial application case study. New York, NY, USA: Elsevier Science Inc. ed. 3, 2000. vol. 53.

[20] S. M. Quadri, S. U. Farooq. Software Testing: Goals, Principles, and Limitations. International Journal of Computer Applications, v. 6, n. 9, p. 7 - 10, 2010.

[21] J. W. Rittinghouse. Managing Software Deliverabels: A Software Development Management Methodology. Digital Press, 2004. p. 111–133.

[22] D. Lorenzoli, L. Mariani, M. Pezze. Automatic generation of software behavioral models. In: International Conference on Software Engineering, 30, 2008. p. 501-510.

[23] G. J. Myers, T. Badgett, C. Sandler. The Art of Software Testing. ed. 3. Canadá: John Wiley Sons, Inc, 2012. 240 p.
[24] A. I. BaarsT. E. J. Vos, D. M. Dimitrov. Using Evolutionary Testing to Find Test Scenarios for Hard to Reproduce Faults, In: Third International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), 2010, Paris, França, p. 173 – 181.
[25] B. Jiang; Zhenyu Zhang; Tse, T.H.; Chen, T.Y. et al. How well does test case prioritization integrate with statistical fault localization?. Information and
[26] Software Technology, v 54, ed. 7, p. 739 - 758, 2012.
[27] E. Aranha, P. Borba. An Estimation Model for Test Execution Effort, In: Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on, 2007, Madri, 2007, 20 - 21 de Setembro de 2007, p. 107 – 116.
[28] M. Hamburg. Basic Statistics: A Modern Approach, Journal of the Royal Statistical Society. Series A (General), v. 146, no. 1, ed. 2.

CLEYDIANE LIMA DE SOUSA Mestrado pela Universidade Federal de Pernambuco (2016), com o tema de dissertação: Dívida Técnica no Processo de Teste de Software. Graduação em Sistemas de Informação pelo Centro Universitário Luterano de Palmas (2012) Trabalha como Engenheira de Testes do Centro de Estudos e Sistemas Avançados do Recife (CESAR). Casada com automação de testes de contrato, API, componentes e e2e. Brinca com testes de carga no Artillery. Ama tartarugas. Maranhense morando em Recife. Estudante de fotografia nas horas vagas.

ARILO CLAUDIO DIAS NETO Possui graduação em Ciência da Computação pela Universidade Federal do Amazonas (2004), mestrado em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro (2006), doutorado em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro (2009). É Professor Adjunto da Universidade Federal do Amazonas, coordenador do Grupo de Pesquisa de Experimentação e Teste de Software. Atua desde 2017 como CPO (Chief Product Officer) do Méliuz.