
Universidade Federal da Bahia
Instituto de Matemática

Programa de Pós-Graduação em Ciência da Computação

# SUSTENTABILIDADE TÉCNICA DE SOFTWARE ACADÊMICO NO DOMÍNIO DE FERRAMENTAS DE ANÁLISE ESTÁTICA

Joenio Marques da Costa

DISSERTAÇÃO DE MESTRADO

Salvador
19 de Dezembro de 2017

JOENIO MARQUES DA COSTA

# SUSTENTABILIDADE TÉCNICA DE SOFTWARE ACADÊMICO NO DOMÍNIO DE FERRAMENTAS DE ANÁLISE ESTÁTICA

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Christina von Flach G. Chavez
Co-orientador: Paulo Roberto Miranda Meirelles

Salvador
19 de Dezembro de 2017

Sistema de Bibliotecas - UFBA

Costa, Joenio Marques da.
Sustentabilidade técnica de software acadêmico no domínio de ferramentas de análise estática / Joenio Marques da Costa – Salvador, 2017.
91p.: il.

Orientadora: Prof. Dr. Christina von Flach G. Chavez.
Co-orientador: Prof. Dr. Paulo Roberto Miranda Meirelles.
Dissertação (Mestrado) – Universidade Federal da Bahia, Instituto de Matemática, 2017.

1. Software Acadêmico. 2. Sustentabilidade Técnica. 3. Evolução de Software. 4. Reprodutibilidade. 5. Publicização de Software. 6. Reconhecimento de Software. . I. Chavez, Christina von Flach Garcia. II. Meirelles, Paulo Roberto Miranda. III. Universidade Federal da Bahia. Instituto de Matemática. IV. Título.

CDD – 005.1
CDU – 004.41

# TERMO DE APROVAÇÃO

## JOENIO MARQUES DA COSTA

## SUSTENTABILIDADE TÉCNICA DE SOFTWARE ACADÊMICO NO DOMÍNIO DE FERRAMENTAS DE ANÁLISE ESTÁTICA

Esta Dissertação de Mestrado foi julgada adequada à obtenção do título de Mestre em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia.

Salvador, 19 de Dezembro de 2017

---

Profa. Dra. Christina von Flach G. Chavez
Universidade Federal Bahia

---

Prof. Dr. Sandro Santos Andrade
Instituto Federal da Bahia

---

Prof. Dr. Rodrigo Rocha Gomes e Souza
Universidade Federal da Bahia

*Dedico este trabalho ao meu pai Jonas Cândido da Costa, minha mãe Neirimar Marques da Costa e ao meu irmão Jonei Marques da Costa.*

# AGRADECIMENTOS

Agradeço as amigas Daniela Soares Feitosa, Fabiana Goa, Hilmer Rodrigues Neri, Lucas Kanashiro Duarte e Marcos Ronaldo Pereira Júnior pelas correções, idéias e dicas valiosas ao texto.

À Mariel Rosauro Zasso pelas inúmeras tardes de estudo e trabalho em colaboração, contribuindo para o avanço do trabalho e para a manutenção do meu foco no estudo, por vezes difícil de manter.

Aos professores do Programa de Pós-Graduação em Ciência da Computação (PGCOMP-UFBA) Cláudio Nogueira Sant'Anna e Ivan do Carmo Machado pelo ótimo *feedback* e sugestões de melhorias a toda a pesquisa.

A todos os amigos, mestrandos e doutorandos, do PGCOMP-UFBA pela troca de conhecimento e apoio, em especial ao amigo Crescencio Rodrigues Lima Neto, pelas ótimas reflexões a respeito do tema desta pesquisa.

A minha irmã Talita Fernanda Gentil pelo apoio e descontração em todos os momentos, exceto na defesa por não estar presente $\neg\neg$.

Aos meus grandes amigos e parceiros de lar José Flávio Fernandino Maciel e Paulo Alexandre Elias Passos por me cuidarem em diversos momentos de isolamento e concentração necessários para a conclusão desta pesquisa.

Ao amigo e ex-sócio de Colivre Antonio Soares de Azevedo Terceiro por ter me incentivado a entrar no programa de mestrado do PGCOMP-UFBA.

À minha grande amiga, Janaína Santos Valente por participar pacientemente com todo apoio desde o início desta pesquisa, e pelo grande apoio, recepção e carinho, especialmente nos últimos meses de conclusão deste trabalho.

Agradeço também à banca de defesa Sandro Santos Andrade e Rodrigo Rocha Gomes e Souza pela atenção dada ao trabalho e pelas dicas valiosas para melhoria da qualidade final do estudo.

À minha professora e orientadora Christina von Flach Garcia Chavez pelo cuidado, pela orientação e pela liberdade dada nos rumos desta pesquisa.

Ao meu amigo e co-orientador Paulo Roberto Miranda Meirelles pela energia investida, me fazendo continuar e seguir em frente, chegando até aqui.

Obrigado a todos!

*Ocorrências não reprodutíveis não têm significado para a Ciência.*

—POPPER, KARL R. (1959)

# RESUMO

O uso crescente de software acadêmico, isto é, software desenvolvido para apoiar pesquisas científicas em diversas áreas do conhecimento, tem feito a Ciência moderna depender da sustentabilidade técnica do software. O desenvolvimento não sustentável de software acadêmico pode ferir um dos fundamentos da Ciência: a reprodutibilidade, ou a capacidade de reprodução de estudos científicos por pesquisadores independentes. Além disso, o desenvolvimento não sustentável de software acadêmico em um domínio pode levar a um quadro de desordem caótica disfuncional (DCD), caracterizado pela existência de muitos projetos similares, com poucos usuários e ciclos de vida curtos, e que terminam em paralelo ao financiamento inicial, comunidades desconectadas e paralelas, incompatibilidade entre projetos e tentativas aparentemente não coordenadas de "reiniciar" tudo. No entanto, não há estudos sobre sustentabilidade técnica ou DCD em software acadêmico da área de Engenharia de Software, especialmente no domínio de análise estática, com uma longa tradição no desenvolvimento de ferramentas para apoiar pesquisas em diferentes áreas. O objetivo geral desta pesquisa de mestrado foi analisar projetos de software acadêmico de análise estática com o propósito de caracterizar sua sustentabilidade técnica, com respeito a publicização, reconhecimento e ciclo de vida, na perspectiva do cientista – desenvolvedor ou usuário – de software acadêmico no contexto das conferências ASE (*Automated Software Engineering*) e SCAM (*Working Conference on Source Code Analysis & Manipulation*). O software acadêmico publicado nessas conferências foi objeto de uma pesquisa documental, realizada com base em código-fonte, manuais e repositórios. Uma revisão da literatura foi realizada nas bases da ACM e IEEE para a caracterização do reconhecimento do software acadêmico em termos de tipos e número de menções feitas por outros artigos científicos e contribuições em seu código-fonte. Para software acadêmico com código-fonte disponível, foi realizada a caracterização de seu ciclo de vida, com base no número de módulos e no número de lançamentos. Foram encontrados 60 projetos de software acadêmico de análise estática publicados em artigos da ASE e SCAM. A caracterização de sua sustentabilidade técnica mostrou que: 40% não está disponível publicamente, ou seja, não é possível obter o software na URL informada pelos autores, dificultando a reprodução de estudos que tenham usado tal software; 23% não possui outra menção nas bases ACM e IEEE além da publicação original do software; e 30% recebeu contribuição em código-fonte. Pôde-se observar alguns indícios de DCD: existência de muitos projetos de software acadêmico de análise estática com poucos usuários, e ciclos de vida curtos. 78% dos projetos de software acadêmico de análise estática estão em estado inicial de desenvolvimento, descontinuado ou encerrado.

**Palavras-chave:** Software acadêmico, sustentabilidade técnica, evolução de software, reprodutibilidade, publicização de software, reconhecimento de software.

# ABSTRACT

The increasing adoption of academic software, the software designed to support scientific research in various areas of knowledge, has made the modern Science depends on the technical sustainability of software. Unsustainable development of academic software makes it dificult one of the Science foundations: the reproducibility, or the reproduction's capacity of scientific studies by independent researchers. In addition, non-sustainable development of academic software can lead to a "dysfunctional chaotic churn" - DCC, characterized by the existence of many similar projects, with few users and short life cycles, ending in parallel with the initial funding, disconnected and parallel communities, incompatibility between projects, and seemingly uncoordinated attempts to "reboot" everything. However, there are no studies on technical sustainability or DCC in academic software of the Software Engineering field, especially in the field of static analysis, with a long tradition in the development of tools to support research in different areas. The overall objective of this master's research was to analyze the static analysis software with the purpose of characterizing its technical sustainability, with respect to publicity, recognition and life cycle, from the perspective of the scientist – developer or user – of academic software in the context of ASE (*Automated Software Engineering*) and SCAM (*Working Conference on Source Code Analysis & Manipulation*). The academic software published at these conferences was the object of a documentary research, carried out based on source code, manuals and repositories. A literature review was carried out at ACM and IEEE for the characterization of academic software recognition in terms of the types and number of mentions made by other scientific articles, including contributions in its source code. For academic software with source code available, we carried out the characterization of its life cycle, based on the number of modules and the number of releases. We found 60 projects published in ASE and SCAM conferences. The characterization of its technical sustainability showed that: 40% is not publicly available, it is not possible to obtain the software in the URL informed by the authors, making it hard to reproduce of studies that have used such software; 23% has no mentions in the ACM and IEEE besides those made in the original publication of the software; and 30% received contribution in source code. We could observe some evidence of DCC: existence of many academic software projects of static analysis with few users, and short life cycles. 78% of the static analysis academic software projects are in the initial state of development, discontinued or terminated.

**Keywords:** Academic software, technical sustainability, software evolution, reproducibility, publicization of software, recognition of software.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

Capítulo 1

# INTRODUÇÃO

A Ciência moderna depende de software. Software resolve problemas comuns de pelo menos metade dos cientistas de todas as áreas do conhecimento (WILSON et al., 2014). Cientistas não apenas utilizam software para fazer Ciência como são também os seus principais desenvolvedores (GOBLE, 2014).

Entretanto, ao desenvolver software, pesquisadores raramente publicam o seu código-fonte (ROBLES, 2010; AMANN et al., 2015), ferindo um dos princípios da Ciência, de que novas descobertas sejam reproduzidas antes de serem consideradas parte da base de conhecimento (STODDEN, 2009). Além de desacelerar o progresso geral da Ciência, a indisponibilidade do software acadêmico pode inviabilizar a replicação, considerado um método comum e cientificamente produtivo de produzir conhecimento novo a partir de pesquisas anteriores (KING, 1995; STODDEN, 2010), gerando retrabalho e consumindo de maneira ineficiente os limitados recursos da Ciência (HOWISON; HERBSLEB, 2013; KATZ, 2014).

Este cenário remete ao fenômeno de desordem caótica disfuncional - DCD ( *"dysfunctional chaotic churn"*) do software acadêmico mencionado por Howison et al. (2015), caracterizado pela existência de muitos projetos, com poucos usuários, com ciclos de vida curtos, que terminam em paralelo ao financiamento inicial, comunidades desconectadas e paralelas, incompatibilidades entre projetos, e tentativas aparentemente não coordenadas de "reiniciar" tudo (*re-boots*) (HOWISON et al., 2015).

Como resultado da indisponibilidade do software acadêmico, dados são perdidos, análises levam mais tempo que o necessário e os pesquisadores não conseguem a eficiência que poderiam ter ao trabalhar com software (WILSON et al., 2017), ocasionando graves erros em conclusões centrais da Ciência (MERALI, 2010).

Isto tem motivado a realização de conferências específicas sobre o tema, como por exemplo o *RSE (Conference of Research Software Engineers)*[1], *WSSSPE (Workshop on Sustainable Software for Science: Practice and Experiences)*[2] e *RESER (Workshop on Replication in Empirical Software Engineering Research)*[3]. Estes fóruns têm contribuído

[1] ⟨http://rse.ac.uk/conf2017⟩
[2] ⟨http://wssspe.researchcomputing.org.uk⟩
[3] ⟨http://sequoia.cs.byu.edu/reser⟩

para a compreensão dos problemas, abordando questões sobre desenvolvimento, qualidade e sustentabilidade, sobre como citar software, como promover e reconhecer o papel do pesquisador engenheiro desenvolvedor de software (*Research Software Engineer*), além de questões sobre infraestrutura, ferramentas e práticas para o desenvolvimento de software de forma sustentável.

O desenvolvimento sustentável de software tem sido alvo de intenso debate e pouco consenso. Sustentabilidade é um tema multidisciplinar, sistêmico e com múltiplas dimensões: individual, social, econômica, ambiental e técnica (BECKER et al., 2014). A dimensão técnica, por exemplo, diz respeito à capacidade do software de perdurar e de continuar sendo suportado ao longo do tempo, o que implica em qualidades de longevidade e manutenibilidade (VENTERS et al., 2014).

Na dimensão técnica, o desenvolvimento de software para a Ciência, ou seja, o desenvolvimento de *software acadêmico*, de forma sustentável, abre portas para elevar a qualidade geral do software e da própria pesquisa científica, proporcionando um ambiente de compartilhamento e colaboração em oposição ao tradicional modelo de competição que permeia o sistema de reputação e crédito científico.

Dessa forma, neste trabalho investigamos como o software acadêmico do domínio de análise estática, desenvolvido e publicado em duas importantes conferências de Engenharia de Software, *ASE (Automated Software Engineering)*[4] e *SCAM (Working Conference on Source Code Analysis & Manipulation)*[5], se apresenta em termos de sustentabilidade técnica, identificando problemas e abrindo caminho para a melhoria geral da qualidade das ferramentas e das pesquisas neste campo.

## 1.1 ESCOPO

### 1.1.1 Objetivo Geral

Este trabalho teve como objetivo geral analisar os *projetos de software acadêmico de análise estática e sua sustentabilidade técnica* com o propósito de *caracterizá-los* com respeito a *publicização, reconhecimento e ciclo de vida* na perspectiva do *cientista – desenvolvedor ou usuário – de software acadêmico* no contexto das *conferências de Engenharia de Software ASE e SCAM*.

### 1.1.2 Objetivos Específicos

São objetivos específicos deste trabalho:

**01** Caracterizar o software acadêmico de análise estática com respeito à sua publicização. A caracterização será feita em um conjunto de projetos de software acadêmico de análise estática, com base em medidas para avaliar a sua disponibilidade de *download*, código-fonte, e presença oficial *online*.

**02** Caracterizar o software acadêmico de análise estática com respeito ao reconhecimento na literatura acadêmica. A caracterização será feita em um conjunto de projetos

[4] ⟨http://ase-conferences.org⟩
[5] ⟨http://www.ieee-scam.org⟩

de software acadêmico de análise estática, com base em uma análise de trabalhos científicos que os utilizam ou adaptam.

**03** Caracterizar o software acadêmico de análise estática com respeito ao seu ciclo de vida. A caracterização será feita em um conjunto de projetos de software acadêmico de análise estática, com base nas informações sobre lançamentos (*releases*) e métricas de código-fonte.

### 1.1.3 Questão de Pesquisa

Neste estudo a seguinte questão de pesquisa, a respeito do software acadêmico de análise estática, será investigada:

**Questão:** Como a desordem caótica disfuncional (DCD) pode explicar a sustentabilidade técnica dos projetos de software acadêmico de análise estática em termos de publicização, reconhecimento e estágio de evolução?

### 1.1.4 Métricas

Para responder à questão de pesquisa, as seguintes métricas foram usadas:

1. Métricas relacionadas à publicização do software (número de projetos disponíveis para *download*, com código-fonte disponível, tipo de licença);
2. Métricas relacionadas ao reconhecimento do software (número de citações, número de menções, número de usos e contribuições);
3. Métricas relacionadas ao ciclo de vida do software (número total de lançamentos, data e número de versão de cada lançamento, variação no número de módulos do código-fonte).

## 1.2 ESTRATÉGIA DE PESQUISA

Este trabalho apresenta um estudo de caso exploratório (*exploratory case study*) sobre a sustentabilidade técnica do software acadêmico de análise estática. O estudo adotou uma estratégia de pesquisa de trabalho de campo (*field studies*) em ambiente natural (*natural settings*) (STOL; FITZGERALD, 2015), com as seguintes características principais:

- Com o foco num fenômeno, organização ou sistema em particular;
- Com um baixo nível de generalização e alto realismo do contexto;
- Sem intervenção do pesquisador no ambiente.

Foram realizados três estudos sobre o software acadêmico de análise estática em termos de sua publicização, reconhecimento e ciclo de vida.

No primeiro estudo, sobre publicização, um conjunto de projetos de software publicados nas conferências de Engenharia de Software ASE e SCAM foi selecionado. Os projetos

foram caracterizados em termos de disponibilidade de *download*, código-fonte, forma de distribuição e licença.

A escolha destas duas conferências baseou-se no princípio de serem conferências tradicionais e importantes para a área de Engenharia de Software, tendo tradição em estudos sobre análise de software e, dessa forma, potencializando aumentar o número de projetos de software acadêmico de análise estática encontrados entre suas publicações.

No segundo estudo, sobre reconhecimento, os projetos selecionados pelo primeiro estudo foram caracterizados em relação ao reconhecimento acadêmico em termos de menções encontradas em publicações nas bases ACM e IEEE. As menções encontradas foram classificadas por tipo e cada artigo mencionando o software selecionado foi caracterizado segundo os tipos encontrados.

No terceiro estudo, sobre ciclo de vida, os projetos de software acadêmico foram caracterizados em relação ao estágio de ciclo de vida em que se encontram, em termos de lançamentos, versões, disponibilidade de download, disponibilidade de código-fonte, e número de módulos para cada versão com código-fonte disponível.

## 1.3 ORGANIZAÇÃO DO TEXTO

O Capítulo 2 apresenta os fundamentos teóricos necessários para a compreensão deste trabalho. O Capítulo 3 traz um estudo sobre a publicização de software acadêmico de análise estática nas conferências de Engenharia de Software ASE e SCAM. O Capítulo 4 descreve um estudo sobre o reconhecimento de software acadêmico de análise estática em publicações nas bases ACM e IEEE. O Capítulo 5 apresenta um estudo sobre o estágio de evolução e o ciclo de vida de software acadêmico de análise estática. O Capítulo 6 discute os resultados em termos da sustentabilidade técnica do software acadêmico de análise estática e traça paralelos com trabalhos relacionados. O Capítulo 7 apresenta as considerações finais da pesquisa e aponta trabalhos futuros.

Capítulo

2

# FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta conceitos necessários para a compreensão do trabalho acerca de ecossistema de software (Seção 2.1), ecossistemas de software acadêmico (Seção 2.2), modelo de desenvolvimento de software acadêmico (Seção 2.3), software de análise estática (Seção 2.4), sustentabilidade (Seção 2.5), modelo de estágios para ciclo de vida do software (Seção 2.6), reprodutibilidade (Seção 2.7) e Ciência Aberta (Seção 2.8).

## 2.1 ECOSSISTEMA DE SOFTWARE

Ecossistema de software é definido, segundo Manikas e Hansen (2013), como a interação entre diversos atores numa plataforma tecnológica comum, resultando em novas soluções de software ou novos serviços. Atores do ecossistema, motivados por um conjunto de interesses, conectam-se entre si e ao próprio sistema numa relação simbiótica, fazendo a plataforma tecnológica evoluir enquanto permite o envolvimento e contribuição de novos e diferentes atores (MANIKAS; HANSEN, 2013).

Nessa relação, os atores são beneficiados de formas diferentes a depender da natureza do ecossistema. Num ambiente comercial, por exemplo, os atores são beneficiados diretamente através de receita financeira (salário, prêmios etc), enquanto num sistema não-comercial os atores estão motivados por questões não-monetárias, como fama, reconhecimento, ideologia etc (MANIKAS; HANSEN, 2013).

De modo geral, em ecossistemas de software, os benefícios recebidos pelos atores e proporcionados pelo ecossistema aumentam com o passar do tempo por meio de uma relação de benefício mútuo. Este modelo geral de funcionamento, no entanto, pode variar a depender do contexto em que se insere o ecossistema, especialmente no relacionamento entre os atores que pode variar entre mutualismo quando o relacionamento é de mútuo interesse, parasitismo quando um ator está sendo prejudicado na relação, antagonismo quando atores estão em competição direta ou neutralismo quando o relacionamento não afeta os atores diretamente (MANIKAS; HANSEN, 2013).

## 2.2 ECOSSISTEMA DE SOFTWARE ACADÊMICO

O ecossistema de software acadêmico possui a particularidade de se relacionar com o sistema econômico de reputação científica, especialmente com o seu modelo de publicações, influenciando e sendo influenciado diretamente pelo impacto de suas publicações (HOWISON et al., 2015).

Neste cenário, interessado em compreender as relações neste ecossistema, Howison et al. (2015) criou um framework para pensar e refletir sobre o processo de produção de software no meio científico, e identificou quatro papéis básicos assumidos pelos atores envolvidos no ecossistema de software acadêmico: (1) cientistas usuários finais, (2) produtores e distribuidores de software, (3) administradores de infraestrutura e (4) pesquisadores preocupados com o funcionamento do ecossistema como um todo.

### 2.2.1 Cientistas usuários finais

Cientistas ocupam papel chave no ecossistema de software acadêmico. Em seus processos de investigação e experimentação, fazem uso crescente de software para coleta, gerenciamento, transformação, análise, modelagem e visualização de dados. Além da preocupação com qualidade e usabilidade, estes cientistas estão também preocupados com a disponibilidade e com a capacidade do software em continuar sendo útil (HOWISON et al., 2015).

Eles estão interessados também em saber o que outros cientistas estão usando em suas pesquisas. A alta adoção de um software, além de ser um bom indicador de qualidade, mantém os cientistas mais livres e com maior foco em suas próprias pesquisas, uma vez que podem encontrar ajuda entre os seus pares para resolver questões sobre o uso do software (HOWISON et al., 2015).

### 2.2.2 Produtor e distribuidor de software acadêmico

O papel de produtor e distribuidor de software costuma ser desempenhado por equipes colaborando entre si, geralmente compostas por cientistas da computação e cientistas do domínio onde a pesquisa se insere. Geralmente, o cientista da computação é o responsável por implementar os algoritmos, métodos ou resultados gerados pelo estudo (HOWISON et al., 2015).

Um desafio comum enfrentado neste papel é conseguir abstrair os problemas e implementar soluções abrangentes em software. Muitas vezes, o software criado fica confinado em seu laboratório ou grupo, mas eventualmente é compartilhado e amplamente adotado, tornando o cientista autor do software e da pesquisa parte do seu ecossistema (HOWISON et al., 2015).

Entre as inúmeras preocupações do produtor e distribuidor de software, podemos destacar a preocupação acerca de como o software contribui para as pesquisas científicas que outros pesquisadores estão realizando. Alguns projetos são gerenciados no estilo de código aberto e têm atraído com sucesso contribuições de muitos cientistas, incluindo contribuidores que fazem pequenas, porém substanciais contribuições (HOWISON et al., 2015).

### 2.2.3 Provedor de infraestrutura

O provedor de infraestrutura é aquele que provê conjuntos de software aos cientistas usuários finais. Este conjunto pode estar disponível em forma de download para que seja utilizado em computadores pessoais ou pode estar disponível em forma de serviços, como por exemplo, ciberinfraestrutura de software, envolvendo muitas vezes soluções de hardware e rede além do próprio software (COUNCIL, 2007; STEWART; ALMES; WHEELER, 2010).

Do ponto de vista do ecossistema, estes dois tipos de distribuição implicam nas mesmas preocupações e questões: quem usa o software? qual versão é utilizada? qual a frequência de atualização? entre outras (HOWISON et al., 2015).

### 2.2.4 Pesquisador

Este último papel, chamado de pesquisador num sentido amplo, refere-se a qualquer pessoa preocupada com o funcionamento do ecossistema e com a sua contribuição para a Ciência. O papel de pesquisador costuma ser desempenhado por agências de fomento ou por cientistas preocupados com o seu trabalho individual e com o impacto em seu campo de pesquisa (HOWISON et al., 2015).

As preocupações incluem questões sobre a operação do ecossistema como um sistema que consome recursos (tempo, dinheiro e atenção) e afeta a conduta da Ciência, tanto no geral como em campos específicos e sobre a compreensão do comportamento desse sistema e de como pode ser influenciado (HOWISON et al., 2015).

## 2.3 MODELO DE DESENVOLVIMENTO DE SOFTWARE ACADÊMICO

Cada ator desempenha um papel importante na estabilidade e sustentabilidade do ecossistema (DHUNGANA et al., 2010). Assim como nos ecossistemas naturais, o ecossistema de software necessita de fornecimento constante de energia, seja na forma de novos desenvolvimentos ou na forma de ações de manutenção (DHUNGANA et al., 2010).

Os atores participam dentro de seus próprios interesses, mas sempre causando impacto no sistema como um todo (MANIKAS; HANSEN, 2013). Cientistas usuários finais usam software acadêmico (direta ou indiretamente) para fazer Ciência, resultando em impacto científico. Tal impacto científico justifica novos investimentos, fazendo o ecossistema crescer (HOWISON et al., 2015).

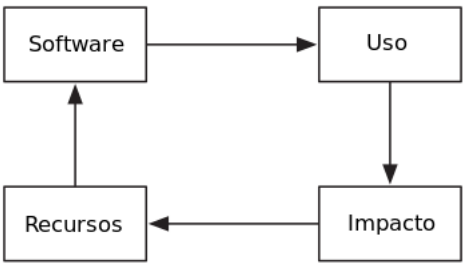



**Figura 2.1** Um modelo de processo de software na Ciência (HOWISON et al., 2015)

A Figura 2.1 apresenta um modelo de desenvolvimento de software acadêmico, explicitando as relações entre os elementos *software*, *recursos*, *uso* e *impacto*, detalhados a seguir.

### 2.3.1 Software acadêmico

Software acadêmico (*academic software*) é todo software usado para coletar, processar ou analisar resultados de pesquisas com intenção de publicação na literatura acadêmica (periódicos, revistas, conferências, monografias, livros ou teses), incluindo desde protótipos escritos pelos próprios cientistas, a produtos completos desenvolvidos profissionalmente (ALLEN et al., 2017).

Podem ser projetos de software desenvolvidos num modelo de **Software como serviço de suporte** (*Software as Supporting Service*), sobrevivendo totalmente à parte do sistema de reputação acadêmica, ou no modelo de **Software para crédito acadêmico** (*Software for academic credit*), estando seu desenvolvimento intrinsecamente associado ao sistema de reputação acadêmica (HOWISON; HERBSLEB, 2011).

No segundo caso, Software para crédito acadêmico, a relação com o sistema de reputação acadêmica pode ainda variar entre motivações distintas resultando em (1) Software Incidental (*Incidental software*) feito puramente para apoiar e facilitar pesquisas, (2) Prática de software paralela (*A parallel software practice*) feito com objetivo de ser utilizado por outros pesquisadores, ou (3) Um subcampo de software (*A Software Subfield*), onde o próprio software é considerado uma contribuição primária para a Ciência (HOWISON; HERBSLEB, 2011).

### 2.3.2 Recursos

Os recursos investidos na produção de software acadêmico vêm de diversas fontes, incluindo ganhos monetários diretos, recursos alocados em projetos e colaboração entre laboratórios de pesquisa (HOWISON et al., 2015). Grande parte dos recursos vem do "tempo livre" dos pesquisadores em busca de soluções para suas pesquisas e perpassa por financiamentos diversos obtidos na carreira individual do cientista, prêmios recebidos, etc (HOWISON et al., 2015).

Independente da origem dos recursos, grande parte do desenvolvimento de software acadêmico é realizado pelos próprios cientistas (HETTRICK et al., 2014; MOMCHEVA; TOLLERUD, 2015). Esta tendência tem sido interpretada como um reflexo do conhecimento sobre o domínio da pesquisa muitas vezes necessário ao ator desenvolvedor do software (SEGAL; MORRIS, 2008).

Nas pesquisas em Engenharia de Software, este conhecimento teórico sobre o domínio da pesquisa se confunde, muitas vezes, com a própria prática de desenvolvimento de software, especialmente no domínio de análise estática, uma área intimamente ligada às pesquisas sobre a construção de software, como linguagens e compiladores, por exemplo.

### 2.3.3 Uso

Metade dos pesquisadores de todas as áreas da Ciência fazem uso intenso de software acadêmico, desde grupos trabalhando exclusivamente com problemas computacionais até grupos em laboratórios tradicionais ou em campo (WILSON et al., 2014).

Este uso é mencionado na literatura acadêmica por meio de citação formal ou informal (SMITH; KATZ; NIEMEYER, 2016) e está estreitamente relacionado ao sistema econômico de reputação científica, uma vez que menções causam impacto científico direto tanto na publicação quanto no ecossistema de software acadêmico (KATZ, 2014).

Este impacto direto geralmente justifica o investimento de novos recursos no ecossistema, seja para fins de planejamento, como por exemplo numa retrospectiva para avaliar investimentos já realizados, ou para fins de promoção e evolução do software acadêmico (HOWISON et al., 2015).

### 2.3.4 Impacto científico

Ao longo da história, a citação formal tem sido utilizada para garantir autenticidade e autoridade, ao invés de crédito e reconhecimento (KATZ, 2014). Na história ocidental, a citação surge no final do século XVI e, no início do século XVIII surge o sistema legal por trás do sistema de citações e a lei de "copyright" para garantir os direitos dos autores (KATZ, 2014).

Apesar desse uso histórico para fins de autenticidade e autoridade, o sistema de citações e as informações de autoria das publicações têm sido utilizados para avaliações importantes dentro do corpo científico (KATZ, 2014). Por exemplo, para realizar "backward citing" o sistema tem sido utilizado para se certificar quem de fato contribuiu para um certo avanço ou descoberta, e "forward citing" tem sido usada em casos onde se quer entender como uma ideia foi usada após o seu surgimento ou publicação (KATZ, 2014).

Conhecimento novo é claramente construído a partir do conhecimento passado e o sistema de citações formais tem promovido avanços significativos (KATZ, 2014). No entanto, esse conceito não tem funcionado tão bem para produtos digitais como o software, que muitas vezes depende de outro software, fragmentos de código, e algoritmos (KATZ, 2014).

Este debate ocorre há bastante tempo entre as diversas áreas da bibliometria, cienciometria, altmetria e áreas similares (GOUVEIA et al., 2013). O fator de impacto, por exemplo, proposto na década de 90 (REUTERS, 2017), apesar de contribuir para a Ciência, por vezes é utilizado da forma errada e mostra deficiências ao lidar com produtos digitais gerados em pesquisas (KATZ, 2014).

Tradicionalmente, autores citam artigos adicionando referência para o autor, título, local de publicação etc. Entretanto, este conceito não funciona bem para produtos digitais como o software por exemplo, que frequentemente depende de bibliotecas, fragmentos de código e outros algoritmos. Para muitos desses, o identificador a ser citado – um "nome" que faz referência para um único produto – não é claro. Adicionalmente, se uma biblioteca citada depende de outra biblioteca, a contribuição desta segunda biblioteca não é capturada (KATZ, 2014).

## 2.4 ECOSSISTEMA DE SOFTWARE ACADÊMICO DE ANÁLISE ESTÁTICA

Ao falarmos sobre ecossistema de software acadêmico estamos nos referindo a qualquer software, de qualquer domínio de aplicação, que tenha sido utilizado ou produzido durante trabalhos de pesquisa com intuito de publicação na literatura acadêmica.

O ecossistema de software acadêmico de análise estática é um recorte deste conjunto, a princípio, com as mesmas características, atores e modelo de funcionamento, mas logicamente podendo apresentar particularidades trazidas pela natureza do domínio de análise estática e suas ferramentas, soluções e algoritmos.

### 2.4.1 Análise estática

A análise estática de código-fonte é o primeiro passo para coletar informações necessárias em diversas atividades de verificação, medição e melhoria da qualidade de produtos de software (CRUZ; HENRIQUES; PINTO, 2009; KIRKOV; AGRE, 2010). Ela é realizada com base no código-fonte de um programa ou sistema de software, e a partir daí descobre problemas e propriedades de sua qualidade estrutural (CHESS; WEST, 2007).

Ferramentas de análise estática estão disponíveis há décadas, em especial, para programadores. A ferramenta Lint (JOHNSON, 1978), considerada uma das primeiras ferramentas de análise estática (GOSAIN; SHARMA, 2015), foi criada para examinar programas escritos em linguagem C e aplicar regras de tipagem mais estritas do que as regras dos próprios compiladores da linguagem.

Análise estática de código-fonte tem como objetivo prover informações acerca de um programa a partir do seu código-fonte sem necessidade de execução, e sem requerer qualquer outro artefato do programa além do próprio código.

É um ramo que possui muitas das suas abordagens em comum com os estudos da área de análise de programas (*program analysis*), especialmente na área de compiladores, onde atua especialmente nas primeiras etapas do processo de compilação.

A análise estática de código-fonte é considerada uma atividade meio com objetivo de suportar uma variedade de tarefas comuns da Engenharia de Software; muitas dessas tarefas são substancialmente úteis em atividades de manutenção, como por exemplo, análise de performance, compreensão de programas, desenvolvimento baseado em modelos, detecção de clones, evolução de software, garantia de qualidade, localizaçao de falhas, manutenção de software, recuperação arquitetural e testes (BINKLEY, 2007).

Seja em qual atividade for, a análise estática possui importância significativa, pois ao ser capaz de extrair informações diretamente do código-fonte de um programa, pode auxiliar a responder perguntas necessárias para as diversas atividades de desenvolvimento e evolução de software. Essa importância se torna ainda mais aparente diante da "lei" da tendência para execução que indica que todos os tipos de notação têm a tendência de se tornar executáveis. Cada nova notação, seja de baixo nível (e, portanto, já código-fonte) ou de alto nível (podendo escapar do nível de código) irá em última análise se tornar código-fonte (HARMAN, 2010).

A análise de programas trata, de modo geral, da descoberta de problemas e fatos sobre programas. Tal análise pode ser realizada sem a necessidade de executar o programa

(análise estática) ou com informações provenientes de sua execução (análise dinâmica).

A ideia de que programas de computador podem ser utilizados para analisar código-fonte de outros programas tem uma história de mais de 40 anos. O programa *PFORT Verifier* (RYDER, 1974) foi projetado para localizar potenciais problemas na portabilidade de código Fortran; em função da diversidade de dialetos de Fortran, uma compilação sem erros não indicava que o programa estava correto segundo os padrões da linguagem (WICHMANN et al., 1995).

Desde então, ferramentas de análise estática de código-fonte têm surgido para os mais diversos fins – muitas delas a partir das pesquisas e desenvolvimentos da área de compiladores. O *parser* utilizado nessas ferramentas têm funcionalidades análogas aos analisadores usados em compiladores (ANDERSON, 2008).

O uso de tais ferramentas tem se tornado mais comum no ciclo de desenvolvimento de software, sendo aplicadas em atividades distintas. O campo de aplicação destas ferramentas é bastante variado, cobrindo diferentes objetivos.

### 2.4.2 Software de análise estática

A variedade de aplicação e a constante evolução da área de análise estática, tanto na indústria quando na academia, resulta em estudos teóricos e práticos, novas ferramentas, modelos e algoritmos de análise estática. Ferramentas de análise estática têm sido continuamente desenvolvidas e seu uso se tornado comum no ciclo de desenvolvimento de software.

Mas apesar da rápida e constante evolução da área, ainda há carência de estudos avaliando estas ferramentas (LI; CUI, 2010). Mesmo com os avanços e com ferramentas de sucesso, o desenvolvimento de análise estática ainda é conhecido por ser um processo doloroso (TOMAN; GROSSMAN, 2017).

A eficiência, confiabilidade e precisão dessas ferramentas têm sido avaliadas e alguns estudos mostram inconsistência entre ferramentas diferentes. Um desses estudos comparou duas ferramentas de análise estática para cálculo de métricas e revelou significantes evidências sobre a inconsistencia entre valores de métricas, mostrando grande diferença entre os valores e discutindo quais problemas e questões levam a tais diferenças (ALEMERIEN; MAGEL, 2013).

Análise estática é a técnica mais amplamente utilizada para análise automatizada de programas devido a sua eficiência, boa cobertura e automação. Estudos mostram que analise estática tem grande adoção em projetos de software livre (BELLER et al., 2016). Entretanto, técnicas de análise estática amplamente adotadas na comunidade de software, por exemplo, para localização de bugs e verificação de programas ainda sofrem um alto índice de falso-positivos (GOSAIN; SHARMA, 2015).

Apesar da ampla adoção de ferramentas de análise estática em estudos acadêmicos e da crescente atenção que as técnicas de análise estática de código tem recebido em pesquisas, nota-se ainda uma enorme distância entre a atenção dada na academia e sua adoção na indústria, identificando um *gap* entre estes dois contextos (ILYAS; ELKHALIFA, 2016).

### 2.4.3 Software acadêmico de análise estática

Na Ciência da Computação, particularmente na Engenharia de Software, tem-se notado um aumento constante no número de novos projetos de software acadêmico (ALLEN et al., 2017), especialmente em estudos de análise estática, uma área com uma longa e respeitável tradição em pesquisas sobre a criação de novas ferramentas, métodos e algoritmos.

O software acadêmico de análise estática e o seu ecossistema, ao estar inserido no sistema acadêmico e intimamente relacionado à economia de reputação científica, sofre as consequências da competição que permeia este modelo, e o seu uso invariavelmente deixa de gerar *feedback* positivo de volta ao seu ecossistema, conforme Figura 2.2, onde se apresenta o relacionamento entre a prática e a pesquisa de software acadêmico.

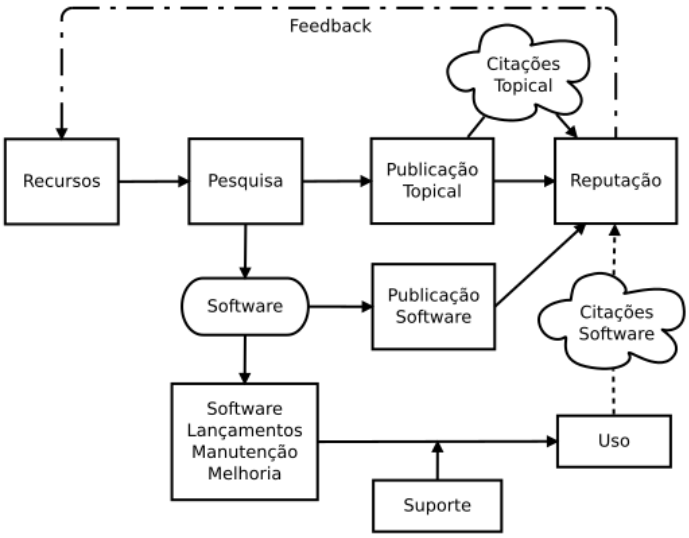



**Figura 2.2** Uma visão dos incentivos de reputação num contexto misto entre Ciência e práticas de software acadêmico (HOWISON; HERBSLEB, 2011)

A Figura 2.2 apresenta a situação enfrentada por muitos cientistas que distribuem software com intenção de ser utilizado por outros, há incertezas sobre o recebimento de citações para o software através do seu uso, esta incerteza é indicada na figura através da ligação pontilhada entre o uso do software e a reputação obtida através de citações. Diferentemente da publicação científica regular (*Publicação topical*), a publicação do software raramente é útil por si mesma; geralmente é o software que é útil para outros cientistas, no entanto, nem todos os usuários de software acadêmico vêem como requisito citar o software, a nem todos os projetos de software acadêmico indicam uma forma conveniente de como deve ser citado.

Diferentemente de outras tecnologias, software pode ser copiado e distribuído essencialmente sem custo, abrindo portas sem precedentes para compartilhamento e inovação colaborativa (HOWISON; HERBSLEB, 2011). No entanto, por estar de alguma forma conectado ao contexto de competição da economia de reputação científica, como no mecanismo de crédito acadêmico aos artigos e publicações, pode ser potencialmente pro-

blemático para a colaboração e manutenção (HOWISON; HERBSLEB, 2011).

Este cenário, além de desacelerar o progresso geral da Ciência gerando retrabalho, faz surgir questionamentos sobre as conclusões dessas pesquisas, especialmente quando grande parte dos pesquisadores não sabem o quão confiável seus projetos de software são. Criando assim, um contexto em que muitos estudos em Engenharia de Software sofrem de dificuldades de repetição (KAZMAN, 2016), além de ocasionar problemas específicos relacionados a manutenibilidade e sustentabilidade técnica do software acadêmico.

## 2.5 SUSTENTABILIDADE DE SOFTWARE ACADÊMICO

O desenvolvimento de software sustentável tem sido identificado como um desafio chave no campo da Ciência e da Engenharia Computacional. Se sustentabilidade não for levada em consideração em projetos de software, não importa qual o domínio ou qual o propósito do software, perde-se a oportunidade de causar mudanças positivas no planeta e na sociedade (BECKER et al., 2014).

Sustentabilidade de software apesar de ser um conceito complexo e com mútiplas dimensões, levando a debates profundos, possui um conceito geral bastante simples: refere-se à capacidade de perdurar e de continuar sendo suportado ao longo do tempo, o que implica nas qualidades de longevidade e manutenibilidade do software (VENTERS et al., 2014).

Software sustentável é aquele que continua a estar disponível no futuro, em novas plataformas, atendendo continuamente às novas necessidades do ambiente através de uma adequada evolução frente a condições em constante mudança (ALLEN et al., 2017). Este conceito, no entanto, ao ser aplicado ao contexto de software acadêmico e artefatos digitais, mostra-nos um cenário bastante preocupante, uma vez que, estudos recentes mostram que há uma tendência ao decaimento de URLs ao longo do tempo em publicações com produção de artefatos digitais (WREN et al., 2017).

Isto tem motivado iniciativas de tornar estes artefatos duráveis e disponíveis, visando especialmente garantir a longevidade dos artefatos e proporcionar que um segundo pesquisador receba todos os benefícios do trabalho do primeiro pesquisador (KING, 1995). O *JASA (Journal of the American Statistical Association)*[1], por exemplo, tem insistido na necessidade de disponibilizar código e dados ao menos durante a revisão dos manuscritos (BAKER, 2016). Agências de financiamento, como o *US National Science Foundation*, estão começando a reconhecer produtos de pesquisa, como software, assim como fazem com as publicações, tornando o software produzido durante pesquisas em cidadão de primeira classe na Ciência (ALLEN et al., 2017).

Uma outra iniciativa neste sentido, liderada pela pela *Free Software Foundation Europe* (FSFE), chamada *Public Money, Public Code*[2], propõe uma nova legislação obrigando que todo software financiado com dinheiro dos contribuintes, desenvolvido para o setor público, seja publicado com licenças de Software Livre. Esta campanha defende que assim é possível obter redução de custos, aumento de colaboração e transparência, ao mesmo tempo que estimula a inovação.

[1] ⟨http://www.amstat.org/PUBLICATIONS/jasa⟩
[2] ⟨https://publiccode.eu/pt⟩

Mas, além das garantias de longevidade ao software, a sua qualidade também deve ser avaliada, uma vez que a maioria dos cientistas autores de software não sabe o quão confiável seu software é (MERALI, 2010). Muitos dos projetos de software acadêmico estão em estado inicial de desenvolvimento (MARSHALL; BRERETON, 2013), e poucos foram testados fora do contexto onde foram desenvolvidos (PORTILLO-RODRíGUEZ et al., 2012).

Dessa forma, não é difícil perceber que o ecossistema de software acadêmico sofre graves problemas, percepção bastante similar ao fenômeno chamado de desordem caótica disfuncional ( *"dysfunctional chaotic churn"*), caracterizado por:

> Existência de muitos projetos com poucos usuários, com ciclos de vida curtos que se encerram junto ao financiamento inicial, comunidades de usuários desconectadas e paralelas, incompatibilidades entre os projetos de maneira persistente e imutável, e tentativas constantes e aparentemente não coordenadas de "reiniciar" tudo (*re-boots*) (HOWISON et al., 2015).

Este problema, apesar de ser apenas uma percepção, coincide com inúmeras evidências a respeito de problemas com o desenvolvimento, reconhecimento e sustentabilidade de software acadêmico (ALLEN et al., 2017).

**Desenvolvimento.** O desenvolvimento de software acadêmico exige, muitas vezes, conhecimento específico sobre o domínio do estudo sendo realizado, por exemplo, entender como o DNA genômico se transforma em cristais de proteína, ou estar familiarizado com os meandros da dinâmica dos fluidos, ou saber como resolver 20 equações diferenciais parciais simultâneas (SEGAL; MORRIS, 2008).

Isto explica a grande participação dos cientistas no desenvolvimento de software acadêmico. Estudos têm mostrado, por exemplo, que no Reino Unido entre todas as áreas da Ciência 56% dos cientistas estão envolvidos no desenvolvimento de software acadêmico (HETTRICK et al., 2014), outros estudos em grupos específicos mostram números ainda maiores, na astronomia, por exemplo, 90% dos cientistas desenvolvem software acadêmico (MOMCHEVA; TOLLERUD, 2015).

No entanto, a maior parte dos cientistas nunca tiveram treinamento algum sobre como escrever software de forma eficiente, muitos não testam ou documentam os seus projetos de software, faltam práticas básicas de desenvolvimento, como escrever código legível, revisão de código, controle de versão, testes unitários, entre outros (WILSON et al., 2017).

Isto tem ocasionado sérios erros computacionais em conclusões centrais da literatura acadêmica, gerando retrabalho para retratar tais erros nas mais diversas áreas da Ciência (MERALI, 2010). Dados são perdidos, análises levam mais tempo que o necessário e os pesquisadores não conseguem a eficiência que poderiam ter ao trabalhar com software acadêmico (WILSON et al., 2017), causando um impacto negativo na visibilidade do software acadêmico e na capacidade de ser encontrado e compartilhado (HOWISON; HERBSLEB, 2013; KATZ, 2014).

**Reconhecimento.**

Apesar do crescimento no uso de software e na consequente dependência entre cientistas de todos os campos, tornando o software acadêmico parte integral da prática científica, estudos têm mostrado que muitas pesquisas não mencionam sequer o uso de software acadêmico em suas publicações mesmo tendo feito uso de tais artefatos (MOMCHEVA; TOLLERUD, 2015; HOWISON; BULLARD, 2016).

Isto tem prejudicado a visibilidade do software acadêmico causando impacto negativo em seu ecossistema, um software invisível é frequentemente excluído de revisões por pares, uma atividade que costuma contribuir para a qualidade geral do trabalho publicado. Além disso, o impacto negativo na visibilidade do software acadêmico faz surgir uma série de questionamentos sobre a sua qualidade e também sobre a capacidade de ser encontrado, compartilhado e co-desenvolvido (HOWISON; HERBSLEB, 2013; KATZ, 2014; HOWISON; BULLARD, 2016).

Apesar de nem sempre ser possível ou viável, ter tudo dentro de padrões estritos, é preciso estar consciente das boas práticas ao produzir e utilizar software acadêmico, tanto para melhorar a própria abordagem quanto para revisar outros trabalhos (WILSON et al., 2014). Um software acadêmico em bom funcionamento deve atingir não apenas os objetivos de entendimento e transparência, mas também os objetivos voltados para replicação, seja logo após sua publicação, seja daqui a 10 ou 50 anos (STODDEN, 2010).

**Manutenibilidade.**

Manutenibilidade é uma característica de qualidade que indica o quão fácil é realizar atividades de evolução e manutenção em software (KUMAR, 2012), um aspecto importante aos pesquisadores interessados em adaptar software acadêmico, algo muitas vezes necessário ao reproduzir pesquisas anteriores (PENG, 2011).

Estudos têm mostrado que grande parte das ferramentas de software criadas na academia estão em estado inicial de desenvolvimento (MARSHALL; BRERETON, 2013) e que apenas uma pequena porcentagem são testados fora do contexto onde foi desenvolvido (PORTILLO-RODRíGUEZ et al., 2012).

A adoção e uso de software acadêmico está relacionado também à sua qualidade, portanto é importante medir e coletar sua qualidade de alguma forma. Manutenibilidade, por exemplo, apesar de ser um atributo de qualidade de software inerentemente externa, pode ser medida através de características de qualidade interna (HASHIM; KEY, 1996; DAGPINAR; JAHNKE, 2003), uma vez que grande parte dos engenheiros de software assumem que uma boa estrutura interna resulta em boa qualidade externa (FENTON; BIEMAN, 2014).

No entanto é importante compreender os projetos de software acadêmico também em termos de evolução e ciclo de vida, algo que pode influenciar enormemente as medidas de qualidade coletadas através de atributos internos do software, como métricas de código-fonte, por exemplo.

## 2.6 CICLO DE VIDA DE SOFTWARE

Engenheiros de software têm tradicionalmente considerado qualquer trabalho após o primeiro lançamento de um software simplesmente como manutenção. Alguns pesquisadores no entanto têm dividido este trabalho em atividades distintas, incluindo adaptação, prevenção, correção, entre outras, mas sempre considerando manutenção basicamente uniforme ao longo do tempo (RAJLICH; BENNETT, 2000).

Entretanto, alguns estudos têm demonstrado que esta visão, onde manutenção ocupa um papel basicamente uniforme ao longo do tempo, não explica muito bem o desenvolvimento de software na maior parte dos cenários e uma das abordagens para explicar o fenômeno tem colocado a atividade de manutenção distribuída ao longo do ciclo de vida do software (RAJLICH; BENNETT, 2000).

Este modelo de evolução do software em estágios, no entanto, foi avaliado e adaptado ao contexto de software livre (CAPILUPPI et al., 2007) e diante das semelhanças com o software acadêmico, este modelo adaptado ao software livre serve ao propósito de ser utilizado para avaliar a evolução do software acadêmico. A Figura 2.3 apresenta o modelo de evolução adaptado ao software livre, adotado neste estudo como aplicável também ao software acadêmico.

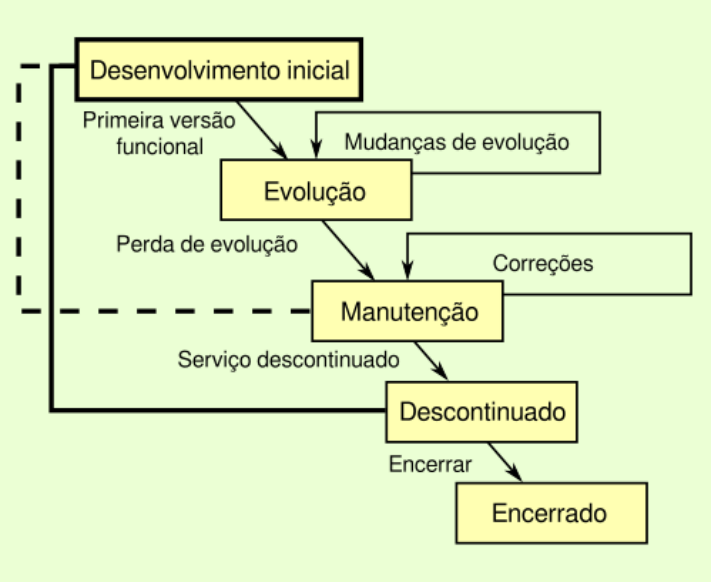



**Figura 2.3** O modelo de evolução *staged model* adaptado a software livre (CAPILUPPI et al., 2007).

A Figura 2.3 apresenta cinco estágios distintos de evolução no ciclo de vida de um software, cada estágio possui atividades distintas, necessidades e ferramentas variadas, e consequencias diversas em relação ao negócio. Na fase de *Desenvolvimento inicial* os engenheiros de software desenvolvem a primeira versão funcional do sistema, passando em seguida para a fase de *Evolução* onde as capacidades e funcionalidades do sistema são expandidas para atender as necessidades dos usuários, chegando ao estágio de *Manutenção* onde os engenheiros fazem pequenos reparos em defeitos e mudanças funcionais simples,

quando os responsáveis pelo sistema decidem não mais oferecer serviço com o software ele é posicionado no estágio *Descontinuado*, os responsáveis procuram gerar receita com o sistema neste estágio pelo maior tempo possível até atingir o estágio final *Encerrado*, quando o sistema é retirado do mercado e os usuários são direcionados para um novo sistema, se houver (RAJLICH; BENNETT, 2000).

A primeira diferença observada é em relação a fase *Desenvolvimento inicial*, dependendo da definição de "fase inicial" muitos projetos de software livre podem nunca ter saído desta fase, assim é também para o software acadêmico. Em respeito aos lançamentos, em sistemas comerciais tradicionais eles devem ser completos, rodando e autorizado pela empresa detentora, enquanto no mundo do software livre é comum permitir acesso público ao código em repositórios de código-fonte, seguindo um modelo de "versão permanente".

A segunda diferença é relacionada à possibilidade de laços entre as fases *Evolução* e *Manutenção*. Muitos projetos de software livre possuem fases de congelamento na adição de novas funcionalidades (*freeze*) enquanto permanece numa fase de *Manutenção* até o descongelamento, voltando a *Evolução*.

A terceira diferença está na comunidade de software livre, novos times de desenvolvimento são formados ao longo do tempo com a saída de desenvolvedores antigos e a entrada de novos. E projetos em fase *Descontinuado* podem experimentar um renascimento retornando ao período de *Evolução*.

Apesar das diferenças, os autores Capiluppi et al. (2007) demonstram em experimentos e revisão de literatura que o modelo adaptado pode ser utilizado para compreender evolução de software livre, e diante as similaridades entre software livre e software acadêmico, este modelo poderá, consequentemente, ser também utilizado para compreender a evolução e ciclo de vida do software acadêmico.

## 2.7 REPRODUTIBILIDADE

Reprodutibilidade (*reproducibility*) é a habilidade de replicar um experimento ou estudo em sua totalidade a fim de confirmar suas hipóteses, seja pelo autor original ou por pesquisadores independentes. Esse conceito é um ponto central do método científico e tem sido uma preocupação crescente da comunidade de pesquisa em Engenharia de Software Empírica (GONZáLEZ-BARAHONA; ROBLES, 2012).

A verificação de reprodutibilidade é essencial para a prática do método científico, pesquisadores reportam seus resultados, que são reforçados à medida que outros grupos independentes da comunidade científica compartilham resultados semelhantes (MCCORMICK et al., 2014). Experimentos são repetidos com objetivo de checar seus resultados, replicações bem sucedidas aumentam a validade e confiabilidade dos resultados observados no experimento (ALMQVIST, 2006; JURISTO; GóMEZ, 2012).

Este conceito apesar de ser relativamente concensual entre as várias áreas da Ciência carece de unidade, inúmeros autores de áreas distintas tem debatido o tema e proposto terminologias (DRUMMOND, 2009; VITEK; KALIBERA, 2011; MENDOZA; GARCIA, 2017; PLESSER, 2018), segundo Feitelson (2015) temos as seguintes definições:

**Repetição (*repetition*)** Refazer exatamente o que outra pessoa fez usando os artefatos

originais.

**Replicação** (*replication*) Replicar com precisão exatamente o que outra pessoa fez, recriando os artefatos.

**Variação** (*variation*) Repetir ou replicar exatamente o que a outra pessoa fez, mas com alguma modificação controlada nos parâmetros.

**Reprodução** (*reproduction*) Recriar o espírito do que outra pessoa fez, usando seus próprios artefatos.

**Corroboração** (*corroboration*) Obter os mesmos resultados de outra pessoa, usando outros meios e procedimentos experimentais.

A comunidade de Engenharia de Software tem tentado por anos replicar experimentos para gerar fragmentos de conhecimento, mas a grande parte das tentativas tem levado a resultados insatisfatórios (a credibilidade dos resultados não aumentou), principalmente por conta das variações nas condições experimentais de uma replicação para outra (JURISTO; VEGAS, 2009).

Um estudo com 88 papers da conferência *MSR (Mining Software Repositories)*[3] entre 2004-2011 evidenciou que apenas 62% são replicáveis ou parcialmente replicáveis e que apenas 20% dos estudos disponibilizam suas ferramentas (AMANN et al., 2015). Um estudo anterior com 171 papers do MSR evidenciou que, entre outros problemas, a maioria não disponibiliza publicamente as ferramentas e *scripts*, mesmo quando os autores explicitamente afirmam que construíram algo (ROBLES, 2010), apenas 2 entre 154 estudos experimentais avaliados fornecem os dados e ferramentas necessárias para replicação e futuras pesquisas (BARR et al., 2010).

Esta dificuldade em atingir reprodutibilidade em estudos científicos baseados em métodos computacionais tem levado a sérios erros em resultados publicados na literatura acadêmica (HINSEN, 2015). O número de repetições apesar de ter crescido nos últimos anos ainda é muito pequeno, especialmente considerando a amplitude de tópicos em Engenharia de Software (SILVA et al., 2011; KITCHENHAM; BUDGEN; BRERETON, 2015).

Em muitos campos científicos onde software tem se tornado ferramenta fundamental para capturar e analisar dados, este requerimento por reprodutibilidade implica que plataformas de software e ferramentas confiáveis e compreensíveis devem estar disponíveis para a comunidade científica (MCCORMICK et al., 2014). No entanto, ainda vemos o seguinte questionamento a respeito dos artefatos de software em pesquisas da Ciência da Computação: "Onde está o software nas pesquisas sobre linguagem de programação?" (KRISHNAMURTHI; VITEK, 2015).

Toda pesquisa que possua qualquer processo computadorizado deve publicar seus códigos, eles precisam estar disponíveis, mesmo que os dados correspondentes não estejam. De acordo com o espectro de reprodutibilidade (Figura 2.4), a disponibilidade de código é o requisito mínimo e é o primeiro passo para possibilitar validação e confirmação dos resultados (PENG, 2011).

[3] ⟨http://msrconf.org⟩

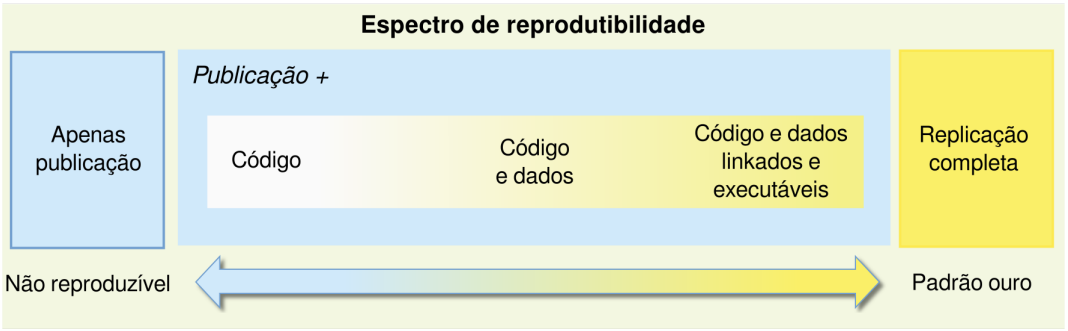



**Figura 2.4** Espectro de reprodutibilidade (PENG, 2011)

No entanto, esta prática ainda carece de maior adoção, as questões por trás destes problemas são, além de outros, possivelmente, ocasionados por questões culturais, como, por exemplo, a tímida adoção de práticas da ciência aberta entre pesquisadores (NIEMEYER, 2017).

## 2.8 CIÊNCIA ABERTA

Ciência Aberta é um movimento que tem por objetivo tornar a pesquisa científica, seus dados e sua disseminação acessíveis à todos os interessados, sejam amadores ou profissionais (WIKIPEDIA, 2015). Sua principal motivação está em possibilitar a reprodução dos resultados de pesquisas e em garantir transparência das metodologias utilizadas, isto aumenta o impacto social das pesquisas e gera economia de tempo e dinheiro para os pesquisadores e para as instituições (RIN/NESTA, 2010).

Este movimento é guiado por princípios básicos de transparência, acessibilidade e reusabilidade universais, disseminados por ferramentas *online*, ele é dividido em quatro grandes áreas (PONTIKA et al., 2015):

1. *Open Access*. Resultados acadêmicos, revisado por pares, disponível *online*, de maneira livre.

2. *Open Data*. Dados coletados durante projetos de pesquisa, disponibilizados *online* com permissão explícita para uso.

3. *Open Source*. Software acessível *online*, com código-fonte disponível, com licenças que permitam uso, modificação e re-distribuição.

4. *Open Reproducible Research*. Ato de aplicar as práticas da Ciência Aberta para permitir reprodutibilidade de maneira independente dos resultados de uma pesquisa.

A Ciência Aberta pode ter praticada tanto por razões filosóficas quanto pragmáticas, os recursos produzidos por projetos abertos são acessíveis ao público, oferecendo um meio inovador para o envolvimento de terceiros no processo de Ciência, além de ser um método inovador para comunicação científica em tempo real (GRAND et al., 2010).

Capítulo

3

# PUBLICIZAÇÃO DE SOFTWARE ACADÊMICO DE ANÁLISE ESTÁTICA

Este capítulo apresenta um estudo sobre a publicização de software acadêmico de análise estática divulgado nas conferências ASE – Automated Software Engineering e SCAM – Working Conference on Source Code Analysis & Manipulation, até o ano de 2015. Nossa revisão de literatura identificou 60 projetos de software de análise estática publicados em artigos nessas duas conferências e caracterizou esses projetos em relação a sua publicização, em termos de disponibilidade para download, acesso ao código-fonte, forma de distribuição e licença.

A seção 3.1 apresenta a motivação para realização do estudo, a seção 3.2 define o escopo, descreve o objetivo e apresenta as questões de pesquisa, a seção 3.3 apresenta um planejamento do estudo, as seções 3.4 e 3.5 apresentam detalhes sobre a preparação e execução da coleta de dados, as seções 3.6 e 3.7 apresentam a análise e interpretação dos dados e a seção 3.9 apresenta as conclusões finais deste estudo.

## 3.1 MOTIVAÇÃO

A *publicização* do software acadêmico, isto é, a ação de tornar o software disponível a partir da URL indicada no artigo científico publicado pelos seus autores, é fundamental. Características importantes do software acadêmico após a sua publicização incluem a forma em que os projetos são disponibilizados (binário ou código-fonte), os tipos de licença utilizados e outras questões relacionadas à sua distribuição.

No entanto, muitos projetos de software acadêmico possuem ciclos de vida curtos, tornando-se indisponíveis tão logo acaba o financiamento inicial (HOWISON et al., 2015). Parte deste problema tem sido atribuído à falta de treinamento em boas práticas para o desenvolvimento de software de qualidade por parte dos cientistas e pesquisadores (WILSON et al., 2017) e de publicização adequada do software acadêmico (ALLEN et al., 2017). A princípio, a falta de treinamento não deveria afetar os cientistas da Engenharia de Software, visto que estes possuem formação e acesso a práticas de desenvolvimento

e estão inseridos num contexto voltado para a compreensão e melhoria dos processos de desenvolvimento de software, tanto do ponto de vista teórico, quanto prático.

Sabendo disso, e atentando para o fato de que grande parte do desenvolvimento de software acadêmico é realizado pelos próprios cientistas (HETTRICK et al., 2014; MOMCHEVA; TOLLERUD, 2015), surge a preocupação de avaliar a publicização do software acadêmico desenvolvido nas pesquisas de Engenharia de Software, especialmente em relação aos sintomas "dysfunctional chaotic churn" apontadas por Howison et al. (2015).

## 3.2 ESCOPO

Projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM são nosso principal objeto de estudo. Queremos saber quantos projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM foram publicizados adequadamente, se permanecem disponíveis e quais são as informações que podem ser obtidas.

O objetivo da pesquisa está definido segundo a estrutura GQM (BASILI; CALDIERA; ROMBACH, 1994).

### 3.2.1 Definição do Objetivo

**Objeto de estudo.** O objeto de estudo são projetos de software acadêmico de análise estática publicados em artigos científicos e sua publicização em termos de disponibilidade, licença e distribuição, conforme definido na seção 3.1.

**Propósito.** O propósito do estudo é caracterizar a publicização de cada software acadêmico de análise estática publicado nas conferências selecionadas. O estudo contribuirá para a análise da desordem disfuncional caótica no domínio de análise estática.

**Perspectiva.** A perspectiva considerada é a do cientista usuário final, isto é, o pesquisador que deseja saber se há DDC no software acadêmico do domínio de interesse. A perspectiva a ser considerada é a do pesquisador que deseja conhecer ferramentas de análise estática para uso em sua pesquisa.

**Foco de qualidade.** O principal foco de qualidade estudado é a publicização do software acadêmico de análise estática, com ênfase nos aspectos de disponibilidade de URL com infornações sobre o projeto, especialmente sobre acesso ao código-fonte.

**Contexto.** O estudo foi conduzido com artigos que apresentam software acadêmico de análise estática, publicados das conferências de Engenharia de Software ASE e SCAM.

### 3.2.2 Sumário da Definição

Analisar os *projetos de software acadêmico de análise estática* publicados com o propósito de *caracterizá-los* com respeito a sua *publicização* na perspectiva de *cientistas usuários*

*finais* no contexto das *conferências de Engenharia de Software ASE e SCAM.*

### 3.2.3 Questões de Pesquisa

Neste estudo as seguintes questões de pesquisa, a respeito dos projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM, e sua publicização serão investigadas:

**Q1:** Os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM possuem alguma presença oficial online?

Por presença oficial online entende-se site ou repositório oficial do projeto, um endereço URL com informações do software, geralmente mantido pelos próprios autores do software.

**Q2:** Os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM estão disponíveis para download?

Com esta questão queremos saber se é possível obter uma cópia online do software de forma que autores independentes possam utilizar e avaliar o software, seja com fins de reprodutibilidade de estudos utilizando o software, seja para fins de novas pesquisas com o apoio do software.

**Q3:** É possível ter acesso ao código-fonte dos projetos de software de análise estática publicados nas conferências ASE e SCAM?

Com esta questão queremos saber se pesquisadores e interessados no conhecimento empregado na construção do software e presente no próprio código-fonte terão acesso a este conhecimento, que usualmente faz parte da própria pesquisa. O acesso ao código-fonte é útil também para aqueles que tenham interesse no uso, eventuais adaptações ou correções podem ser necessárias para executar em outros ambientes ou sistemas operacionais.

**Q4:** Os projetos de software com código-fonte disponível usam licenças de software livre?

Com esta questão queremos saber se os projetos possuem permissão explícita para contribuição via código-fonte de forma que possam ser adaptados para atender necessidades emergentes e que estas adaptações possam também ser distribuídas livremente.

### 3.2.4 Métricas

Para responder às questões de pesquisas, as seguintes métricas serão usadas:

1. Número de projetos com identificação de nome e URL
2. Número de projetos com URL disponível
3. Número de projetos disponíveis para download

4. Número de projetos com código-fonte disponível
5. Número de projetos usando licenças de software livre ou código aberto

## 3.3 PLANEJAMENTO DO ESTUDO

Este estudo foi realizado em duas etapas principais, conforme ilustrado na Figura 3.1. Na primeira etapa, uma revisão da literatura foi realizada, com o objetivo selecionar artigos com publicação de software acadêmico de análise estática. Na segunda etapa, os projetos de software selecionados foram caracterizados em relação à sua publicização.

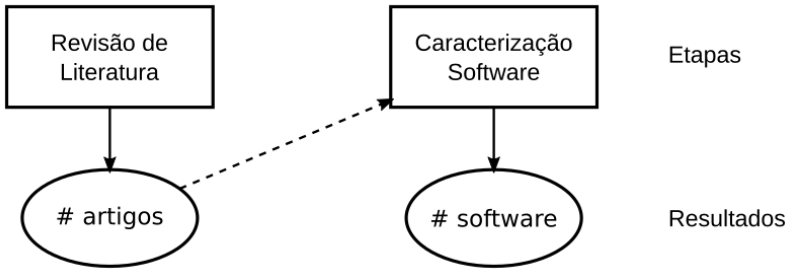



**Figura 3.1** Etapas do estudo e seus resultados

### 3.3.1 Revisão de literatura com publicação de software de análise estática

O objetivo da revisão de literatura realizada foi encontrar artigos mencionando software de análise estática entre os resultados do estudo. A revisão foi organizada em três passos, detalhados a seguir.

**Passo 1: Escopo** Este passo teve como objetivo a seleção das conferências a serem utilizadas como fonte dos artigos para a revisão de literatura. Para cada conferência, todas as publicações serão selecionadas e incluídas no conjunto inicial de artigos da revisão, iniciando pela primeira edição de cada conferência indo até a última edição disponível no ano da execução da revisão.

As conferências devem ser selecionadas tendo em vista aumentar e potencializar o número total de projetos de software acadêmico do domínio de aplicação de interesse do estudo, neste caso, análise estática. Deve-se buscar preferencialmente conferências com bom histórico de publicações sobre o domínio de aplicação do estudo.

É importante investigar o histórico de cada conferência com atenção ao nome do evento em cada edição; não é raro que conferências com muitos anos de existência mudem de nome ao longo do tempo. Nestes casos, deve-se adotar o nome da conferência mais recente e considerar todas as edições anteriores com este mesmo nome.

Uma vez definidos a data limite e as conferências, deve-se fazer o download de todos os artigos até a data limite, e os arquivos devem ser organizados em pastas por nome

e ano da conferência. Os títulos de todos os artigo, a correspondente conferência e o ano de publicação devem ser registrados e armazenados em arquivos ou banco de dados. Neste estudo fizemos uso do LibreOffice Calc[1].

**Passo 2: Triagem Automática** O escopo definido no passo anterior irá, potencialmente, resultar num enorme conjunto de artigos para revisão. Assim, o objetivo da triagem automática é reduzir o tamanho deste conjunto, mantendo apenas as publicações relevantes para os objetivos do estudo.

A triagem é realizada por meio de uma busca textual no conteúdo de cada artigo. A busca deve ser realizada de forma automática com auxílio de ferramentas de software para busca textual em conteúdo de arquivos pdf, com base em um conjunto de palavras-chave representando os critérios de inclusão da triagem. Apenas os artigos combinando com todos os critérios serão incluídos, conforme Tabela 3.1.

**Tabela 3.1** Critérios de inclusão e palavras-chave para a triagem de artigos.

| Critério | Palavras chave |
|---|---|
| Menciona projeto, software ou ferramenta | `tool` ou `framework` |
| Disponibiliza download do projeto | `download` ou `available` |
| Identifica URL do projeto | `http`, `https` ou `ftp` |
| Domínio de análise estática | `static analysis` ou `parser` |

Se um determinado artigo não atender a algum dos critérios, ele será excluído. Ao final da triagem, espera-se obter um conjunto reduzido de artigos, contendo apenas as publicações relevantes para o objetivo deste estudo.

**Passo 3: Extração** Os artigos incluídos serão inspecionados manualmente em busca de software acadêmico de análise estática entre os resultados do estudo. Os critérios de inclusão desta etapa têm como objetivo incluir apenas os projetos de software identificados minimamente pelos seus autores com nome e URL para download, conforme Tabela 3.2.

**Tabela 3.2** Critério de inclusão para o passo de extração dos artigos (adaptado de Howison e Bullard (2016)).

| Critério | Explicação |
|---|---|
| Identificável | É possível identificar um projeto de software entre as contribuições do artigo? (ex: “um programa que nós escrevemos“, “nossa implementação“, “nosso protótipo”) |
| Disponível | Podemos encontrar menção a URL do projeto de software para download? |

O nome e o endereço URL de cada projeto encontrado durante a inspeção manual devem ser extraídos e armazenados junto aos dados já coletados nos passos anteriores da revisão. A inspeção deve incluir a leitura do título, introdução, resultados e

[1] ⟨https://www.libreoffice.org⟩

conclusões de cada artigo e, caso necessário, outras seções do artigo. Alguns artigos podem descrever a contribuição de software acadêmico em seções específicas, por exemplo, em notas de rodapé com a URL do software.

Os artigos devem ser inspecionados num ciclo, um por vez, até chegar ao útlimo artigo dentre os selecionados no passo anterior. Ao final da extração, teremos dados de um conjunto de projetos de software acadêmico de análise estática, incluindo nome do projeto e URL para download. Estes dados extraídos devem ser atualizados nos arquivos ou banco de dados já utilizados nos passos anteriores.

### 3.3.2 Caracterização de software acadêmico de análise estática

Os dados coletados na revisão de literatura, devidamente armazenados em arquivos ou banco de dados, incluem projetos de software acadêmico de análise estática, identificados com nome e URL, nome e ano da conferência e título do artigo onde foi publicado.

Estes dados irão compor uma nova estrutura de armazenamento projetada para receber informações adicionais sobre os projetos, extraídas de outras fontes, conforme Tabela 3.3. Essas informações serão coletadas em documentos relacionados aos projetos, como website, manuais, repositórios e código-fonte.

**Tabela 3.3** Esquema para caracterização dos projetos de software acadêmico

| Característica | Explicação |
|---|---|
| Nome do software | O nome do projeto de software |
| URL | Endereço web do software ou projeto, site ou repositório de código-fonte, com o software disponível |
| Título do artigo | Título do artigo onde o software é citado como contribuição, seja principal ou secundária |
| Nome do evento | Nome da conferência onde o software foi publicado |
| Ano do evento | Ano da edição da conferência onde o artigo foi publicado |
| Descrição do software | A descrição do projeto de software |
| Acesso | Podemos acessar a URL do projeto agora? Pode receber os seguintes valores: Sem Acesso, Acesso Pago, Acesso Gratuito |
| Distribuição | Como o software é distribuído e pode ser acessado? Pode receber os seguintes valores: gratis, livre, proprietário |
| Código-fonte disponível | É possível acessar o código-fonte de alguma forma? |
| Licença | O software deixa explícito sob qual licença é distribuído? |
| Linguagem de programação | Em qual linguagem de programação o software acadêmico foi desenvolvido |

O primeiro documento consultado deve ser o website indicado pela URL. Esta consulta resultará em novos documentos, como, manuais de uso, informações sobre desenvolvimento, código-fonte, notas de lançamento, entre outros. Tais documentos devem ser inspecionados em busca de informações para caracterização dos projetos segundo a Tabela 3.3.

A identificação da linguagem de programação em que projeto está escrito deve ser realizada de maneira automática com apoio de alguma ferramenta de análise de código-fonte, como por exemplo, o software livre `sloccount`[2].

## 3.4 PREPARAÇÃO

Nesta seção apresentamos a preparação do estudo para a revisão da literatura e realização da coleta de dados.

### 3.4.1 Revisão de literatura com publicação de software de análise estática

**Passo 1: Escopo** Seguindo o planejamento descrito em 3.3, definimos como fonte de busca as conferências SCAM e ASE, ambas conferências com um grande histórico de edições. Definimos como data limite o ano de 2015.

Os artigos das duas conferências estão disponíveis para download: as publicações da conferência SCAM estão todas disponibilizadas no IEEE, e a conferência ASE possui algumas edições no IEEE outras na ACM. Os endereços URL de cada edição de cada conferência estão documentadas no repositório[3] dessa dissertação, apresentado em detalhes no Apêndice A.

Até o ano de 1996, a conferencia ASE chamava-se *KBSE (Knowledge-Based Software Engineering Conference)*. Teve sua primeira edição no ano de 1991 e, a partir de 1997 passou a se chamar *ASE (Automated Software Engineering Conference)*. A conferência SCAM teve sua primeira edição no ano de 2001.

**Passo 2: Triagem Automática** Para a execução da triagem automática a partir dos critérios definidos no planejamento (Seção 3.3), implementamos um script, que utiliza como base o software livre `pdftotext`[4], para converter o conteúdo pdf dos arquivos em texto antes de realizar o filtro pelas palavras-chave.

**Passo 3: Extração** Para a execução da extração, criamos uma planilha LibreOffice Calc para coletar os dados de cada passo da revisão de literatura com as seguintes colunas:

**Evento** Nome e ano da conferência de Engenharia de Software.

**Escopo** Título do artigo incluído no Passo 1: Escopo.

**Triagem** 'SIM' se o artigo foi incluído no Passo 2: Triagem Automática.

**Extração** 'SIM' se o artigo foi incluído na Passo 3: Extração.

**Software** Nome do software identificado no artigo.

**Anotações** Observações sobre o contexto em que o software é mencionado.

[2] http://www.dwheeler.com/sloccount

[3] ⟨https://github.com/joenio/dissertacao-ufba-2016⟩

[4] ⟨https://en.wikipedia.org/wiki/Pdftotext⟩

### 3.4.2 Caracterização de software acadêmico de análise estática

Nesta fase de preparação para a caracterização dos projetos, definimos a estrutura de diretórios e nomes e formatos de arquivos utilizados para armazenar os dados coletados. Optamos por criar arquivos no formato YAML com o nome `software.yml` para receber os dados do projeto e outro arquivo no formato BibTeX chamado `paper.bib` para armazenar os dados do artigo onde o software foi publicado.

Instalamos o software livre `sloccount`[5] para coletar a linguagem de programação em que o projeto está escrito.

## 3.5 COLETA DE DADOS

Seguindo o planejamento e preparação descritos nas seções 3.3 e 3.4, iniciamos a coleta dos dados, organizada em duas grandes etapas: uma de revisão de literatura, e outra de caracterização dos projetos de software acadêmico.

### 3.5.1 Revisão de literatura com publicação de software de análise estática

A revisão de literatura encontrou 61 artigos com publicação de projeto de software acadêmico de análise estática entre os 1873 artigos publicados nas conferências ASE e SCAM. As atividades e resultados de cada passo da revisão são representadas na Figura 3.2.

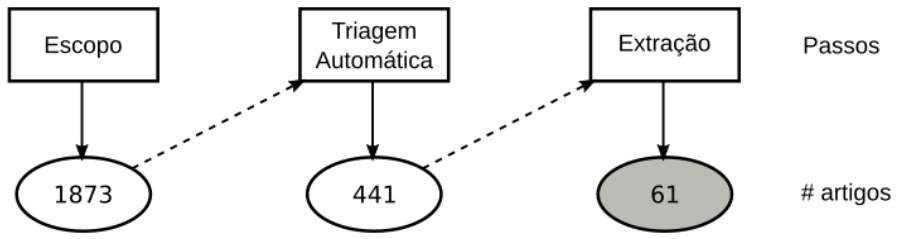



**Figura 3.2** Passos da revisão de literatura realizada nas conferências ASE e SCAM.

**Passo 1: Escopo** A busca realizada nas conferências ASE e SCAM até o ano de 2015 resultou em 1873 artigos; deste total, 346 artigos são do SCAM e 1527 artigos do ASE.

**Passo 2: Triagem Automática** O filtro executado automaticamente usando os critérios definidos na fase de preparação reduziu o conjunto de 1873 para 441 artigos, sendo 155 artigos do SCAM e 286 artigos do ASE selecionados para posterior inspeção manual.

**Passo 3: Extração** Os 441 artigos selecionados pela triagem automática foram inspecionados manualmente, tendo como guia os critérios descritos na Tabela 3.2 na

[5]http://www.dwheeler.com/sloccount

fase de planejamento. Ao final da inspeção manual, selecionamos 61 artigos com publicação de software acadêmico de análise estática.

### 3.5.2 Caracterização de software acadêmico de análise estática

Conforme planejado, coletamos informações adicionais para cada projeto de software, descritas na Tabela 3.3, consultando documentos de cada projeto. As informações coletadas são apresentadas, de forma resumida, na Tabela 3.4.

Tabela 3.4: Software acadêmico para análise estática.

| ID | Nome do software | Acesso | Linguagem de programação | Distribuição |
|---|---|---|---|---|
| s1 | 2LS | Gratuito | C++ | Livre |
| s2 | AccessAnalysis | Gratuito | Java | Livre |
| s3 | APIExample | Sem Acesso | - | - |
| s4 | BEG | Sem Acesso | - | - |
| s5 | ccJava | Sem Acesso | - | - |
| s6 | CIVL | Gratuito | C | Livre |
| s7 | CodeBoost | Gratuito | C | Livre |
| s8 | CSL | Gratuito | C | Grátis |
| s9 | CPA+ | Sem Acesso | - | - |
| s10 | CSeq | Gratuito | C | Livre |
| s11 | DDVerify | Sem Acesso | - | - |
| s12 | Derailer | Gratuito | Ruby | Livre |
| s13 | Diagnosys | Sem Acesso | - | - |
| s14 | DOMPLETION | Gratuito | Javascript | Grátis |
| s15 | DRC | Sem Acesso | - | - |
| s17 | e-munity | Gratuito | C | Grátis |
| s16 | EJB | Gratuito | Java | Grátis |
| s18 | Error Prone | Gratuito | Java | Livre |
| s19 | ESBMC | Sem Acesso | - | - |
| s20 | ETXL | Sem Acesso | - | - |
| s21 | FaultBuster | Gratuito | - | Grátis |
| s22 | Flowgen | Gratuito | Python | Livre |
| s23 | GRT | Sem Acesso | - | - |
| s24 | GUIZMO | Gratuito | Java | Livre |
| s25 | GumTree | Gratuito | Java | Livre |
| s26 | HUSACCT | Gratuito | Java | Livre |
| s27 | Indus | Gratuito | Java | Livre |
| s28 | JastAdd | Gratuito | Java | Livre |
| s29 | JFlow | Gratuito | Java | Livre |
| s30 | JstereoCode | Sem Acesso | - | - |
| s31 | Jtop | Sem Acesso | - | Livre |

continua na próxima página

| ID | Nome do software | Acesso | Linguagem de programação | Distribuição |
|---|---|---|---|---|
| s32 | Bogor/Kiasan | Gratuito | Java | Livre |
| s33 | Loopfrog | Gratuito | - | Grátis |
| s34 | Lotrack | Gratuito | Java | Grátis |
| s35 | MPAnalyzer | Gratuito | Java | Grátis |
| s36 | MSP | Sem Acesso | - | - |
| s37 | mygcc | Gratuito | C | Livre |
| s38 | PARSEWeb | Sem Acesso | - | - |
| s39 | PAT | Sem Acesso | - | - |
| s40 | PHP AiR | Gratuito | Rascal | Grátis |
| s41 | protopurity | Gratuito | Javascript | Grátis |
| s42 | Pseudogen | Gratuito | Python | Grátis |
| s43 | PtYasm | Gratuito | Java | Grátis |
| s44 | PuMoC | Sem Acesso | - | - |
| s45 | PYTHIA | Sem Acesso | - | - |
| s46 | ReAssert | Gratuito | Java | Livre |
| s47 | Rêve | Sem Acesso | - | - |
| s48 | RRFinder | Sem Acesso | - | - |
| s49 | Sapid/XML | Sem Acesso | - | - |
| s50 | Sonar Qube Plug-in | Gratuito | Java | Grátis |
| s51 | SPARTA | Gratuito | Java | Grátis |
| s52 | srcML | Gratuito | C++ | Livre |
| s53 | SWAT | Sem Acesso | - | - |
| s54 | TACLE | Gratuito | Java | Grátis |
| s55 | TEBA | Gratuito | Perl | Grátis |
| s56 | TestEra | Sem Acesso | - | - |
| s57 | Vdiff | Sem Acesso | - | - |
| s58 | WALA | Gratuito | Java | Livre |
| s59 | Wrangler | Gratuito | Erlang | Livre |
| s60 | XOgastan | Sem Acesso | - | - |

Dentre os 61 artigos selecionados, dois artigos faziam referência a um mesmo software e, portanto, um deles não foi contabilizado, resultando em um total 60 projetos de software selecionados. Para cada projeto foi criado um diretório contendo o nome do software e um arquivo chamado `software.yml` com os dados coletados.

## 3.6 ANÁLISE DOS DADOS

Coletamos 1873 artigos publicados nas conferências de Engenharia de Software ASE e SCAM, selecionamos 61 artigos com publicação de software acadêmico de análise estática, e entre estes artigos caracterizamos 60 projetos de software. Para cada um dos 60 projetos de software, coletamos ao menos o nome e a URL do projeto; informaçoes adicionais a respeito do projeto foram coletadas em fontes documentais diversas. A seguir apresentamos

a análise dos dados coletados.

### 3.6.1 Revisão de literatura com publicação de software de análise estática

**Passo 1: Escopo** Até 2015, a conferência ASE publicou quase 4 vezes mais do que a conferência SCAM. A edição com o maior número de publicações foi a de 2011, com 112 artigos publicados, seguido de 2014 com 104, e 2007 com 102; a edição com o menor número foi a de 1996 com apenas 15 artigos publicados, A Figura 3.3 apresenta estes totais distribuídos ao longo de cada edição.

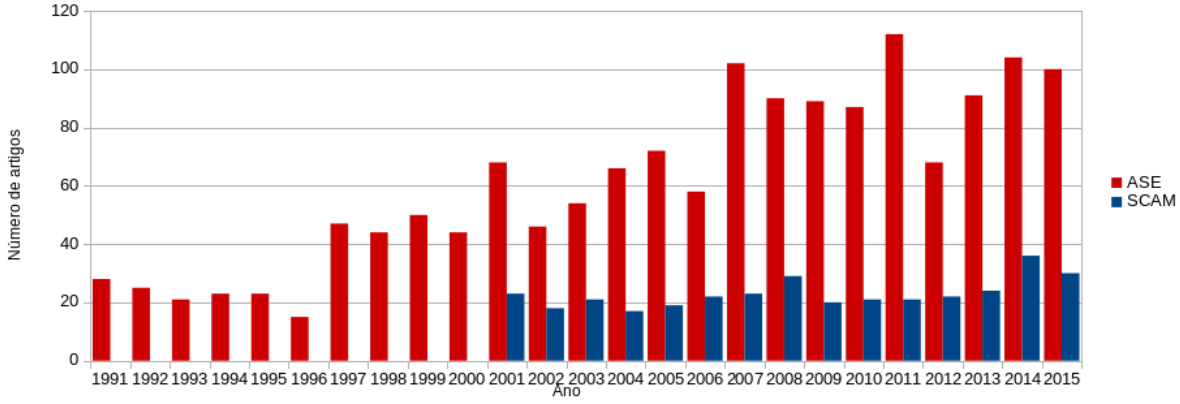



**Figura 3.3** Total de artigos por ano publicados nas conferências ASE e SCAM.

**Passo 2: Triagem Automática** Até o ano de 1996, nenhum artigo foi encontrado na conferência ASE com ocorrência das palavras-chave utilizadas no filtro. Apenas a partir de 1997 temos ocorrência dos termos pesquisados nas publicações desta conferência.

**Passo 3: Extração** Considerando todo o período incluído na revisão de literatura, encontramos uma média de 2 artigos por ano com publicação de software acadêmico. A Figura 3.4 apresenta os resultados distribuídos por ano.

A revisão de literatura teve como critério primordial encontrar publicação de software com indicação de URL para download; ainda assim, no processo de revisão, encontramos 51 projetos sem indicação de URL para download, sendo 30 projetos na conferência ASE e 21 projetos na conferência SCAM.

## 3.7 INTERPRETAÇÃO DOS RESULTADOS

### 3.7.1 Q1 - Os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM possuem alguma presença oficial online?

Dos 60 projetos de software estudados neste trabalho, 15 não têm presença oficial online no endereço de URL informado pelos seus autores, 45 projetos estão presentes através

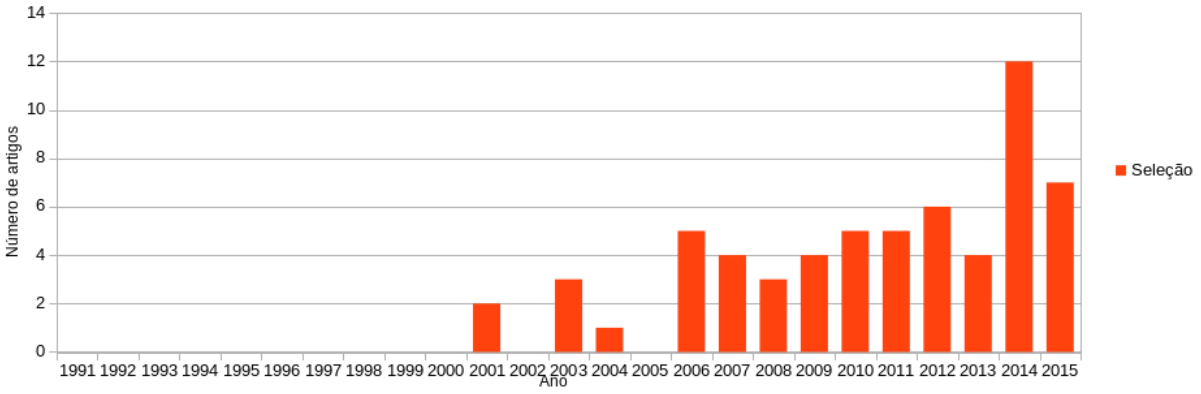



**Figura 3.4** Artigos com publicação de software acadêmico selecionados na revisão de literatura.

da URL informada de alguma forma, significando que as URL informada pelos autores estão ativas e online.

### 3.7.2 Q2 - Os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM estão disponíveis para download?

Entre os 60 projetos selecionados, 24 não estão disponíveis para download, ou seja, apesar dos autores informarem no estudo uma URL, indicarem que o software está disponível nesta URL para download, a URL não mais está disponível e o software não pode ser encontrado. Isso significa que 40% dos projetos de software acadêmico de análise estática publicados nas conferências de Engenharia de Software ASE e SCAM não estão disponíveis ou estão disponíveis em outra URL.

Os demais 36 projetos, que equivalem a 60% do conjunto total, estão disponíveis para download e é possível obter uma cópia destes projetos na URL indicada pelos seus autores. Um resumo deste total por ano é apresentado na Tabela 3.5.

### 3.7.3 Q3 - É possível ter acesso ao código-fonte dos projetos de software de análise estática publicados nas conferências ASE e SCAM?

Do conjunto de 36 projetos de software disponíveis para download, apenas 2 não disponibilizam o código-fonte. Tais projetos – FaultBuster e Loopfrog – são disponibilizados gratuitamente mas apenas em formato binário. A Tabela 3.6 traz um resumo das linguagens de programação usadas para a implementação dos projetos de software.

### 3.7.4 Q4 - Os projetos de software com código-fonte disponível usam licenças de software livre?

Dentre os 34 projetos que disponibilizaram o código-fonte, 13 não informaram licença alguma e 21 informaram licenças de software livre. A Tabela 3.7 resume as licenças utilizadas e o número de projetos em cada uma.

**Tabela 3.5** Número de projetos disponível para download por ano.

| Ano | Total | Disponível | Indisponível |
|---|---|---|---|
| 2001 | 2 | 1 | 1 |
| 2003 | 3 | 1 | 2 |
| 2004 | 1 | 0 | 1 |
| 2006 | 5 | 4 | 1 |
| 2007 | 4 | 1 | 3 |
| 2008 | 2 | 1 | 1 |
| 2009 | 4 | 2 | 2 |
| 2010 | 5 | 3 | 2 |
| 2011 | 5 | 2 | 3 |
| 2012 | 6 | 2 | 4 |
| 2013 | 4 | 2 | 2 |
| 2014 | 12 | 11 | 1 |
| 2015 | 7 | 6 | 1 |
| **Total** | 60 | 36 | 24 |

**Tabela 3.6** Linguagem de programação dos projetos com código-fonte disponível.

| Linguagem de programação | Número de projetos |
|---|---|
| C | 6 |
| C++ | 2 |
| Erlang | 1 |
| Java | 18 |
| Javascript | 2 |
| Perl | 1 |
| Python | 2 |
| Rascal | 1 |
| Ruby | 1 |
| **Total** | 34 |

Todo software com o código-fonte disponível pode ser adaptado para uso pessoal sem qualquer restrição prévia pelos seus autores, mas os 13 projetos sem licença definida impedem eventuais cientistas a publicar suas modificações e redistribuir qualquer melhoria sem prévia solicitação aos autores originais.

Considerando o total de 60 projetos selecionados neste estudo, apenas 35% (21 projetos) podem ser adaptados; os demais com código-fonte disponível, 21% (13 projetos) podem ser adaptados, mas as melhorias ou correções não podem ser redistribuídas sem autorização prévia dos autores.

## 3.8 AMEAÇAS À VALIDADE

O passo de triagem automática dos artigos no processo de revisão de literatura pode ter excluído do processo de revisão algumas publicações com software acadêmico de análise

**Tabela 3.7** Licenças de software dos projetos com código-fonte disponível.

| **Licença de software** | **Número de projetos** |
|---|---|
| Affero General Public License | 1 |
| Apache License | 2 |
| BSD License | 4 |
| Eclipse Public License | 3 |
| FrontEndART Software Ltd | 1 |
| GNU General Public License | 6 |
| GNU Lesser General Public License | 1 |
| Illinois/NCSA Open Source License | 2 |
| SAnToS Laboratory Open Academic License | 1 |
| (sem licença) | 13 |
| **Total** | 34 |

estática, não incluindo algum software acadêmico publicado no conjunto total de projetos selecionados.

O escopo do estudo foi reduzido ao domínio de análise estática, portanto as conclusões e interpretações estão limitadas neste contexto. Acreditamos que outros domínios de aplicação possuem características semelhantes, mas é conhecido que o domínio de aplicação pode ser um fator de influência relevante nas características dos produtos de software. Por exemplo, Zhang et al. (2013), num estudo sobre como o contexto afeta a distribuição de métricas de manutenibilidade, conclui que os maiores fatores de influência são o domínio de aplicação, a linguagem de programação e o número de mudanças.

A leitura dos artigos na fase final de Extração foi realizada apenas pelo autor deste trabalho. Tal ameaça poderia ter sido minimizada com revisão em par, ou por outros pesquisadores.

O contexto selecionado com apenas duas conferências nos leva à possibilidade de ter resultados com viés atrelado a estes dois eventos; é possível que as características dos projetos publicados em outras conferências da Engenharia de Software apresentem situação diferentes.

A avaliação da disponibilidade do software para download usando apenas a URL informada pelos seus autores pode levar a falso-negativos, uma vez que é possível a URL mudar com o tempo, esta ameaça apesar de não ter sido tratada não invalida nossas conclusões, uma vez que o estudo teve como objetivo inicial avaliar primariamente a própria URL indicada pelos autores originais.

## 3.9 CONCLUSÕES

Este estudo selecionou 60 projetos de software de análise estática de código-fonte publicados nas conferências ASE e SCAM. Os autores afirmam que os projetos estão disponíveis para obtenção na URL indicada, mas os resultados da caracterização feita mostram que apenas uma parcela destes projetos continuam disponíveis.

Entre os projetos selecionados, 15 não têm qualquer presença oficial online, com a URL

indicada pelos autores indisponível. Nos 45 projetos com presença online através da URL informada nos artigos, 9 não disponibilizam qualquer artefato relacionado ao software, ou seja, apesar de ser possível acessar a URL, não é possível encontrar o software para download.

Os outros 36 projetos estão disponíveis para download, sendo que a maior parte, 34 projetos, disponibiliza o código-fonte, e apenas 2 estão disponíveis em formato binário. Dentre os 34 projetos com código-fonte disponível, 13 não informam qualquer licença de uso.

É conhecido que os projetos de software desenvolvidos na academia não são ainda reconhecidos pela Ciência como cidadãos de primeira classe, mesmo quando tais projetos são indispensáveis para a compreensão dos estudos em que foram desenvolvidos. Neste estudo mostramos que uma grande parte (40%) de um conjunto de projetos de software acadêmico minimamente identificados pelos seus autores com nome e URL encontra-se hoje indisponível.

No entanto, não sabemos quais os fatores que influenciam um projeto estar ou não disponível, e ainda, como tais projetos são vistos por outros pesquisadores, ou se são utilizados e referenciados por outros autores. Assim planejamos outros estudos para investigar tais questões, apresentados nos próximos capítulos.

Capítulo 4

# RECONHECIMENTO DE SOFTWARE ACADÊMICO DE ANÁLISE ESTÁTICA

Este capítulo apresenta um estudo de caracterização do reconhecimento de software acadêmico de análise estática. Com base em informações sobre citações formais e informais ao software acadêmico de análise estática extraídas de um conjunto de artigos científicos publicados em periódicos ou conferências, as relações entre software acadêmico e artigos científicos são identificadas e classificadas para caracterizar o grau de reconhecimento do software acadêmico.

O estudo realizado teve como ponto de partida o conjunto de 60 projetos de software identificado no estudo sobre publicização do software acadêmico (Capítulo 3). Uma revisão da literatura nas bases da ACM e IEEE encontrou 806 artigos com menções do tipo Cita, Usa e Contribui a este conjunto de software acadêmico.

A seção 4.1 contextualiza o estudo, a seção 4.2 apresenta os conceitos necessários para compreensão do estudo, a seção 4.3 apresenta seu objetivo e questões de pesquisa. A seção 4.4 apresenta o planejamento do estudo, e as seções 4.5 e 4.6 descrevem a preparação e execução da coleta de dados. As seções 4.7, 4.8 e 4.9 apresentam a análise de dados, interpretação de resultados e ameaças à validade, respectivamente. Finalmente, a seção 4.10 traça as conclusões do estudo.

## 4.1 MOTIVAÇÃO

Um dos sintomas de *"dysfunctional chaotic churn"* (HOWISON et al., 2015) em software acadêmico é a quantidade elevada de projetos com características e funcionalidades parecidas e comunidades desconectadas e paralelas. Neste cenário, alguns projetos de software acadêmico podem se destacar e obter o reconhecimento de seus pares.

A prática científica de citação formal entre publicações, amplamente adotada e praticada entre cientistas de todas as áreas, não é realidade quando se trata de artefatos digitais, como dados ou código. Em geral, software é citado na literatura acadêmica de maneira informal: em notas de rodapé, seções de agradecimentos e outras seções ao longo

do texto. Usualmente se faz referência ao nome, ocasionalmente cita-se o número da versão e raramente usa-se citação formal.

Qualquer um que trabalhe com software acadêmico saberá que o nível de reconhecimento e retribuição para o software e para quem revisa ou desenvolve o software não é proporcional à sua importância para a Ciência. O sistema de recompensa científico é quase que exclusivamente baseado em publicações e, infelizmente, compartilhar software ainda provê pouco mérito dentro do sistema de recompensa científico, mesmo que o software seja realmente útil para outros pesquisadores (GOBLE, 2014).

No entanto, o desenvolvimento de software acadêmico é uma atividade que consome recursos (tempo, dinheiro e atenção) e afeta a conduta da Ciência, tanto no geral como em campos específicos. O compartilhamento e colaboração no desenvolvimento destes projetos é vista neste estudo como uma importante estratégia para lidar com o baixo orçamento das pesquisas, bem como para mitigar os problemas do limitado tempo dos cientistas e da alta rotatividade entre os grupos de pesquisa.

Esta estratégia de colaboração se torna especialmente interessante entre as pesquisas da Engenharia de Software, onde, a princípio os cientistas possuem acesso ao conhecimento e práticas necessárias para o trabalho colaborativo. Neste estudo, investigamos como os projetos de software acadêmico de análise estática são mencionados em publicações nas bases da ACM e IEEE e se há colaboração entre eles.

## 4.2 FUNDAMENTAÇÃO

Neste estudo, o termo *menção* é usado para tratar, genericamente, qualquer ocorrência de nome de projeto de software acadêmico de análise estática em artigo científico, incluindo tanto citação formal, quanto informal. Uma menção pode estar associada ao uso do software no contexto de outras pesquisas e até mesmo às contribuições ao projeto de software acadêmico. Desse modo, uma menção define uma relação entre um artigo científico e um software acadêmico.

## 4.3 ESCOPO

Esta pesquisa teve como ponto de partida o conjunto de projetos de software identificado no estudo sobre publicização do software acadêmico de análise estática (capítulo 3). O objetivo de pesquisa está definido segundo a estrutura GQM (BASILI; CALDIERA; ROMBACH, 1994).

### 4.3.1 Definição do Objetivo

**Objeto de estudo.** O objeto de estudo são projetos de software de análise estática publicados nas conferências ASE e SCAM, identificados no estudo sobre publicização do software acadêmico (Capítulo 3).

**Propósito.** O propósito deste estudo é caracterizar as menções feitas a projetos de software acadêmico de análise estática.

**Perspectiva.** A perspectiva considerada é a de cientista preocupado com o funcionamento do ecossistema de software acadêmico e interessado em saber quais são os os projetos de software acadêmico de análise estática mais citados na literatura acadêmica.

**Foco de qualidade.** O principal aspecto de qualidade estudado é a citabilidade (SMITH; KATZ; NIEMEYER, 2016) do software acadêmico em termos de número e tipo de menções.

**Contexto.** As menções ao software acadêmico de análise estática devem ser extraídas de artigos científicos publicados até 2017 e disponíveis nas bases ACM e IEEE. Adicionalmente, as menções serão classificadas com base no reconhecimento dado pelo artigo científico ao software acadêmico mencionado (conforme Tabela 4.1, seção 4.6).

### 4.3.2 Sumário da Definição

Analisar os *projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM* com o propósito de *caracterizar menções* com respeito a *citabilidade do software acadêmico* na perspectiva de *cientistas* no contexto de *publicações nas bases ACM e IEEE*.

### 4.3.3 Questões de Pesquisa

Neste estudo, três questões de pesquisa a respeito dos projetos de software acadêmico de análise estática serão investigadas:

**Q1:** Como os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM são *mencionados* em publicações nas bases ACM e IEEE?

Com esta questão queremos saber como os artigos se relacionam com os projetos de software acadêmico, quantos estudos mencionam cada projeto, e qual o tipo de citação de cada menção encontrada.

**Q2:** Os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM são *usados* como apoio metodológico ou como objeto de estudo em publicações nas bases ACM e IEEE?

Com esta questão queremos investigar se há entre os estudos encontrados nas bases ACM e IEEE artigos utilizando os projetos como apoio metodológico ou mesmo como objeto de estudo.

**Q3:** Os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM *recebem contribuições de código-fonte* em publicações nas bases ACM e IEEE?

Com esta questão queremos trazer evidências sobre quanta colaboração é feita aos projetos de software acadêmico de análise estática entre as publicações indexadas nas bases ACM e IEEE.

### 4.3.4 Métricas

Para responder às questões de pesquisas, as seguintes métricas foram usadas:

1. Número de publicações nas bases ACM e IEEE com menção a projetos de software acadêmico de análise estática.
2. Número de publicações nas bases ACM e IEEE com menção a uso de software acadêmico de análise estática.
3. Número de publicações nas bases ACM e IEEE com menção a contribuição de código-fonte em projetos de software acadêmico de análise estática.

## 4.4 PLANEJAMENTO DO ESTUDO

O estudo foi realizado a partir de uma revisão de literatura nas bases ACM e IEEE em busca de menções aos 60 projetos de software acadêmico de análise estática caracterizados pelo estudo apresentado no Capítulo 3. Durante a revisão da literatura, um esquema de caracterização para classificação das menções emergiu e foi documentado; os artigos foram então inspecionados e as menções encontradas foram classificadas segundo este esquema.

A revisão de literatura foi organizada em quatro passos, detalhados a seguir.

### 4.4.1 Passo 1: Busca

Este passo tem como objetivo encontrar artigos nas bases ACM[1] e IEEE[2] a partir das características dos projetos de software acadêmico de análise estática. As duas bases serão pesquisadas através de strings de busca previamente elaboradas para cada um dos projetos.

A elaboração das strings deve ser realizada com o apoio dos dados já coletados e disponíveis para cada projeto de software acadêmico, por exemplo, nome, descrição, autores, URL, entre outros dados. Este processo de criação das strings é realizado de maneira incremental e iterativa, iniciando por uma busca utilizando apenas o nome do projeto, avaliando os resultados, inspecionando o título dos artigos, refinando as strings e repetindo todo o processo até atingir resultados dentro dos critérios desejados, a Listagem A.3 no Apêndice A traz alguns exemplos de strings de busca.

Estes critérios são definidos principalmente pelo número de resultados obtidos na busca. Sabemos, por experiência prática adquirida em buscas exploratórias, que o número de referências a cada projeto na literatura acadêmica não chega à casa da centena; dessa forma, o critério de aceitação adotado para parada do processo de elaboração e refinamento das strings é quando os resultados apresentem um número inferior ou próximo a esta realidade.

A elaboração das strings e a busca com a versão final foi realizada manualmente, usando a busca avançada dos sites das bases pesquisadas, representadas nas Figuras 4.1 e 4.2, com destaque para os campos e elementos de formulário utilizados. É importante

[1] ⟨http://dl.acm.org⟩
[2] ⟨http://ieeexplore.ieee.org⟩

destacar que durante a busca na base IEEE (Figura 4.2) é necessário marcar a opção *Full Text & Metadata* para que a busca considere tanto os metadados quanto o conteúdo dos artigos.


**Figura 4.1** Captura de tela da busca avançada da base ACM com destaque para (A) campo de entrada da string de busca. Cada palavra ou termo informado no campo de busca é pesquisado de maneira exata pela ACM Digital Library.

O resultado da busca final de cada projeto em cada base deve ser exportado e copiado localmente em formato BibTeX e ambas as bases exportam os resultados neste formato. Os resultados finais da busca de todos os projetos serão combinados e agregados num resultado único, sem duplicação de resultados, contendo os metadados de todos os artigos, sem repetição.

Este resultado final, além de eliminar a duplicação de resultados, foi a base para a definição de uma identificação única para cada artigo encontrado; esta identificação será utilizada nos próximos passos para relacionar os projetos aos artigos através das menções encontradas. Por fim, será feito o download de cada artigo em formato pdf para inspeção manual na etapa seguinte.

### 4.4.2 Passo 2: Triagem

No passo 2, uma triagem de artigos relevantes é realizada por meio de inspeção manual de cada artigo associado aos projetos de software acadêmico que foi identificado no passo 1. O critério de inclusão para a seleção de artigos relevantes é a simples ocorrência do nome do projeto de software acadêmico no artigo. Entre os resultados de cada projeto, um artigo deverá ser identificado como relevante para o projeto caso o critério de inclusão seja satisfeito durante a inspeção manual do artigo.

Ao final deste passo, teremos, para cada projeto de software acadêmico, um conjunto de artigos identificados como relevantes que fazem menção ao nome do projeto.

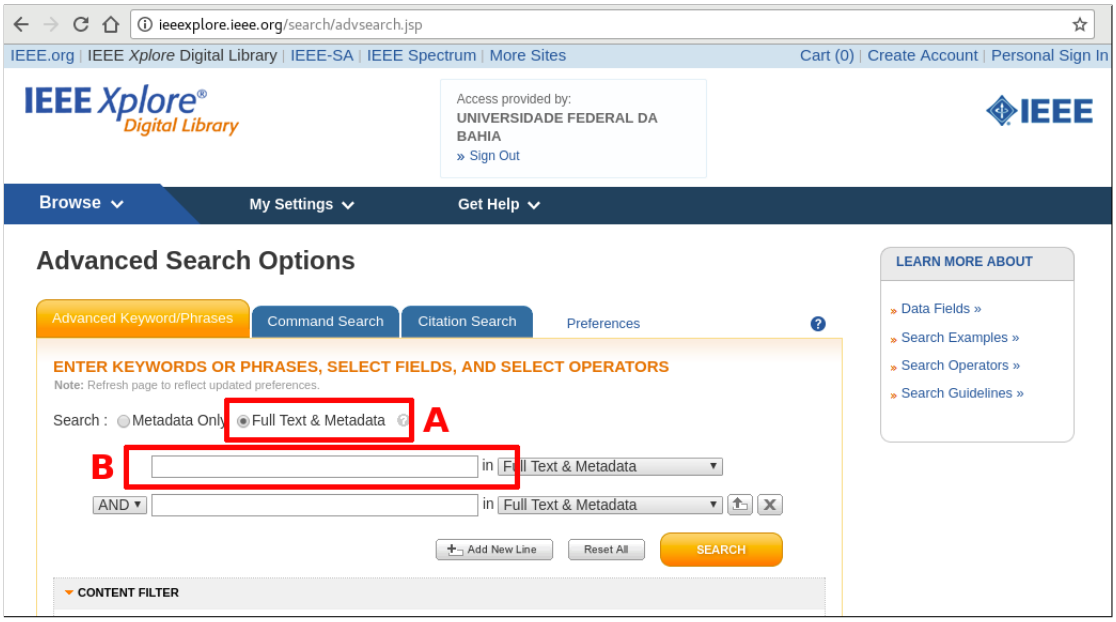



**Figura 4.2** Captura de tela da busca avançada da base IEEE com destaque para (A) opção de busca completa e (B) campo de entrada da string de busca. Cada palavra ou termo informado no campo de busca é pesquisado de maneira exata pela IEEE Xplore Digital Library.

### 4.4.3 Passo 3: Keywording

Neste passo será criado um esquema de codificação para caracterização das menções. O esquema é criado a partir da *identificação do contexto* em que os projetos são mencionados. Para cada menção, toma-se nota sobre como o projeto de software é mencionado no artigo.

As notas são comentários livres descrevendo as menções encontradas no artigo sobre um determinado software. Um artigo científico pode mencionar um software diversas vezes, de diversas formas, desde uma simples menção nos trabalhos relacionados até uma grande contribuição ao software.

O conjunto final de todas as anotações é analisado para definição do esquema de codificação para caracterização das menções. Este esquema deve ser construído com o objetivo de agrupar e caracterizar os tipos de menção encontrados para cada projeto de software em termos da contribuição que a menção traz ao ecossistema de software daquele projeto acadêmico.

### 4.4.4 Passo 4: Extração

Neste último passo da revisão de literatura, utiliza-se o esquema de codificação para caracterização de menções criado no Passo 3 (Keywording) para classificar as menções de cada projeto de software encontradas nos artigos relevantes selecionados no Passo 2 (Triagem).

Para cada artigo que menciona um certo projeto de software foi associada uma informação que indique o tipo de menção entre as opções do esquema de codificação para

caracterização das menções. Os artigos selecionados no estudo anterior (Capítulo 3) foram adicionados aqui, já que sabe-se que eles mencionam os projetos de software acadêmico estudados.

Ao final da revisão de literatura, tivemos, para cada projeto de software, um conjunto de artigos relevantes mencionando o software, com indicação do tipo de menção através do esquema de codificação para caracterização de menções elaboradas no Passo 3 (Keywording).

## 4.5 PREPARAÇÃO

Nesta seção, apresentamos a preparação do estudo para a realização da coleta de dados, incluindo a definição de arquivos e formatos de armazenamento, bem como a implementação de scripts para coleta, análise e transformação. A Figura 4.3 apresenta uma visão geral da solução adotada e da relação entre os passos da revisão de literatura, detalhados a seguir.

Seguindo o planejamento descrito na seção 4.4, no passo de busca, criamos as strings de busca e as registramos nos arquivos `search.yml` de cada projeto. Este arquivo também registra o número de resultados e a data de execução da busca.

Para cada projeto de software acadêmico, outros dois arquivos, `acm.bib` e `ieee.bib`, serão criados para armazenar os dados dos artigos retornados pela busca. Estes dados, de todos os projetos, serão agregados no arquivo `documents/references.bib` e identificados com o campo `id` recebendo valores no formato `p + NÚMERO`, exemplo, `p1`, `p2`, `p3`, ..., `p806`.

Foi implementado o script `bin/ids` para automatizar a definição deste novo campo para cada artigo; ele recebe como entrada um arquivo no formato BibTeX e gera sequencialmente o campo `id` para cada item.

Na preparação para a Triagem, definimos a estrutura do arquivo `references.yml` onde será registrada a informação dos artigos relevantes para cada projeto.

Em paralelo à definição deste arquivo, implementamos um mecanismo de templates, por meio dos scripts `bin/cache` e `bin/render`, para criação automática desses arquivos para cada projeto. Este mecanismo é apresentado em resumo na Figura 4.4. O mecanismo de templates será utilizado em outros pontos de estudo para automatizar e produzir documentos a partir dos dados coletados.

O script `bin/cache` cria o arquivo `cache/dataset.yml` ao agregar todos os dados coletados de todos os projetos a partir dos arquivos `software.yml`, `acm.bib`, `ieee.bib`, `paper.bib`, `references.yml` e `search.yml`. O script `bin/render` lê este arquivo `cache/dataset.yml`, carrega os dados em memória e passa estes dados como parâmetro para os arquivos de template.

Para apoiar o passo de Keywording, adicionamos aos arquivos `references.yml` um novo campo para coletar o tipo de menção feita em cada artigo aos projetos, `mention_type`. Este campo assume um dos valores do esquema de classificação de menções a ser criado durante a execução desse passo.

Finalmente, para apoiar o passo de extração, implementamos arquivos de template para representar os dados coletados em outros formatos para visualização e análise, incluindo documentos em formatos CSV, Markdown e LaTeX. Alguns destes documentos

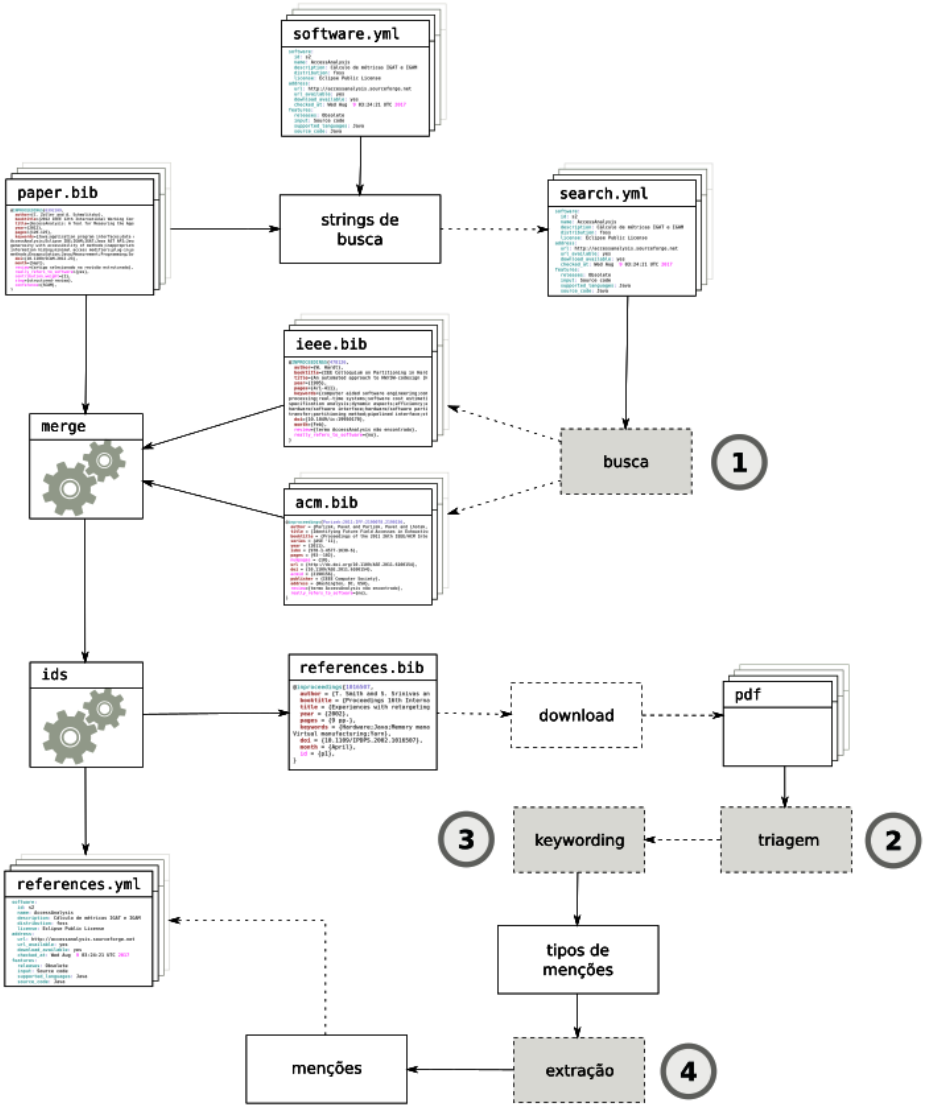



**Figura 4.3** Fluxo de coleta, análise e transformação dos dados.

são incluídos automaticamente neste texto.

## 4.6 COLETA DE DADOS

Seguindo o planejamento e preparação descritos nas seções 4.4 e 4.5, iniciamos a atividade de coleta de dados através da revisão de literatura. A Figura 4.5 apresenta um resumo dos passos da revisão da literatura realizada nas bases ACM e IEEE.

A busca de todos os projetos retornou 897 resultados, 438 na base ACM e 459 na base IEEE. Estes resultados incluem duplicação de artigos, ou seja, um mesmo artigo encontrado nas duas bases, ou, um mesmo artigo encontrado em relação a mais de um projeto, dentre o resultado total.

Ao remover as duplicidades temos um conjunto de 806 artigos únicos. Durante a

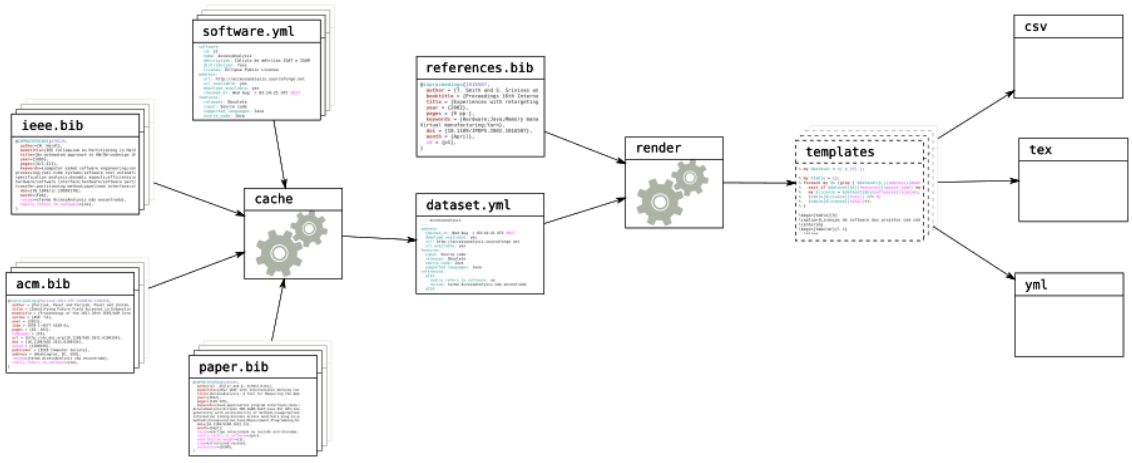



**Figura 4.4** Mecanismo de template para transformar dados em documentos para visualização e análise.

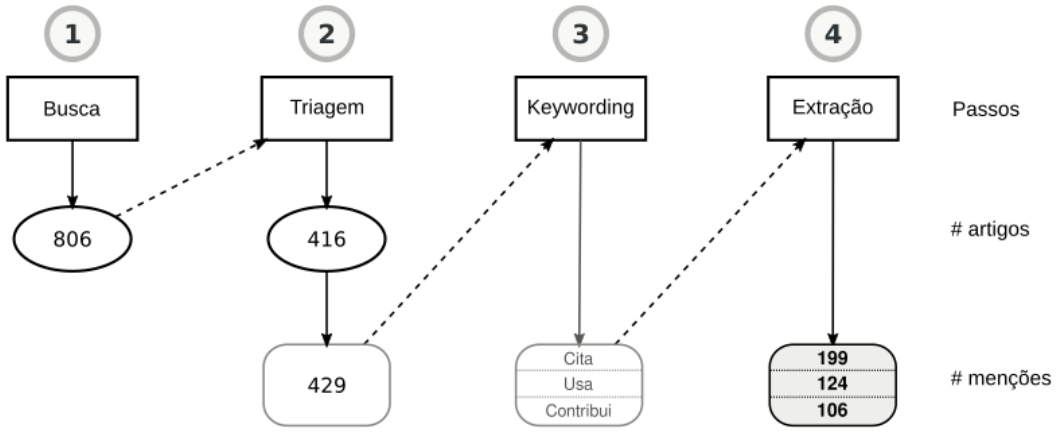



**Figura 4.5** Passos da revisão de literatura realizada nas bases ACM e IEEE.

triagem, os 806 artigos foram inspecionados em busca de menções aos projetos de software. Encontramos 429 menções entre 416 artigos.

Alguns projetos de software possuem nomes comuns que foram encontrados nos artigos mas que não faziam referência ao software; em alguns casos, os nomes apareciam em fórmulas matemáticas ou como parte do texto em contextos não relacionados ao projeto de software acadêmico. O nome do projeto `s1` (2LS), por exemplo, foi encontrado no artigo `p148` (*Statistical Design Framework of Submicron Flip-Flop Circuits Considering Process Variations*) em contexto sem relação alguma com o projeto em si.

Ao final da triagem, tivemos, para cada projeto, um conjunto de artigos que, de fato, mencionam o software acadêmico, mas sem caracterizar o tipo de menção. Na execução do passo de Keywording, a anotação de cada uma das 429 menções foi analisada e o esquema para caracterização foi criado, com três valores distintos que indicam o nível de contribuição da menção ao ecossistema de software do projeto, conforme Tabela 4.1.

A partir do esquema para caracterização das menções, coletamos os valores assumindo

**Tabela 4.1** Esquema para classificação de menções aos projetos software acadêmico.

| Tipo de menção | Explicação |
|---|---|
| Cita | É o mesmo artigo selecionado em `paper.bib` (artigo com "mesmo" conteúdo publicado na "mesma" época); Descreve o software; Menciona o software numa tabela com outros, classifica, cita como exemplo; Menciona em trabalhos relacionados ou em trabalhos futuros. |
| Usa | Avalia ou caracteriza o software; Usa para coleta ou análise de dados; Usa como objeto de estudo; Usa o software como parte de uma solução, implementação, etc; Cria um software derivado sem disponibilizar as contribuições. |
| Contribui | Contribuição inicial publicando o software; Refatora o software; Abre o código de um software que antes era código fechado; Contribuição em código-fonte; Estende o software; Integra o software a outros sistemas, formatos de entrada/saída ou APIs; Implementa parte do software em outro projeto e compara resultados. |

um dos valores da Tabela 4.1. Encontramos 199 menções do tipo Cita, 124 menções do tipo Usa e 106 menções do tipo Contribui. Estes dados são apresentados em resumo na Tabela 4.2; a coluna **Seleção** apresenta os artigos selecionados no estudo anterior (Capítulo 3), a coluna **ACM** e **IEEE** o número de resultados das strings de busca em cada uma das bases, e a coluna **Artigos** contabiliza o número de artigos, excluindo repetições, para cada software.

Tabela 4.2: Número de resultados obtidos na busca de cada projeto de software.

| **ID** | **Seleção** | **Busca** | | **Artigos** | **Menções** | | |
|---|---|---|---|---|---|---|---|
| | | **ACM** | **IEEE** | | **Cita** | **Usa** | **Contribui** |
| `s1` | 1 | 11 | 26 | 38 | - | - | 1 |
| `s2` | 1 | 5 | 3 | 8 | 1 | - | 1 |
| `s3` | 1 | 10 | 6 | 15 | 3 | - | 1 |
| `s4` | 1 | 4 | 6 | 9 | 4 | 3 | 2 |
| `s5` | 1 | 4 | 3 | 6 | 2 | - | 3 |
| `s6` | 1 | 6 | 2 | 7 | 4 | - | 1 |
| `s7` | 1 | 16 | 9 | 25 | 9 | - | 1 |
| `s8` | 1 | 7 | 7 | 13 | 3 | 1 | 2 |
| `s9` | 1 | 4 | 4 | 8 | - | - | 5 |
| `s10` | 1 | 12 | 4 | 14 | 1 | 2 | 2 |
| `s11` | 1 | 2 | 2 | 3 | 2 | - | 1 |

continua na próxima página

| ID | Seleção | Busca | | Artigos | Menções | | |
|---|---|---|---|---|---|---|---|
| | | ACM | IEEE | | Cita | Usa | Contribui |
| s12 | 1 | 6 | 1 | 6 | 1 | - | 1 |
| s13 | 1 | 6 | 3 | 9 | - | - | 1 |
| s14 | 1 | 1 | 1 | 2 | 1 | - | 1 |
| s15 | 1 | 15 | 4 | 16 | 4 | - | 1 |
| s17 | 1 | - | 1 | 1 | - | - | 1 |
| s16 | 1 | 1 | 4 | 5 | 1 | 1 | 2 |
| s18 | 1 | 23 | 24 | 47 | - | 1 | 1 |
| s19 | 1 | 20 | 30 | 45 | 14 | 18 | 8 |
| s20 | 1 | 7 | 3 | 10 | - | - | 1 |
| s21 | 1 | 1 | 3 | 3 | - | - | 1 |
| s22 | 1 | 4 | 4 | 5 | 2 | - | 1 |
| s23 | 2 | 2 | 11 | 12 | 3 | 2 | 3 |
| s24 | 1 | - | - | 1 | - | - | 1 |
| s25 | 1 | 20 | 17 | 34 | 5 | 10 | 4 |
| s26 | 1 | 5 | 2 | 7 | 3 | 2 | 2 |
| s27 | 1 | 2 | 4 | 5 | - | 2 | 1 |
| s28 | 1 | 26 | 24 | 50 | 22 | 14 | 4 |
| s29 | 1 | 11 | 5 | 15 | 5 | 1 | 1 |
| s30 | 1 | 4 | 9 | 10 | 2 | 5 | 1 |
| s31 | 1 | 2 | 2 | 3 | - | 1 | 1 |
| s32 | 1 | 19 | 18 | 32 | 8 | - | 6 |
| s33 | 1 | 3 | 3 | 5 | 2 | 2 | 1 |
| s34 | 1 | 1 | 3 | 4 | 1 | - | 1 |
| s35 | 1 | 2 | 1 | 2 | - | - | 1 |
| s36 | 1 | 17 | 20 | 34 | 1 | - | 1 |
| s37 | 1 | 2 | 5 | 7 | 3 | 1 | 3 |
| s38 | 1 | 29 | 20 | 42 | 17 | - | 1 |
| s39 | 1 | 3 | 10 | 11 | 1 | - | 1 |
| s40 | 1 | 3 | 8 | 11 | 1 | 5 | 3 |
| s41 | 1 | - | - | 1 | - | - | 1 |
| s42 | 1 | 1 | 4 | 5 | - | - | 1 |
| s43 | 2 | - | 2 | 2 | - | - | 2 |
| s44 | 1 | 3 | 1 | 3 | 1 | - | 1 |
| s45 | 1 | 5 | 10 | 14 | - | - | 2 |
| s46 | 1 | 16 | 12 | 24 | 9 | 3 | 1 |
| s47 | 1 | 6 | 7 | 13 | - | - | 1 |
| s48 | 1 | 3 | 2 | 4 | 1 | - | 1 |
| s49 | 1 | 1 | 4 | 5 | 1 | 2 | 2 |
| s50 | 1 | 1 | 1 | 2 | - | - | 1 |
| s51 | 1 | 2 | 5 | 7 | 1 | 2 | 1 |
| s52 | 1 | 9 | 34 | 41 | 14 | 23 | 3 |

continua na próxima página

| ID | Seleção | Busca | | Artigos | Menções | | |
|---|---|---|---|---|---|---|---|
| | | ACM | IEEE | | Cita | Usa | Contribui |
| s53 | 1 | 10 | 5 | 13 | 1 | 2 | 1 |
| s54 | 1 | 2 | 2 | 4 | 1 | 1 | 1 |
| s55 | 1 | 3 | 8 | 12 | - | - | 1 |
| s56 | 1 | 23 | 21 | 38 | 15 | 5 | 1 |
| s57 | 1 | 4 | 6 | 8 | 4 | - | 1 |
| s58 | 1 | 8 | 4 | 11 | 5 | 5 | 1 |
| s59 | 1 | 25 | 12 | 36 | 16 | 10 | 7 |
| s60 | 1 | - | 7 | 7 | 4 | - | 1 |

## 4.7 ANÁLISE DOS DADOS

A análise de dados foi realizada considerando dados de 60 projetos desenvolvidos e publicados na literatura acadêmica de Engenharia de Software, informações sobre diversas formas de menção ao nome destes projetos e 416 artigos distintos encontrados nas bases ACM e IEEE mencionando estes projetos.

A busca nas bases ACM e IEEE retornou uma média de 13.4 artigos por projeto de software acadêmico. Os projetos `s24` (GUIZMO) e `s41` (protopurity) não foram encontrados em nenhuma das bases. Os projetos `s17` (e-munity), `s43` (PtYasm) e `s60` (XOgastan) não foram encontrados na base ACM. Mesmo os artigos selecionados no estudo anterior (Capítulo 3) não foram encontrados nas busca nas bases ACM ou IEEE.

Entre os projetos com resultados acima da média, a maior parte está disponível para download, sendo 11 com acesso e 7 sem acesso. Do mesmo modo, entre os projetos com resultados abaixo da média, a maior parte, 25 projetos, está disponível para download e 16 não estão disponíveis para download. Não foi encontrada correlação entre menções e disponibilidade.

Na triagem, observou-se que:

- Entre os resultados da busca, apenas 51% dos artigos são relevantes e fazem menção aos projetos, com uma média de 6.9 artigos por projeto e 7.2 menções por projeto;
- Entre os projetos com menções acima da média, 9 estão disponíveis para download e 6 não estão disponíveis. Entre os projetos com menções abaixo da média, 25 possuem download disponível e 20 não estão disponíveis.

Durante o Keywording notamos que os três tipos de menção – Cita, Usa e Contribui – que emergiram das avaliações a respeito do contexto em que as menções ocorrem, possuem a natureza de uma escala ordinal na classificação de Stevens (1946), sendo este valor uma representação da contribuição que cada menção causa no software acadêmico ou no seu ecossistema, sendo o tipo Cita o primeiro valor de menor peso e o tipo Contribui com o maior peso de contribuição ao software e ao seu ecossistema.

Durante a extração, observou-se que, entre os 60 projetos, 13 projetos (`s1`, `s13`, `s17`, `s20`, `s21`, `s24`, `s35`, `s41`, `s42`, `s43`, `s47`, `s50`, `s55` ) não foram encontrados em menções após a publicação inicial, sendo a única publicação aquela localizada no arquivo `paper.bib`,

referente a seleção do software no estudo apresentado no Capítulo 3. Os demais 47 projetos são mencionados em publicações nas bases ACM e IEEE. Deste grupo, apenas 17 projetos (`s4`, `s5`, `s8`, `s9`, `s10`, `s19`, `s23`, `s25`, `s26`, `s28`, `s32`, `s37`, `s40`, `s45`, `s49`, `s52`, `s59` ) recebem contribuições através de 43 menções.

A Figura 4.6 apresenta uma visualização da linha do tempo de cada software acadêmico em relação a menções encontradas nas bases ACM e IEEE.

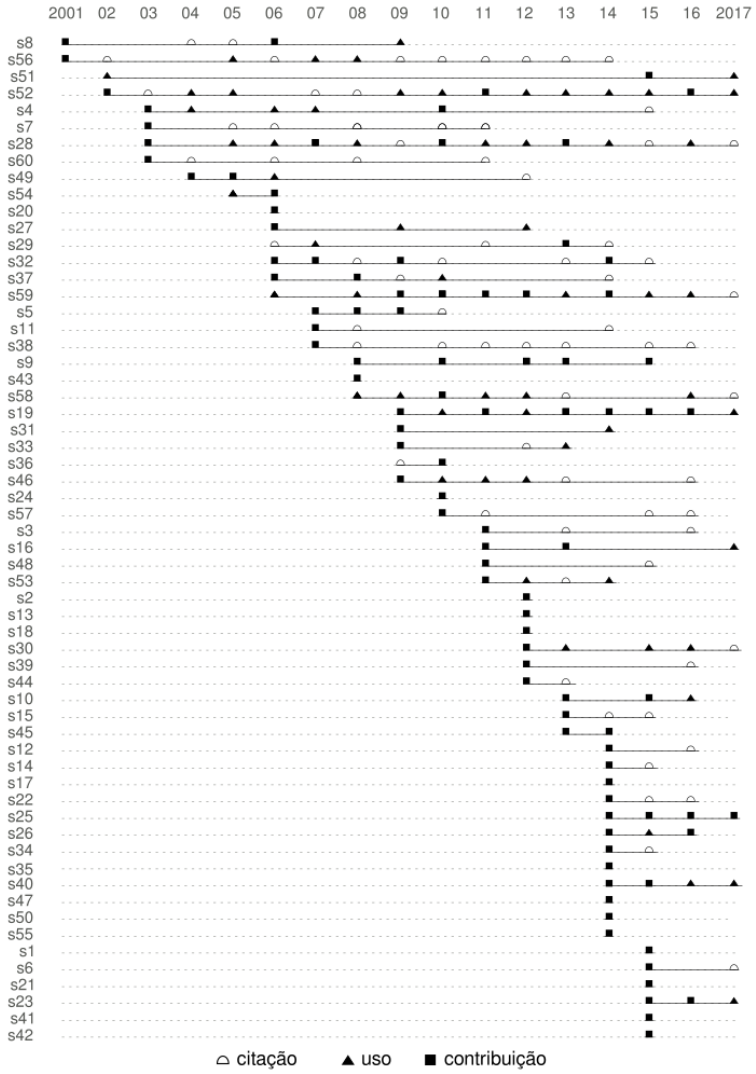



**Figura 4.6** Linha do tempo de menções (uso e contribuição) aos projetos nas bases ACM e IEEE. Os pontos no gráfico indicam a menção de maior valor na escala ordinal para o software no ano indicado.

## 4.8 INTERPRETAÇÃO DOS RESULTADOS

### 4.8.1 Q1 - Como os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM são mencionados em publicações nas bases ACM e IEEE?

Os 60 projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM são mencionados 429 vezes em 416 artigos distintos, estas menções foram encontradas em buscas nas bases ACM e IEEE realizadas entre os meses de Julho e Agosto de 2017. 2 projetos não foram encontrados nas buscas realizadas na ACM e IEEE.

Entre todas as menções emergiram três tipos distintos em relação ao nível de contribuição ao ecossistema dos projetos: Cita, Usa e Contribui. A maior parte das menções encontradas é do tipo Cita (199 menções), em seguida o tipo de menção Usa com 124 ocorrências, e por fim, com menor número, o tipo de menção Contribui com 106 ocorrências.

### 4.8.2 Q2 - Os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM são usados como apoio metodológico ou como objeto de estudo em publicações nas bases ACM e IEEE?

Encontramos 124 menções usando os projetos como apoio metodológico para coleta ou análise de dados, como objeto de estudo, ou como ponto de partida para implementação de outras soluções de pesquisa.

Estas menções do tipo Uso estão associadas a 26 dos 60 projetos de software, indicando que menos da metade dos projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM são utilizados em pesquisas nas bases ACM e IEEE.

### 4.8.3 Q3 - Os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM recebem contribuições de código-fonte em publicações nas bases ACM e IEEE?

Encontramos 106 menções contribuindo com os projetos, este total inclui os artigos publicando os projetos de software pela primeira vez, ou seja, inclui os artigos que deram origem ao conjunto de 60 projetos de software registrados nos arquivos `paper.bib`.

Ao analisar apenas contribuições posteriores a publicação inicial do projeto, ou seja, artigos e estudos contribuindo com código-fonte aos projetos após a sua publicação inicial, encontramos 43 menções a um conjunto de 17 projetos de software.

Ou seja, apenas 28% dos projetos (`s4`, `s5`, `s8`, `s9`, `s10`, `s19`, `s23`, `s25`, `s26`, `s28`, `s32`, `s37`, `s40`, `s45`, `s49`, `s52`, `s59`) recebem contribuição em código-fonte de estudos após a publicação inicial, ao menos entre estudos encontrados nas bases ACM e IEEE.

## 4.9 AMEAÇAS À VALIDADE

A inspeção manual dos artigos na revisão de literatura em busca de caracterizar as menções e seus tipos foi realizada apenas pelo autor desta dissertação, isto pode levar a inconsistência na interpretação das menções em relação aos projetos investigados, uma

vez que a inspeção é realizada num processo estritamente subjetivo, esta ameaça não foi tratada.

A busca realizada apenas nas bases da ACM e IEEE pode ter deixado fora dos resultados possíveis artigos mencionando os projetos investigados, uma vez que é possível que existam publicações não indexadas por estas bases. Apesar disso, todas as conclusões deste trabalho estão estreitamente associadas a este recorte e não sugerimos serem generalizadas, desta forma esta ameaça não conflita com as conclusões do estudo.

## 4.10 CONCLUSÕES

Este estudo inspecionou 806 artigos encontrados nas bases ACM e IEEE através de busca avançada, usando características de 60 projetos de software acadêmico de análise estática. Durante a inspeção foram encontradas 429 menções dos tipos Cita, Usa ou Contribui.

Estes tipos emergiram da análise sobre como os projetos são mencionados nos artigos. Entre todas as menções, 199 são do tipo Cita, 124 Usa e 106 Contribui. As menções do tipo Contribui incluem os artigos iniciais que publicaram os projetos pela primeira vez pelos autores originais; ao excluir estes artigos, encontramos apenas 43 menções do tipo Contribui.

Este número de menções está relacionado a apenas 28% do conjunto total de projetos. Isto indica que não parece haver uma prática de contribuição entre os pesquisadores em relação a código-fonte dos seus projetos de software, ao menos no contexto de análise estática.

Não encontramos relação entre a disponibilidade de download, forma de distribuição ou linguagem de programação e o número de menções. Entre as linguagens com maior contribuição temos Java, C, Erlang e Rascal.

Acreditamos que o número de menções, seja de Uso ou Contribuição, está intimamente ligado à relevância prática do projeto e aos atributos de qualidade, como usabilidade, portabilidade, facilidade de instalação, entre outros. Além, claro de questões sociais como por exemplo o *networking* do grupo de pesquisa responsável pelo projeto. No entanto, não investigamos estes atributos neste estudo.

Entretanto, o estágio de evolução no ciclo de vida de um software é um aspecto de grande influência na adoção e uso, e potencialmente também no número de contribuições. Desta forma, planejamos um terceiro estudo para investigar este aspecto entre os projetos estudados.

Capítulo 5

# CICLO DE VIDA DE SOFTWARE ACADÊMICO DE ANÁLISE ESTÁTICA

Este capítulo apresenta um estudo para caracterização do estágio de evolução de software acadêmico de análise estática em termos de número de lançamentos (releases) e módulos no código-fonte.

O estudo avaliou o ciclo de vida de software acadêmico coletando dados do código-fonte através de análise estática com o apoio da ferramenta Analizo, que é um software acadêmico de análise estática que não aparece neste estudo uma vez que não foi publicado em nenhuma das conferências ASE ou SCAM (TERCEIRO et al., 2010). Esta análise revelou que a maior parte dos projetos estão em estágio inicial de desenvolvimento ou encerrados.

A seção 5.1 contextualiza o estudo, a seção 5.2 apresenta os conceitos necessários para compreensão do estudo, a seção 5.3 descreve o objetivo e apresenta as questões de pesquisa, a seção 5.4 apresenta um planejamento do estudo, as seções 5.5 e 5.6 apresentam detalhes sobre a preparação e execução da coleta de dados, as seções 5.7 e 5.8 apresentam a análise e interpretação dos dados e a seção 5.10 traça as conclusões finais deste estudo.

## 5.1 MOTIVAÇÃO

Os projetos de software acadêmico caracterizados neste estudo são considerados como projetos de software livre sob a perspectiva do modelo de ciclo de vida proposto por Capiluppi et al. (2007); uma vez que não há entre eles projetos com ciclos de vida típicos de produtos de software comerciais de propriedade exclusiva de empresas.

A caracterização de projetos de software acadêmico de análise estática quanto ao ciclo de vida será feita sob a perspectiva de cientistas desenvolvedores e usuários finais de software interessados em compreender em que estágio de desenvolvimento e evolução estão os projetos do ecossistema de software acadêmico de análise estática. Tal informação pode ser útil ao se tomar decisão de adotar um certo software acadêmico para uso ou mesmo como objeto de contribuição (ver Capítulo 4).

Neste estudo, *módulo* refere-se às unidades que compõem um sistema de software. Paradigmas e linguagens de programação possuem uma ou mais construções que fazem o papel de módulo – "classes", "aspectos", "tipos abstratos de dados", ou "arquivos-fonte" – com propriedades que poderão servir para caracterização do software acadêmico.

## 5.2 FUNDAMENTAÇÃO

Neste estudo, o número de módulos será utilizado como fator de avaliação do estágio de evolução do software acadêmico, será medida através da análise estática do código-fonte. Estudos apontam que o número de módulos de um software possui a tendência de estabilizar com o tempo e dessa forma esta medida pode ser utilizada para caracterizar o seu estágio de evolução (CAPILUPPI et al., 2007).

## 5.3 ESCOPO

Projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM são nosso principal objeto de estudo. Queremos saber em qual estágio de evolução estão os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM.

O objetivo da pesquisa está definido segundo a estrutura GQM (BASILI; CALDIERA; ROMBACH, 1994).

### 5.3.1 Definição do Objetivo

**Objeto de estudo.** O objeto de estudo são projetos de software acadêmico de análise estática publicados em artigos científicos e seu estágio de evolução no ciclo de vida de software, conforme definido na Seção 5.1.

**Propósito.** O propósito do estudo é caracterizar em qual estágio de evolução encontra-se cada software acadêmico de análise estática. O estudo contribuirá para a análise da desordem disfuncional caótica no domínio de análise estática.

**Perspectiva.** A perspectiva considerada é a do cientista desenvolvedor e usuário final, isto é, o pesquisador que deseja saber em que estágio de evolução estão os projetos de software acadêmico do domínio de interesse. A perspectiva a ser considerada também é a do pesquisador que deseja conhecer ferramentas de análise estática para uso em sua pesquisa.

**Foco de qualidade.** O principal foco de qualidade estudado é evolução do software acadêmico de análise estática, com ênfase nos aspectos de lançamentos e versões do projeto, especialmente sobre o tamanho do software em número de módulos.

**Contexto.** O estudo foi conduzido com projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM.

### 5.3.2 Sumário da Definição

Analisar os *projetos de software acadêmico de análise estática* publicados com o propósito de *caracterizá-los* com respeito ao seu *estágio de evolução no ciclo de vida* na perspectiva de *cientistas usuários finais e desenvolvedores de software* no contexto das *conferências de Engenharia de Software ASE e SCAM.*

### 5.3.3 Questões de Pesquisa

Neste estudo as seguintes questões de pesquisa, a respeito dos projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM, e seu ciclo de vida serão investigadas:

**Q1:** Em qual estágio de evolução, segundo o modelo proposto por Capiluppi et al. (2007), estão os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM em relação ao seu ciclo de vida?

Queremos saber com esta questão quais projetos de software acadêmico de análise estática continuam evoluindo após a sua publicação e em qual estágio de evolução se encontram.

### 5.3.4 Métricas

Para responder às questões de pesquisas, as seguintes métricas serão usadas:

1. Número de lançamentos de cada projeto com informações sobre versão e data
2. Número de lançamentos com código-fonte disponível para download
3. Número de módulos no código-fonte de cada lançamento/versão

## 5.4 PLANEJAMENTO DO ESTUDO

Este estudo foi realizado a partir de uma análise preliminar dos dados existentes de cada software, coletados previamente nos Capítulos 3 e 4. O objetivo de tal análise foi identificar quais projetos são candidatos a terem mais dados coletados para compor a caracterização do seu ciclo de vida. Ao final do estudo, cada projeto será caracterizado em um dos estágios de evolução apresentados na Seção 5.1.

Na análise preliminar deve-se identificar entre os projetos de software acadêmico quais possuem disponibilidade de download, ou ao menos, presença oficial online com informações do projeto sobre lançamentos, para coleta dos seguintes dados:

- Número da versão
- Data do lançamento
- URL para download do código-fonte

As fontes para coleta de dados são o site do projeto, manuais, código-fonte e repositórios. Usualmente será utilizada mais de uma fonte para compor todas as informações sobre os lançamentos de um certo software. Não é raro que os projetos mudem a forma de lançamento, passando de um formato para outro, de uma plataforma a outra, entre outras mudanças; essas informações devem ser investigadas a fim de encontrar o máximo de informação possível.

Os projetos sem presença oficial online ou sem disponibilidade de download, ou seja, com a URL indicada pelos seus autores originais indisponível, não terão, consequentemente, informações sobre os lançamentos; dessa forma, estes projetos serão caracterizados como software com ciclo de vida encerrado (*Encerrado*), uma vez que eles estão indisponíveis e inacessíveis a qualquer potencial usuário interessado em utilizar o software.

Os demais projetos, aqueles com presença online, e com código-fonte disponível, tiveram o código-fonte de cada lançamento copiado localmente para análise estática e coleta da métrica representando o tamanho do software em número de módulos. Foi feito o download do código-fonte de cada versão e a ferramenta de análise estática Analizo foi utilizada para extrair as informações do código-fonte.

Analizo é software livre, distribuído sob a licença GNU General Public License versão 3. Seu código-fonte, bem como pacotes binários, manuais e tutoriais podem ser obtidos em ⟨http://www.analizo.org⟩. Analizo é escrito em Perl, sua última versão 1.19.1 lançada em 01 de Setembro de 2016 foi a versão utilizada neste estudo.

Apenas os projetos escritos nas linguagens de programação suportadas pelo Analizo – C, C++ C# e Java – serão analisados. O código-fonte dos projetos escritos em outras linguagens de programação não serão analisados; estes projetos serão caracterizados com base em informações disponíveis ou serão definidos como *Desconhecido* quando não for possível determinar o estágio de evolução com as informações disponíveis. Projetos com apenas um lançamento serão considerados projetos em estágio inicial de desenvolvimento (*Initial development*).

## 5.5 PREPARAÇÃO

Seguindo o planejamento apresentado na seção 5.4, preparamos os arquivos e templates necessários para realizar a coleta e análise dos dados, o conjunto de projetos de software acadêmico de análise estática e os dados coletados nos Capítulos 3 e 4 serão também utilizados aqui neste estudo.

Para cada projeto foi criado um arquivo `releases.yml` onde serão registradas as informações sobre os lançamentos de cada software. Os arquivos foram criados manualmente. Para os projetos sem informações sobre lançamentos o arquivo será criado com um único campo indicando indisponibilidade com o seguinte conteúdo `source: unavailable`.

Após criar os arquivos para a coleta de informações sobre os lançamentos, implementamos um arquivo de template em `templates/releases-table.tex.epl` para agregar os dados e apresentá-los na seção 5.6, Tabela 5.1.

O Analizo e suas dependências foram instaladas, um script para automatizar a execução da ferramenta *analizo metrics* para coletar o número de módulos de cada lançamento foi

desenvolvido. Este script, disponível em `bin/run-analizo`, foi implementado em linguagem Perl e será apresentado em mais detalhes no Apêndice A.

As métricas serão agregadas no arquivo CSV `documents/metrics.csv` gerado pelo template `templates/metrics.csv.epl`, este template consulta os arquivos com os dados de cada lançamento criado pela execução do *analizo metrics* e exibe, para cada projeto, todos os lançamentos e suas métricas.

Por fim, definimos uma estrutura para adicionar aos arquivos `software.yml` de cada projeto a informação final sobre o atual estágio de evolução em relação ao ciclo de vida do software.

O campo `review` foi utilizado para registrar anotações em forma livre sobre a análise dos dados, a avaliação de cada projeto é realizada inspecionando manualmente os dados coletados, incluindo dados trazidos pelos estudos anteriores, o campo `stage` contém o resultado final da caracterização do ciclo de vida de cada software, este campo assumirá um dos estágios de evolução conforme o modelo de evolução apresentado em 5.1.

## 5.6 COLETA DE DADOS

Seguindo o planejamento e preparação descritos nas Seções 5.4 e 5.5, iniciamos a coleta dos dados, incluindo a coleta do número de lançamentos de cada projeto, informações e código-fonte de cada lançamento, bem como a métrica representando o número de módulos do software em cada versão. Coletamos os dados sobre lançamentos, fizemos o download do código-fonte e executamos a ferramenta `analizo metrics` para cada projeto, em cada versão.

Foram encontrados ao todo 303 lançamentos. Coletamos para cada lançamento, quando disponível, número da versão, data de lançamento e URL para download do código-fonte. Destes lançamentos, 211 possuem informação para download, 5 destes releases não têm código-fonte, apenas binários do software `s33`. Assim, deste total extraímos métricas de código-fonte de 206 lançamentos.

Este total de métricas foi extraído de um conjunto de 24 projetos com lançamentos entre os anos de 2001 e 2017. Os demais projetos não possuem código-fonte disponível ou são escritos em linguagem de programação não suportada pelo Analizo.

Os lançamentos coletados do repositório, código-fonte, sites, devem ser organizados em ordem de data de lançamento, os mais antigos primeiro, numa lista em ordem, sendo o último release o lançamento mais recente. Isto é necessário uma vez que o número de versão de cada projeto não segue o mesmo padrão, alterando de formatos entre releases distintos, a data de lançamento de cada release também nem sempre está disponível, então a coleta deve garantir que haja alguma ordem, então no momento da coleta deve-se avaliar a ordem dos lançamentos e registrar no arquivo `releases.yml` na ordem correta.

## 5.7 ANÁLISE DOS DADOS

Analisamos as métricas número de lançamentos, lançamentos com código-fonte disponível e número de módulos de cada lançamento, analisamos também as menções do tipo Uso e Contribuiçao para determinar se há atividade recente para os projetos sem lançamentos

atuais.

Consideramos projetos em fase de serviço ou estagnação (*Manutenção*) aqueles que apresentem variação pequena entre os lançamentos em relação ao tamanho do projeto. Em termos de número de módulos, abaixo de 10% de variação ao longo das versões será considerado *Manutenção*, acima de 10% está em evolução (*Evolução*). Este número 10% foi baseado no trabalho de Capiluppi et al. (2007).

Cinco projetos não puderam ser caracterizados em relação ao estágio de evolução. A Tabela 5.1 apresenta a caracterização final de todos os projetos incluindo o número total de lançamentos e o período em que estes lançamentos aconteceram; a coluna **Módulos** apresenta o número de módulos da versão mais recente.

Tabela 5.1: Número e período dos lançamentos de software acadêmico de análise estática, número de módulos do último lançamento e estágio de evolução segundo modelo *Staged model* (CAPILUPPI et al., 2007).

| **ID** | **Lançamentos** | | **Módulos** | **Evolução** |
|---|---|---|---|---|
| | **Total** | **Período** | | |
| s1 | 7 | 2015-2017 | 671 | Evolução |
| s2 | 4 | 2010-2012 | 91 | Desenvolvimento inicial |
| s3 | - | - | - | Encerrado |
| s4 | - | - | - | Encerrado |
| s5 | - | - | - | Encerrado |
| s6 | 36 | 2013-2017 | 2096 | Manutenção |
| s7 | 7 | 2001-2004 | 93 | Desenvolvimento inicial |
| s8 | 4 | 2005 | 17 | Desenvolvimento inicial |
| s9 | - | - | - | Encerrado |
| s10 | 4 | - | 106 | Desenvolvimento inicial |
| s11 | - | - | - | Encerrado |
| s12 | 2 | 2013-2014 | - | Desenvolvimento inicial |
| s13 | - | - | - | Encerrado |
| s14 | 1 | 2014 | - | Desenvolvimento inicial |
| s15 | - | - | - | Encerrado |
| s17 | 1 | 2014 | - | Desenvolvimento inicial |
| s16 | 1 | - | 244 | Desenvolvimento inicial |
| s18 | 25 | 2015-2017 | 2504 | Manutenção |
| s19 | - | - | - | Encerrado |
| s20 | - | - | - | Encerrado |
| s21 | 1 | - | - | Desenvolvimento inicial |
| s22 | 1 | 2017 | - | Desenvolvimento inicial |
| s23 | - | - | - | Encerrado |
| s24 | 1 | - | 306 | Descontinuado |
| s25 | 3 | 2013-2015 | 190 | Desenvolvimento inicial |

continua na próxima página

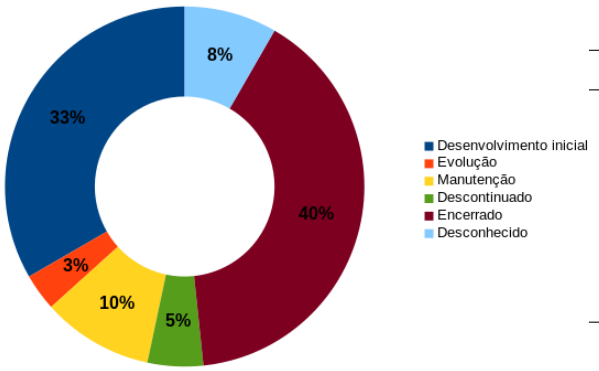
| ID | Lançamentos | | Módulos | Evolução |
|---|---|---|---|---|
| | Total | Período | | |
| s26 | 22 | 2013-2017 | 1580 | Manutenção |
| s27 | 1 | - | - | Desenvolvimento inicial |
| s28 | 26 | 2013-2017 | 57 | Manutenção |
| s29 | 1 | 2013 | 177 | Desenvolvimento inicial |
| s30 | - | - | - | Encerrado |
| s31 | - | - | - | Encerrado |
| s32 | 1 | 2006 | 545 | Desconhecido |
| s33 | 5 | 2007-2010 | - | Desconhecido |
| s34 | 1 | 2017 | 4059 | Desenvolvimento inicial |
| s35 | 1 | 2017 | 240 | Desenvolvimento inicial |
| s36 | - | - | - | Encerrado |
| s37 | 5 | - | - | Desenvolvimento inicial |
| s38 | - | - | - | Encerrado |
| s39 | - | - | - | Encerrado |
| s40 | 4 | 2011-2014 | - | Desconhecido |
| s41 | - | - | - | Descontinuado |
| s42 | 1 | 2017 | - | Desenvolvimento inicial |
| s43 | 1 | 2008 | 883 | Desenvolvimento inicial |
| s44 | - | - | - | Encerrado |
| s45 | - | - | - | Encerrado |
| s46 | 5 | 2009-2010 | 169 | Desenvolvimento inicial |
| s47 | - | - | - | Encerrado |
| s48 | - | - | - | Encerrado |
| s49 | - | - | - | Encerrado |
| s50 | 6 | 2015-2017 | 143 | Evolução |
| s51 | 18 | 2012-2016 | 268 | Manutenção |
| s52 | 14 | 2011-2015 | 871 | Desenvolvimento inicial |
| s53 | - | - | - | Encerrado |
| s54 | 1 | - | - | Descontinuado |
| s55 | 21 | 2010-2016 | - | Desconhecido |
| s56 | - | - | - | Encerrado |
| s57 | - | - | - | Encerrado |
| s58 | 37 | 2006-2017 | 2769 | Manutenção |
| s59 | 34 | 2013-2015 | - | Desconhecido |
| s60 | - | - | - | Encerrado |

O projeto s59 não foi caracterizado em relação ao estágio de evolução apesar de possuir muitas versões entre 2013 e 2015 por ser escrito em Erlang e esta linguagem não ser suportada pelo Analizo. O mesmo ocorre com os projetos s55, escrito em Perl, e s40, escrito em Rascal. O projeto s33 disponível apenas em binários não possível ser analisado e o projeto s32 possui um histórico de lançamentos bastante confuso, dificultando compreender quais pacotes fazem parte de qual versão ou lançamento.

## 5.8 INTERPRETAÇÃO DOS RESULTADOS

### 5.8.1 Q1 - Em qual estágio de evolução, segundo o modelo proposto por Capiluppi et al. (2007), estão os projetos de software acadêmico de análise estática publicados nas conferências ASE e SCAM em relação ao seu ciclo de vida?

A maior parte dos 60 projetos de software acadêmico de análise estão em em estágio inicial de desenvolvimento ou encerrados, isto equivale a 73% de todos os projetos. Os demais 22% dos projetos estão distribuídos entre projetos em evolução, manutenção ou descontinuado, 5% dos projetos não puderam ser determinados uma vez que não encontrou-se informação suficiente para caracterizar o estágio de evolução, conforme Figura 5.1.




| Estágio | Projetos |
|---|---|
| Desenvolvimento inicial | 20 |
| Evolução | 2 |
| Manutenção | 6 |
| Descontinuado | 3 |
| Encerrado | 24 |
| Desconhecido | 5 |

**Figura 5.1** Número total de projetos identificados em cada estágio de evolução.

Os projetos em estágio de *Desenvolvimento inicial* possuem características de ser software acadêmico do tipo Software Incidental (*Incidental software*) e ter sido feito puramente para apoiar e facilitar pesquisas, enquanto os projetos em estágio *Evolução* e *Manutenção* possuem características de ser software acadêmico do tipo Prática de software paralela (*A parallel software practice*) feito com objetivo de ser utilizado por outros pesquisadores ou do tipo Um subcampo de software (*A Software Sub-field*) onde o próprio software é considerado uma contribuição primária para a Ciência.

## 5.9 AMEAÇAS À VALIDADE

Toda a análise e interpretação dos resultados tomou como base apenas os dados coletados do site, publicações e repositórios dos projetos de software acadêmico de análise estática, o fato de não incluirmos coleta de dados a partir das pessoas envolvidas nos projetos pode enfraquecer alguma conclusão no sentido de que alguns projetos podem ter definições explícitas por parte dos seus desenvolvedores mas que não se refletem ainda nos documentos dos projetos, apesar disso o nosso estudo tem um caráter exploratório, com interesse especial em uma figura geral do domínio de análise estática e não necessariamente trazer evidências sobre os projetos inidivualmente.

## 5.10 CONCLUSÕES

Este estudo coletou informações de 303 lançamentos em relação a 35 projetos de software acadêmico de análise estática. Analisou e extraiu o número de módulos do código-fonte de 206 versões distintas destes projetos.

A partir destes dados caracterizou cada um dos 60 projetos estudados em relação ao estágio de evolução do ciclo de vida do software e identificou que a grande maioria encontra-se em estado inicial de desenvolvimento ou encerrados (73%).

Apenas 2 projetos encontram-se em evoluçao atualmente, com variação no número de módulos nos últimos lançamentos recentes superior a 10%. E 6 projetos em fase de serviço, indicando que são projetos estáveis com atualizações constantes mas com pouca variação na base de código do projeto.

Os dados, interpretações e conclusões deste estudo e dos dois estudos anteriores apresentados nos Capítulos 3 e 4 serão analisados em conjunto e discutidos no capítulo seguinte sob uma mesma perspectiva em relação à sustentabilidade técnica do ecossistema de software acadêmico de análise estática, no recorte de projetos selecionados nas conferências de Engenharia de Software ASE e SCAM, entre menções encontradas nas bases ACM e IEEE, e com análise ao nível de código-fonte dos projetos escritos em C, C++ e Java.

Capítulo

6

# SÍNTESE DE RESULTADOS

Este capítulo apresenta a síntese dos resultados desta pesquisa através de uma discussão guiada pelas questões apresentadas na Seção 6.1.

A Seção 6.2 apresenta uma discussão geral a respeito da sustentabilidade técnica do software acadêmico de análise estática, a partir dos resultados dos estudos apresentados nos Capítulos 3, 4 e 5 sobre a publicização, reconhecimento e ciclo de vida do software acadêmico de análise estática. A Seção 6.3 apresenta recomendações para desenvolvedores e usuários de software acadêmico no sentido de dar corpo a práticas já conhecidas mas que trazem uma grande contribuição a este cenário. Finalmente, a Seção 6.4 apresenta trabalhos relacionados ao software acadêmico em termos de melhorias no processo de desenvolvimento, reconhecimento e sustentabilidade.

## 6.1 QUESTÕES

**Q1** Como evolui o reconhecimento ao software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM?

**Q2** A publicização do software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM influencia o seu reconhecimento?

**Q3** O ciclo de vida do software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM influencia o seu reconhecimento?

**Q4** Qual o tamanho do software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM?

**Q5** Como evolui o tamanho do software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM?

**Q6** É possível replicar ou reproduzir pesquisas que mencionam software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM?

## 6.2 DISCUSSÃO

Considerando que esta pesquisa adotou uma estratégia de estudo de campo em ambiente natural, com alto realismo no contexto estudado e, consequentemente, com baixa generalização, seus resultados e conclusões são específicos ao domínio de análise estática e, em especial, ao conjunto de projetos observados.

### 6.2.1 Q1 - Como evolui o reconhecimento ao software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM?

O software acadêmico tem recebido maior atenção na literatura acadêmica com o passar do tempo; há uma clara evolução no número total de menções a software acadêmico de análise estática em publicações das bases ACM e IEEE, conforme Figura 6.1.

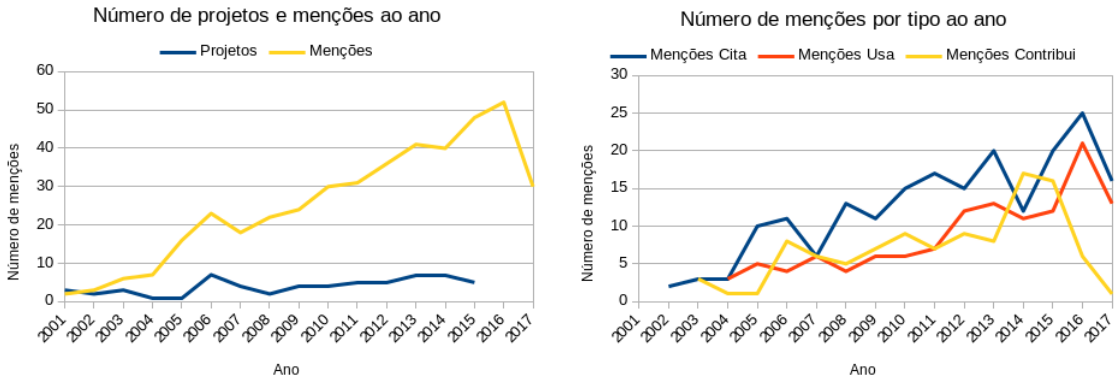



**Figura 6.1** Número de projetos e menções por tipo ao ano.

Este crescimento, no entanto, pode estar sofrendo influência do aumento de projetos, uma vez que há também crescimento no número de projetos a cada ano. Entretanto, ao isolar os dados, mantendo o número de projetos constante ao longo do tempo, há um crescimento de 38% ao ano, conforme Figura 6.2.

Entre os 60 projetos de software acadêmico de análise estática estudados, 13 não possuem reconhecimento acadêmico, ou seja, são mencionados unicamente no artigo original publicado na ASE ou na SCAM, que apresentou o software pela primeira vez. Os outros 47 projetos possuem maior reconhecimento acadêmico, considerando que foram encontradas menções em um ou mais artigos diferentes da primeira publicação.

### 6.2.2 Q2 - A publicização do software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM influencia o seu reconhecimento?

Há suspeitas de correlação entre o número de menções e o uso de licenças de software livre. Há 201 menções aos 38 projetos software acadêmico de análise estática sem licença definida, enquanto há 228 menções aos 22 projetos com licença, ou seja, mesmo com um

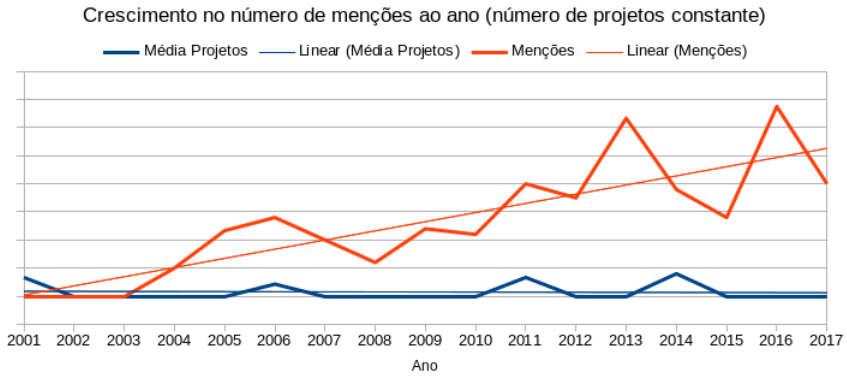



**Figura 6.2** Crescimento no número de menções ao ano (crescimento médio = 38%).

número menor de projetos, o número de menções encontradas foi 14% maior, conforme Figura 6.3.

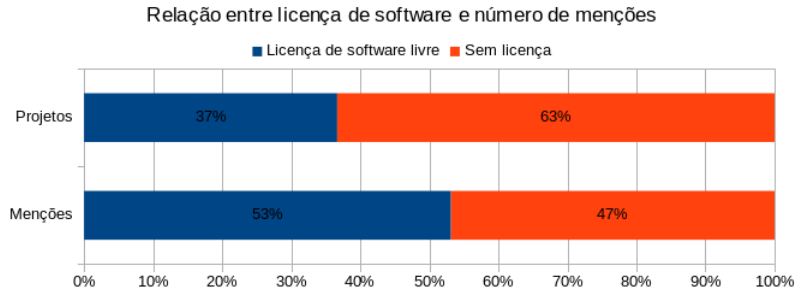



**Figura 6.3** Relação entre o uso de licença de software livre e o número de menções.

Não foi encontrada relação entre o número de menções e outras características do software acadêmico, por exemplo, disponibilidade para download, linguagem de programação, acesso ao site ou número de lançamentos.

### 6.2.3 Q3 - O ciclo de vida do software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM influencia o seu reconhecimento?

Os projetos em estágio de *Desenvolvimento inicial* possuem menos menções em comparação com o tempo de vida do que os projetos em estágio de *Evolução* ou de *Manutenção* que possuem mais lançamentos do que o tempo de vida do projeto.

Foram encontradas 160 menções para projetos em estágio *Encerrado*, possivelmente indicando artigos publicados antes do projeto entrar nesta fase ou indicando que o projeto está acessível apenas para os autores destes artigos mas não para o público em geral, 71 menções para projetos em estágio de *Evolução* ou *Manutenção* e 131 menções para projetos em estágio de *Desenvolvimento inicial*.

Os projetos `s19`, `s38` e `s56` foram encontrados em um grande número de menções,

incluindo publicações recentes entre os anos de 2016 e 2017, no entanto o fato de serem projetos em estágio *Encerrado* chama atenção, uma vez que não estão disponíveis publicamente mas continuam recebendo atenção da academia, este fato não foi investigado, mas estes projetos são objetos de estudo interessantes para compreender este fenômeno.

Não encontramos relação entre o ciclo de vida do software acadêmico e o reconhecimento medido através de menções na literatura acadêmica nas bases ACM e IEEE.

### 6.2.4 Q4 - Qual o tamanho do software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM?

O tamanho médio do software acadêmico de análise estática publicado nas conferências ASE e SCAM é de 820 módulos, neste contexto módulo refere-se às unidades que compõem um sistema de software. Paradigmas e linguagens de programação possuem uma ou mais construções que fazem o papel de módulo – "classes", "aspectos", "tipos abstratos de dados", ou "arquivos-fonte". Esta média considera a última versão disponível em código-fonte de cada projeto.

O tamanho médio dos projetos em estágio de *Desenvolvimento inicial* é de 595 módulos. Estes projetos são muito menores que os projetos em estágio de *Evolução* ou *Manutenção* que possuem 1261 módulos, em média. Os projetos em estágio de *Descontinuado* ou *Encerrado* não possuem código disponível e, portanto, não sabemos seu tamanho. A Figura 6.4 apresenta estes números.

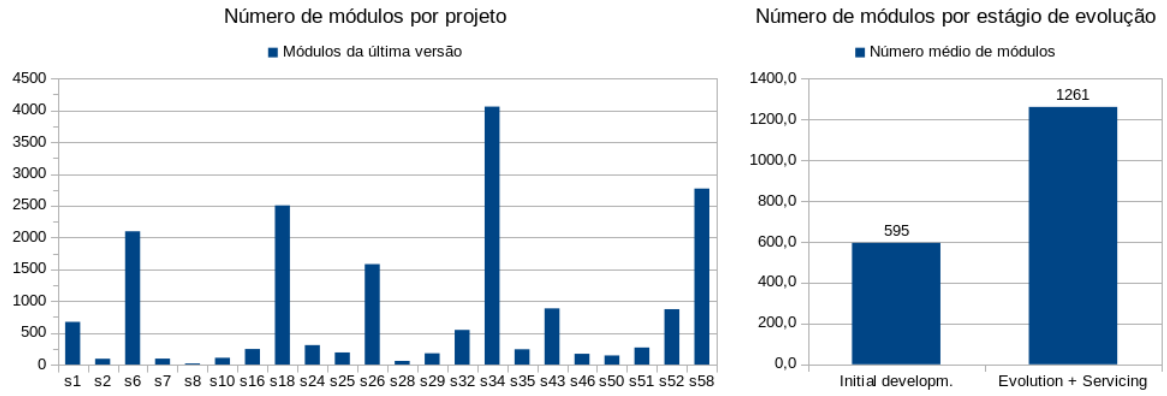



**Figura 6.4** Número de módulos por projeto e por estágio de evolução no ciclo de vida.

De 20 projetos em estágio de *Desenvolvimento inicial*, 12 projetos tiveram o código-fonte analisado. Todos os 8 projetos em estágio de *Evolução* ou *Manutenção* tiveram seu código-fonte analisado. Vale destacar que estes dois estágios foram agregados num único conjunto pois possuem características próximas: os 2 projetos em estágio de *Evolução* estão muito mais próximos dos projetos em estágio de *Manutenção* do que do projetos em *Desenvolvimento inicial*.

### 6.2.5 Q5 - Como evolui o tamanho do software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM?

Ao analisar a evolução no tamanho (em número de módulos) dos projetos em estágio de *Manutenção*, apresentados na Figura 6.5, nota-se um crescimento no número total e médio de módulos em todos os projetos ao longo dos anos.

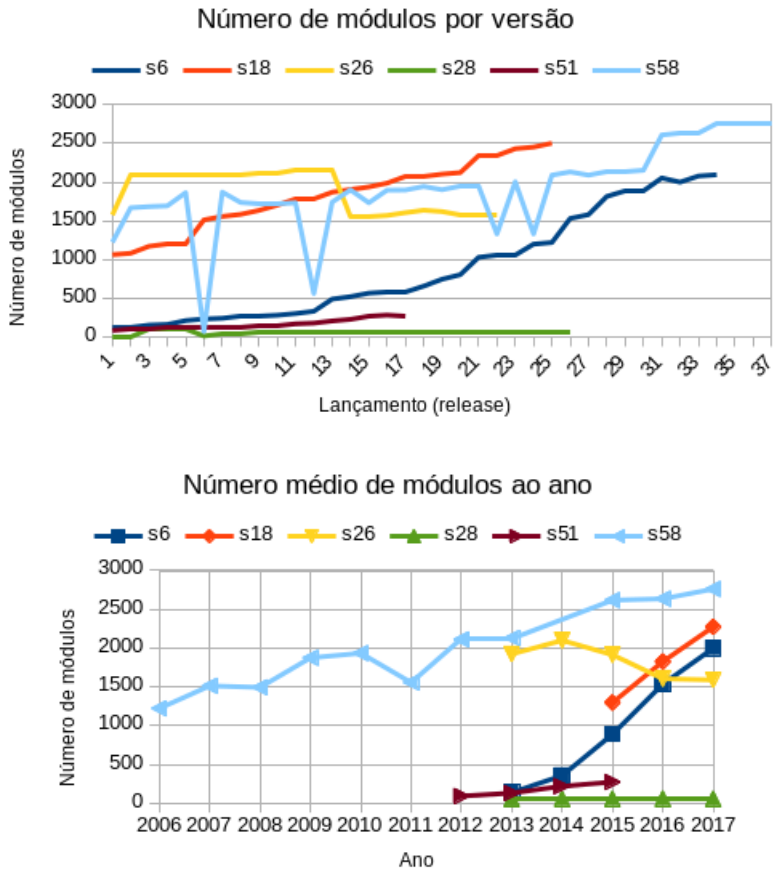



**Figura 6.5** Evolução no número de módulos dos projetos em *Manutenção*. Os picos de queda do projeto `s58` estão associados a lançamentos de versões intermediárias ou de pré-lançamento identificadas como: 1.0M, dilla_0.0.2-RC2, X10_2.1.1 e X10_2.1.2

Ao analisar os projetos individualmente, em busca de relação entre menções e a evolução no tamanho do código-fonte (em número de módulos), observamos que, para alguns projetos, existe tal relação. Especulamos que outros projetos podem ter surgido na academia mas tiveram seu desenvolvimento posterior externo ao meio científico, ao menos para os que foram publicados e indexados nas bases da ACM e do IEEE.

**Software `s6` (CIVL).** Este projeto teve 5 menções, sendo que em 2015 houve menção do tipo `Contribui` e em 2017 apenas menções do tipo `Cita`. Apesar de poucas menções do tipo `Contribui`, o projeto apresenta características de estar evoluindo juntamente a atividade acadêmica.

**Software `s18` (Error Prone).** Este projeto foi mencionado em apenas 1 artigo em 2012, sendo que o primeiro lançamento da versão 2.0 foi em 2015. Este projeto aparenta ser um projeto que ganhou vida própria e seguiu fora da academia.

**Software `s26` (HUSACCT).** Este projeto foi mencionado em 7 publicações distintas; em 2014 houve menção Contribui, nos anos seguintes possui menções de uso e contribuição. A princípio é um projeto com ligação estreita com a atividade acadêmica.

**Software `s28` (JastAdd).** Este projeto está entre os que mais receberam menções, entre 2003 até 2013. O projeto tem muitas menções, incluindo contribuições; após 2013, possui apenas citações e uso sem novas contribuições. Observa-se uma clara ligação entre a evolução do projeto e a atividade acadêmica.

**Software `s51` (SPARTA).** Este projeto teve 4 menções, sendo apenas 1 menção Contribui em 2015, e menção Cita e Usa em 2017. O histórico de lançamentos inicia em 2012 sendo que as primeiras menções aparecem em 2002, o que nos leva a questionar: onde estão os lançamentos anteriores a 2012?

**Software `s58` (WALA).** Este projeto foi mencionado 11 vezes, sendo apenas 1 menção Contribui em 2010. As demais menções são, em grande parte, de uso do software. Entre todos os projetos estudados, este é o que possui maior janela de tempo entre lançamentos, com um histórico de lançamentos de 11 anos. A média dos demais projetos neste estágio de evolução é de 4 anos.

Em resumo, há um crescimento constante no tamanho em número de módulos dos projetos de software acadêmico de análise estática. Entretanto, alguns projeto apresentam características de evolução totalmente independentes da atividade acadêmica, indicando que os seus desenvolvedores, apesar de continuar evoluindo os projetos de software, não tem publicado artigos científicos que os mencionam.

### 6.2.6 Q6 - É possível replicar ou reproduzir pesquisas que mencionam software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM?

Projetos de software acadêmico de análise estática em estágio de *Encerrado* foram encontrados em menções do tipo Usa em 38 artigos e em menções do tipo Contribui em 42 artigos. Estes projetos não estão disponíveis para download, seja binário ou código-fonte, e a URL indicada pelos autores não permanece acessível, indicando que os resultados de tais artigos não podem ser reproduzidos.

Em resumo, as pesquisas reportadas em 80 artigos com menção de Uso ou Contribuição em software acadêmico de análise estática publicados nas conferências ASE e SCAM não podem ser replicadas ou reproduzidas por conta da ausência do software, caso o software seja disponibilizado outros fatores podem levar a dificuldades de replicação, como o acesso aos dados, instruções e detalhes sobre coleta e análise de dados, entre outros fatores.

## 6.3 RECOMENDAÇÕES

Os problemas identificados neste estudo podem ser atribuídos, em grande parte, a baixos orçamentos, limitação de tempo e alta rotatividade entre os grupos de pesquisa. Outros são, possivelmente, ocasionados por questões culturais (NIEMEYER, 2017), como, por exemplo, a tímida adoção de práticas da Ciência Aberta entre pesquisadores.

Ciência Aberta é um movimento que tem por objetivo tornar a pesquisa científica, seus dados e sua disseminação acessíveis a todos os interessados, sejam amadores ou profissionais (WIKIPEDIA, 2015). Sua principal motivação está em possibilitar a reprodução dos resultados de pesquisas e em garantir transparência das metodologias utilizadas. Isto aumenta o impacto social das pesquisas e gera economia de tempo e dinheiro para os pesquisadores e para as instituições (RIN/NESTA, 2010).

Assim, surge um conjunto de ações que podem ser tomadas pelos diferentes atores em direção a garantir sustentabilidade nos projetos de software, ações para praticantes de software, pesquisadores, associações profissionais, educadores, clientes e usuários. No que tange aos resultados deste estudo, as recomendações são bastante simples mas muito efetivas.

Recomendações aos desenvolvedores de software acadêmico (JIMéNEZ et al., 2017):

**Torne o código-fonte do software público o mais cedo possível.** Desenvolva o código-fonte de maneira pública e acessível; utilizar repositórios de controle de versão (exemplos, GitHub, Gitlab ou Savannah) desde o início do projeto. Quanto mais tempo o projeto seguir um modelo fechado, mais difícil se torna abrir. Evite publicar o software em infraestrutura particular ou própria, como por exemplo, servidores da universidade, que tendem a mudar de endereço ou serem descontinuados.

**Faça o software fácil de ser encontrado e forneça metadados.** Forneça metadados a respeito do software através de registros comumente adotados pela sua comunidade. Facilitar a descoberta do projeto de software e seu código-fonte registrando os metadados a respeito do software em locais de registro populares e conhecidos pela comunidade. Metadados devem incluir informações de localização do código-fonte, contribuidores, licença, versão, identificados, referências e como citar o software.

Boas opções são periódicos específicos para software e ferramentas, exemplos: JOSS, JORS e SoftwareX. Fornecer instrução sobre como citar o software adequadamente, se possível incluir no repositório do projeto um arquivo BibTeX com os metadados de como deve ser citado.

**Adote uma licença e respeite as licenças de outros pacotes de software.** Adote uma licença de software livre adequada para deixar claro como usar, modificar e redistribuir o código-fonte em termos e condições claros e bem definidos. Defina a licença de maneira pública e acessível no repositório de código-fonte e garanta que o software está em conformidade com as licenças de todas as dependências do software.

Sugestão, preferir especialmente licenças com mecanismo de copyleft, como por exemplo: GPL.

**Defina processos claros de contribuição, governança e comunicação.** Abrir o código do software não significa que o software será desenvolvido de maneira pública e colaborativa. Apesar de ser algo desejável, recomendações por licenças e práticas comuns de software livre não determinam a estratégia para colaborar com a comunidade de desenvolvimento. Entretanto, projetos devem ser claros sobre como contribuições devem ser feitas e incorporadas tendo modelos de governança transparentes e canais de comunicação claros.

Inevitavelmente alguns projetos de software acadêmico irão continuar sendo úteis após o primeiro lançamento, alguns terão algumas gerações de melhorias, outros serão usados na sua versão original sem atualização ou manutenção e alguns outros serão lançados e nunca utilizados. Isto é natural e a comunidade ao redor do software irá decidir qual é o melhor caminho a se tomar num processo evolutivo (WEINER et al., 2009).

No entanto, sendo o software acadêmico uma parte primordial da produção científica, ele deve estar disponível e continuar disponível para futuras gerações. Dessa forma, realizar estudos desta natureza, refletindo sobre o papel do software e evidenciando o quanto ele é tratado em termos de publicização e reconhecimento é primordial.

No entanto, algumas lições aprendidas na realização deste estudo nos dizem que atividades de revisão de literatura em busca de menções a software representam um alto grau de dificuldade e volume de trabalho "braçal". Este tipo de trabalho seria simplificado enormemente caso fosse dado ao software um maior reconhecimento e incentivos a citação formal destes artefatos digitais, ou ao menos, que haja alguma padronização sobre como citar software. Tal padronização permitiria automatizar e recuperar, de forma segura, a relação entre os projetos, entre artigos e o software, entre pesquisadores e software, assim como é possível hoje para toda a produção de literatura.

## 6.4 TRABALHOS RELACIONADOS

Não encontramos artigos científicos que tratam de sustentabilidade técnica de software acadêmico segundo publicização, reconhecimento e ciclo de vida, conforme proposto e aplicado em nossa pesquisa sobre sustentabilidade técnica de software acadêmico de análise estática.

Entretanto, a preocupação com software acadêmico e sua sustentabilidade extrapola a área de engenharia de software. A pesquisa sobre software acadêmico abrange aspectos direta ou indiretamente relacionados a sua sustentabilidade, incluindo o envolvimento de cientistas de diversas áreas sem conhecimento sobre boas práticas de desenvolvimento até questões relacionadas a reprodutibilidade científica.

Em 2013, Barnes et al. (2013) criaram o manifesto *Science Code Manifesto* que enfatiza que todo código-fonte escrito especificamente para processar dados de publicações deve estar disponível aos revisores e leitores da publicação.

Hettrick et al. (2014) mostraram que, no Reino Unido, em todas as áreas da ciência, 56% dos cientistas estão envolvidos no desenvolvimento de software acadêmico. Em 2015, Amann et al. (2015) investigaram, por meio de uma revisão sistemática de literatura, uma década de publicações e concluíram que poucos estudos são replicáveis: faltam informações incluindo dados e ferramentas, e apenas 20% dos estudos possuem ferramentas

disponíveis. Ainda em 2015, Momcheva e Tollerud (2015) realizaram um survey com 1142 participantes sobre o uso de software em pesquisas da astronomia e mostraram que 90% dos cientistas escreve software e 100% usa software em suas pesquisas.

Smith, Katz e Niemeyer (2016) apresentaram recomendações sobre como citar software na literatura acadêmica com objetivo de encorajar uma ampla adoção e uma política consistente para citação de software entre as múltiplas disciplinas. Smith, Katz e Niemeyer (2016) afirmaram que "citações ao software devem permitir e facilitar acesso ao software, metadados, documentação, dados e outros materiais necessários tanto para humanos quanto para máquinas".

Howison e Bullard (2016), por meio de uma revisão de literatura, mostraram que, em publicações usando software como método, apenas 59 de um total de 90 artigos mencionavam o uso de software de alguma forma; os demais 31 artigos, apesar de usar software acadêmico, não mencionavam nada a respeito.

Wilson et al. (2017) apresentaram um conjunto de boas práticas que todo pesquisador pode adotar, independente do seu nível de habilidade em computação. Essas práticas passam por gerenciamento de dados, programação, colaboração com colegas, organização de projetos, *tracking work*, e escrita da manuscritos.

Na Engenharia de Software, Segal e Morris (2008) investigaram como o desenvolvimento de software acadêmico poderia ser melhorado e enfatizaram as diferenças do software acadêmico em relação aos demais tipos de software, onde o conhecimento sobre o domínio pode muitas vezes incluir temas avançados e pouco comuns fora do meio científico.

Knutson et al. (2010), ao resumir as conclusões do evento RESER (*Workshop on Replication in Empirical Software Engineering Research*) de 2011, pontuou que ferramentas de software acadêmico estão indisponíveis ou não são usáveis, tornando replicação precisa impraticável.

Robles (2010), em revisão de 171 artigos do MSR entre 2004 e 2009, em busca de conjunto de dados, artefatos e ferramentas utilizadas nos estudos necessárias para replicação, mostrou que a maioria dos artigos não conseguiu encontrar as ferramentas mesmo quando o autor explicitamente afirma que uma ferramenta foi desenvolvida.

Kon et al. (2011), numa revisão de literatura da conferência SBES entre os anos 2000 e 2010 mostrou que há um número crescente de pesquisadores Brasileiros disponibilizando suas ferramentas como software livre, no entanto, no período pesquisado, encontraram 14 autores com publicação de software acadêmico que apesar de estarem dispostos a disponibilizar os seus projetos como software livre não obtiveram sucesso nesta tarefa.

Portillo-Rodríguez et al. (2012), por meio de um mapeamento sistemático, mostraram que grande parte das ferramentas de software criadas na academia estão em estado inicial de desenvolvimento, e que apenas uma pequena porcentagem é testada fora do contexto onde foi desenvolvida.

Chaturvedi, Sing e Singh (2013) fizeram uma revisão de literatura de artigos submetidos ao MSR de 2007 até 2013, e identificaram conjuntos de dados, ferramentas e técnicas utilizados pelos autores. A revisão identificou que mais da metade dos artigos do MSR usa ou cria ferramentas, e categorizou as ferramentas em: ferramentas novas, ferramentas tradicionais, protótipos e scripts para mineração de dados.

Marshall e Brereton (2013) realizaram um mapeamento sistemático sobre artigos que apresentam ferramentas de apoio a revisão sistemática no domínio da Engenharia de Software e concluiu que as ferramentas encontradas estão em estado inicial de desenvolvimento.

Wilson et al. (2014) resumiram as melhores práticas para melhoria da manutenibilidade e disponibilidade do software acadêmico desenvolvido por cientistas. Wilson (2014) reportou as lições aprendidas em 20 anos da iniciativa *Software Carpentry* para atividades de melhoria das habilidades dos pesquisadores com computação.

Em artigo recente no contexto de adoção de ferramentas de análise estática, Beller et al. (2016) avaliaram e sugeriram caminhos para melhorar o desenvolvimento de ferramentas de análise estática com o objetivo de aumentar sua adoção.

# Capítulo 7

# CONCLUSÕES

O desenvolvimento de software acadêmico, de forma sustentável, abre portas para elevar a qualidade geral do software em questão e da pesquisa científica, promovendo a reprodutibilidade e proporcionando um ambiente de compartilhamento e colaboração em oposição ao tradicional modelo de competição que permeia o sistema de reputação e crédito científico. Na área de engenharia de software, especialmente no domínio de análise estática, com tradição no desenvolvimento de ferramentas para apoiar pesquisas em diferentes áreas da ciência da computação, a preocupação com a sustentabilidade técnica em software acadêmico não pode ser desconsiderada.

Esta pesquisa de mestrado caracterizou projetos de software acadêmico de análise estática, publicados até 2015 em artigos científicos das conferências ASE e SCAM, em relação a sua sustentabilidade técnica, definida em termos de publicização, reconhecimento e ciclo de vida. O estudo sobre a publicização identificou 60 projetos de software acadêmico de análise estática publicados originalmente nas conferências ASE e SCAM. A caracterização da publicização desses projetos considerou disponibilidade para download, acesso ao código-fonte, forma de distribuição e licença. Apenas 3% dos 1873 artigos publicados nas conferências ASE e SCAM publicizaram software acadêmico de análise estática de forma adequada, com indicação de URL para download. O estudo sobre reconhecimento inspecionou 806 artigos encontrados nas bases ACM e IEEE através de busca avançada, usando características de 60 projetos de software acadêmico de análise estática. A inspeção identificou 429 menções dos tipos `Cita`, `Usa` ou `Contribui`. Houve um crescimento de 38% ao ano no número de menções ao software acadêmico, e apenas 10% do total de menções realizam contribuição de código-fonte dos projetos. O estudo sobre ciclo de vida caracterizou o estágio de evolução de software acadêmico de análise estática com código-fonte disponível, considerando o número de lançamentos e de módulos no código-fonte, e revelou que a maior parte dos projetos encontra-se em estágio inicial de desenvolvimento ou encerrado.

Ao sintetizar os resultados (Capítulo 6) para responder a questão geral de pesquisa ( *Como a desordem caótica disfuncional (DCD) pode explicar a sustentabilidade técnica*

*dos projetos de software acadêmico de análise estática em termos de publicização, reconhecimento e estágio de evolução?* ) percebemos que a DCD é útil para explicar a sustentabilidade técnica de um domínio de aplicação em níveis distintos de profundidade, através da seguintes características:

**C1** Existência de muitos projetos com poucos usuários;

**C2** Projetos com ciclos de vida curtos que se encerram junto ao financiamento inicial;

**C3** Comunidades de usuários desconectadas e paralelas;

**C4** Incompatibilidades entre os projetos de maneira persistente e imutável;

**C5** Tentativas constantes e aparentemente não coordenadas de "reiniciar" tudo (*reboots*).

Em nosso estudo utilizamos as características **C1** e **C2** da DCD para explicar a sustentabilidade técnica dos projetos do ecossistema de análise estática e percebemos que neste domínio há muitos projetos de software acadêmico indisponíveis ou encerrados (78%), com pouco reconhecimento e com poucos usuários, com ciclos de vida curtos ou em estágio inicial de desenvolvimento, revelando um ecossistema em que há pouca colaboração e indícios de graves problemas de sustentabilidade.

As demais características de DCD – **C3**, **C4** e **C5** – apesar de estarem fora do escopo deste estudo, se mostram fundamentais para uma avaliação mais profunda e abrangente da sustentabilidade técnica de um domínio de aplicação, complementando as duas outras características utilizadas neste estudo – **C1** e **C2** – especialmente em termos de colaboração e coordenação entre os projetos e suas comunidades.

## 7.1 CONTRIBUIÇÕES

A caracterização da sustentabilidade técnica de software acadêmico de análise estática publicado nas conferências de Engenharia de Software ASE e SCAM mapeou os projetos de software disponíveis e o grau de evolução que se encontram no ciclo de vida.

Este mapeamento abre caminho para a compreensão de problemas relacionados a sustentabilidade de software acadêmico de análise estática e posterior definição de estratégias para solucionar e melhorar o campo, tanto em termos práticos quanto teóricos.

O conhecimento a respeito dos projetos de software acadêmico de análise estática existentes serve aos interessados em utilizar tais projetos em novas pesquisas, seja como objeto de estudo, como apoio metodológico, ou ainda, como base para iniciar novos desenvolvimentos.

Esta pesquisa contribui ainda com uma reflexão sobre o campo de análise estática e seus projetos de software acadêmico, especialmente em relação ao esforço e recursos investidos no desenvolvimento de software neste domínio de aplicação sendo consumidos de maneira ineficiente. Consideramos que cada software acadêmico é o resultado de investimentos, seja de pesquisadores individuais, seja de grupos e pesquisas, com financiamento ou com o "tempo livre" dos pesquisadores, uma vez que estes projetos de software

tornam-se indisponíveis e inacessíveis, abrem espaço para duplicação de esforço através de re-trabalho para re-implementar tais projetos.

No campo teórico, contribui para uma melhor compreensão do que vem a ser sustentabilidade de software, um tema que ainda carece de definição clara, especialmente sustentabilidade de software acadêmico de análise estática.

Por fim, contribui alertando para os prejuízos que a indisponibilidade de código destes projetos causam para a Ciência, uma vez que acesso ao código é parte fundamental para validar ou refutar conclusões de pesquisas através da reprodução e replicação, apesar deste aspecto não ter sido explicitamente avaliado, inúmeros estudos apontam para a importância de estarem disponíveis quaquer artefato produzido em pesquisas, isto inclui o código-fonte dos projetos de software acadêmico.

## 7.2 LIMITAÇÕES

O escopo do estudo limitou-se a selecionar software acadêmico em apenas duas conferências de Engenharia de Software, ASE e SCAM, e ainda, usando como data limite o ano de 2015. A busca por menções aos projetos também foi limitado a apenas duas bibliotecas de indexação de publicações, ACM e IEEE.

Além da limitação de escopo, este estudo realizou também uma série de procedimentos manuais: a revisão de literatura para seleção de software acadêmico foi realizada manualmente, a execução da busca nas bases ACM e IEEE foi feita em atividades manuais.

## 7.3 TRABALHOS FUTUROS

O trabalho mais imediato para estender esta pesquisa seria atualizar a revisão de literatura inicial para estender o período de seleção de software acadêmico de análise estática incluindo os anos de 2016 e 2017, uma vez que esta dissertação limitou a seleção até o ano de 2015.

Outro trabalho importante é ampliar o escopo do estudo incluindo mais conferências visando aumentar o realismo sobre o domínio estudado, especialmente conferências tradicionais de Engenharia de Software, como por exemplo, a conferência ICSE[1] (*International Conference on Software Engineering*) e ICSME[2] (*International Conference on Software Maintenance and Evolution*), por serem duas das mais tradicionais e importantes conferências da área de Engenharia de Software. Consideramos também ser importante incluir o SBES (Simpósio Brasileiro de Engenharia de Software) por ser uma importante conferência no contexto Brasileiro de Engenharia de Software, com longo histórico, e trilhas de publicação de ferramentas.

Outro ponto em relação a conferências é coletar o fator de impacto da conferência e utilizar esta informação na análise dos dados e discussão. É possível que conferências com maior fator de impacto apresentem projetos com maior reconhecimento e com características de publicização mais adequadas.

Além de ampliar o escopo de forma horizontal incluindo novas conferências, é inte-

[1] ⟨http://www.icse-conferences.org⟩
[2] ⟨http://conferences.computer.org/icsm/⟩

ressante também aumentar a qualidade da caracterização do software incluindo novas dimensões de caracterização, especialmente dimensões na visão de usuário, como por exemplo, documentação, facilidade de instalação, execução, existência de manuais, avaliar o nível de descrição e apresentação do software no site ou repositório, ou mesmo incluir aspectos de Engenharia de Software, como por exemplo, testes, resolução de *issues*, métricas de qualidade como por exemplo complexidade estrutural, número de contribuidores, número de commits e uso de controle de versão.

Uma outra forma de enriquecer a caracterização de cada software é usar caracterizações já realizadas em outros trabalhos científicos. Muitos estudos avaliam e comparam ferramentas de análise estática entre si; os artigos encontrados na seleção de software acadêmico são os primeiros candidatos a serem utilizados como fonte de coleta de dados sobre cada software.

Outra possibilidade de trabalho futuro é revisar artigos publicados em periódicos com foco específico em publicação de software em busca mais projetos de software acadêmico de análise estática. Entre estes periódicos podemos citar, por exemplo, o JORS[3] (*Journal of Open Research Software*), JOSS[4] (*Journal of Open Source Software*) (SMITH et al., 2017) e o SoftwareX[5] da Elsevier.

[3] ⟨https://openresearchsoftware.metajnl.com/articles/10.5334/jors.bt⟩
[4] ⟨http://joss.theoj.org⟩
[5] ⟨https://www.journals.elsevier.com/softwarex⟩

# REFERÊNCIAS BIBLIOGRÁFICAS


ALEMERIEN, K.; MAGEL, K. Experimental evaluation of static source code analysis tools. In: *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*. [S.l.: s.n.], 2013. p. 1.

ALLEN, A. et al. Engineering academic software (dagstuhl perspectives workshop 16252). In: *Dagstuhl Manifestos*. [S.l.]: Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017. v. 6, n. 1.

ALMQVIST, J. P. F. Replication of Controlled Experiments in Empirical Software Engineering - A Survey. 2006. Disponível em: ⟨http://lup.lub.lu.se/student-papers/record/1330459⟩.

AMANN, S. et al. Software mining studies: Goals, approaches, artifacts, and replicability. In: *Software Engineering*. [S.l.]: Springer, 2015. p. 121–158.

ANDERSON, P. The use and limitations of static-analysis tools to improve software quality. *CrossTalk: The Journal of Defense Software Engineering*, Citeseer, v. 21, n. 6, p. 18–21, 2008.

BAKER, M. Why scientists must share their research code. *Nature*, 2016. Disponível em: ⟨http://dx.doi.org/10.1038/nature.2016.20504⟩.

BARNES, N. et al. *Science code manifesto*. 2013. Online; acessado em 30 de novembro de 2017. Disponível em: ⟨http://sciencecodemanifesto.org/⟩.

BARR, E. et al. On the Shoulders of Giants. In: *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*. New York, NY, USA: ACM, 2010. (FoSER '10), p. 23–28. ISBN 978-1-4503-0427-6. Disponível em: ⟨http://doi.acm.org/10.1145/1882362.1882368⟩.

BASILI, V. R.; CALDIERA, G.; ROMBACH, H. D. The goal question metric approach. *Encyclopedia of Software Engineering*, Wiley, v. 2, p. 528–532, 1994.

BECKER, C. et al. The karlskrona manifesto for sustainability design. *CoRR*, abs/1410.6968, 2014. Disponível em: ⟨http://arxiv.org/abs/1410.6968⟩.

BELLER, M. et al. Analyzing the state of static analysis: A large-scale evaluation in open source software. In: *Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on*. [S.l.: s.n.], 2016. v. 1, p. 470–481.

BINKLEY, D. Source code analysis: A road map. In: IEEE COMPUTER SOCIETY. *2007 Future of Software Engineering*. [S.l.], 2007. p. 104–119.

CAPILUPPI, A. et al. Adapting the staged model for software evolution to free/libre/open source software. In: *Ninth international workshop on Principles of software evolution: in conjunction with the 6th ESEC/FSE joint meeting*. [S.l.: s.n.], 2007. p. 79–82.

CHATURVEDI, K. K.; SING, V. B.; SINGH, P. Tools in mining software repositories. In: *Computational Science and Its Applications (ICCSA), 2013 13th International Conference on*. [S.l.: s.n.], 2013. p. 89–98.

CHESS, B.; WEST, J. *Secure programming with static analysis*. [S.l.]: Pearson Education, 2007.

COUNCIL, C. Cyberinfrastructure vision for 21st century discovery. *National Science Foundation, USA*, v. 64, p. 1–66, 2007.

CRUZ, D. d.; HENRIQUES, P. R.; PINTO, J. S. Code analysis: Past and present. 2009.

DAGPINAR, M.; JAHNKE, J. H. Predicting maintainability with object-oriented metrics - an empirical comparison. *10th WCRE Proceedings*, 2003.

DHUNGANA, D. et al. Software ecosystems vs. natural ecosystems: learning from the ingenious mind of nature. In: *Proceedings of the Fourth European Conference on Software Architecture: Companion Volume*. [S.l.: s.n.], 2010. p. 96–102.

DRUMMOND, D. C. Replicability is not Reproducibility: Nor is it Good Science. In: . [s.n.], 2009. Disponível em: ⟨http://cogprints.org/7691/⟩.

FEITELSON, D. G. From Repeatability to Reproducibility and Corroboration. *SIGOPS Oper. Syst. Rev.*, v. 49, n. 1, p. 3–11, jan. 2015. ISSN 0163-5980. Disponível em: ⟨http://doi.acm.org/10.1145/2723872.2723875⟩.

FENTON, N.; BIEMAN, J. *Software metrics: a rigorous and practical approach*. [S.l.]: CRC Press, 2014.

GOBLE, C. Better software, better research. *IEEE Internet Computing*, IEEE, v. 18, n. 5, p. 4–8, 2014.

GONZáLEZ-BARAHONA, J. M.; ROBLES, G. On the reproducibility of empirical software engineering studies based on data retrieved from development repositories. *Empirical Software Engineering*, v. 17, n. 1-2, p. 75–89, fev. 2012. ISSN 1382-3256, 1573-7616. Disponível em: ⟨https://link.springer.com/article/10.1007/s10664-011-9181-9⟩.

GOSAIN, A.; SHARMA, G. Static analysis: A survey of techniques and tools. In: *Intelligent Computing and Applications*. [S.l.]: Springer, 2015. p. 581–591.

GOUVEIA, F. C. et al. Altmetria: métricas de produção científica para além das citações. *Liinc em Revista*, v. 9, n. 1, p. 214–227, 2013.

GRAND, A. et al. conference, *On open science and public engagement with engineering*. Trento, Italy: [s.n.], 2010. Disponível em: ⟨http://events.unitn.it/en/easst010⟩.

HARMAN, M. Why source code analysis and manipulation will always be important. In: *Source Code Analysis and Manipulation (SCAM), 2010 10th IEEE Working Conference on*. [S.l.: s.n.], 2010. p. 7–19.

HASHIM, K.; KEY, E. A software maintainability attributes model. *Malaysian Journal of Computer Science*, Faculty of Computer Science & Information Technology, v. 9, n. 2, p. 92–97, 1996.

HETTRICK, S. et al. *UK Research Software Survey 2014*. 2014. Disponível em: ⟨https://doi.org/10.5281/zenodo.14809⟩.

HINSEN, K. ActivePapers: a platform for publishing and archiving computer-aided research. *F1000Research*, v. 3, jul. 2015. ISSN 2046-1402. Disponível em: ⟨https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4448745/⟩.

HOWISON, J.; BULLARD, J. Software in the scientific literature: Problems with seeing, finding, and using software mentioned in the biology literature. *Journal of the Association for Information Science and Technology*, v. 67, n. 9, p. 2137–2155, 2016. ISSN 2330-1643. Disponível em: ⟨http://dx.doi.org/10.1002/asi.23538⟩.

HOWISON, J. et al. Understanding the scientific software ecosystem and its impact: Current and future measures. *Research Evaluation*, v. 24, n. 4, p. 454–470, 2015. Disponível em: ⟨+http://dx.doi.org/10.1093/reseval/rvv014⟩.

HOWISON, J.; HERBSLEB, J. D. Scientific software production: incentives and collaboration. In: *Proceedings of the ACM 2011 conference on Computer supported cooperative work*. [S.l.: s.n.], 2011. p. 513–522.

HOWISON, J.; HERBSLEB, J. D. Incentives and integration in scientific software production. In: _____. [S.l.: s.n.], 2013. p. 459–470. ISBN 9781450313315.

ILYAS, B.; ELKHALIFA, I. *Static Code Analysis: A Systematic Literature Review and an Industrial Survey*. 2016.

JIMéNEZ, R. C. et al. Four simple recommendations to encourage best practices in research software. *F1000Research*, v. 6, p. 876, jun. 2017. ISSN 2046-1402. Disponível em: ⟨https://f1000research.com/articles/6-876/v1⟩.

JOHNSON, S. C. Lint, a c program checker. In: *COMP. SCI. TECH. REP.* [S.l.: s.n.], 1978. p. 78–1273.

JURISTO, N.; GóMEZ, O. S. Replication of Software Engineering Experiments. In: *Empirical Software Engineering and Verification*. Springer, Berlin, Heidelberg, 2012, (Lecture Notes in Computer Science). p. 60–88. ISBN 978-3-642-25230-3 978-3-642-25231-0. DOI: 10.1007/978-3-642-25231-0_2. Disponível em: ⟨https://link.springer.com/chapter/10.1007/978-3-642-25231-0_2⟩.

JURISTO, N.; VEGAS, S. Using differences among replications of software engineering experiments to gain knowledge. In: *2009 3rd International Symposium on Empirical Software Engineering and Measurement*. [S.l.: s.n.], 2009. p. 356–366.

KATZ, D. Transitive credit as a means to address social and technological concerns stemming from citation and attribution of digital products. *Journal of Open Research Software*, Ubiquity Press, v. 2, n. 1, 2014.

KAZMAN, A. T. R. On the worthiness of software engineering research. 2016. Disponível em: ⟨http://shidler.hawaii.edu/sites/shidler.hawaii.edu/files/users/kazman/se_research_worthiness.pdf⟩.

KING, G. Replication, replication. *PS: Political Science & Politics*, Cambridge University Press, v. 28, n. 3, p. 444–452, 1995.

KIRKOV, R.; AGRE, G. Source code analysis - an overview. *Cybernetics and Information Technologies*, v. 10, n. 2, p. 60–77, 2010.

KITCHENHAM, B. A.; BUDGEN, D.; BRERETON, P. *Evidence-Based Software Engineering and Systematic Reviews*. [S.l.]: CRC Press, 2015. Google-Books-ID: bGfdCgAAQBAJ. ISBN 978-1-4822-2866-3.

KNUTSON, C. D. et al. Report from the 1st international workshop on replication in empirical software engineering research (reser 2010). *ACM SIGSOFT Software Engineering Notes*, ACM, v. 35, n. 5, p. 42–44, 2010.

KON, F. et al. Free and open source software development and research: Opportunities for software engineering. In: *SBES*. [s.n.], 2011. p. 82–91. Disponível em: ⟨http://dblp.org/db/conf/sbes/sbes2011.html\\#KonMLTCM11⟩.

KRISHNAMURTHI, S.; VITEK, J. The Real Software Crisis: Repeatability As a Core Value. *Commun. ACM*, v. 58, n. 3, p. 34–36, fev. 2015. ISSN 0001-0782. Disponível em: ⟨http://doi.acm.org/10.1145/2658987⟩.

KUMAR, B. A survey of key factors affecting software maintainability. In: *Computing Sciences (ICCS), 2012 International Conference on*. [S.l.: s.n.], 2012. p. 261–266.

LI, P.; CUI, B. A comparative study on software vulnerability static analysis techniques and tools. In: *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference on*. [S.l.: s.n.], 2010. p. 521–524.

MANIKAS, K.; HANSEN, K. M. Software ecosystems–a systematic literature review. *Journal of Systems and Software*, Elsevier, v. 86, n. 5, p. 1294–1306, 2013.

MARSHALL, C.; BRERETON, P. Tools to support systematic literature reviews in software engineering: A mapping study. In: *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*. [S.l.: s.n.], 2013. p. 296–299.

MCCORMICK, M. M. et al. ITK: enabling reproducible research and open science. *Frontiers in Neuroinformatics*, v. 8, 2014. ISSN 1662-5196. Disponível em: ⟨https://www.frontiersin.org/articles/10.3389/fninf.2014.00013/full⟩.

MENDOZA, D.; GARCIA, C. A. Defining Research Reproducibility: What Do You Mean? *Clinical Chemistry*, v. 63, n. 11, p. 1777–1777, nov. 2017. ISSN 0009-9147, 1530-8561. Disponível em: ⟨http://clinchem.aaccjnls.org/content/63/11/1777⟩.

MERALI, Z. Computational science: Error, why scientific programming does not compute. *Nature*, Nature Publishing Group, v. 467, n. 7317, p. 775–777, oct 2010. ISSN 0028-0836. Disponível em: ⟨http://dx.doi.org/10.1038/467775a⟩.

MOMCHEVA, I.; TOLLERUD, E. Software use in astronomy: an informal survey. *arXiv preprint arXiv:1507.03989*, 2015.

NIEMEYER, K. Open science and the future university researcher. engrXiv, 2017.

PENG, R. D. Reproducible research in computational science. *Science (New York, Ny)*, NIH Public Access, v. 334, n. 6060, p. 1226, 2011.

PLESSER, H. E. Reproducibility vs. Replicability: A Brief History of a Confused Terminology. *Frontiers in Neuroinformatics*, v. 11, 2018. ISSN 1662-5196. Disponível em: ⟨https://www.frontiersin.org/articles/10.3389/fninf.2017.00076/full⟩.

PONTIKA, N. et al. Fostering Open Science to Research Using a Taxonomy and an eLearning Portal. In: *Proceedings of the 15th International Conference on Knowledge Technologies and Data-driven Business*. New York, NY, USA: ACM, 2015. (i-KNOW '15), p. 11:1–11:8. ISBN 978-1-4503-3721-2. Disponível em: ⟨http://doi.acm.org/10.1145/2809563.2809571⟩.

PORTILLO-RODRíGUEZ, J. et al. Tools used in global software engineering: A systematic mapping review. p. 663–685, 2012. Disponível em: ⟨http://dblp.org/db/journals/infsof/infsof54.html\\#Portillo-RodriguezVPB12⟩.

RAJLICH, V. T.; BENNETT, K. H. A staged model for the software life cycle. *Computer*, IEEE, v. 33, n. 7, p. 66–71, 2000.

REUTERS, T. *History of Citation Indexing*. 2017. Página web. [Online; acessado 07 Outubro de 2017]. Disponível em: ⟨https://clarivate.com/essays/history-citation-indexing/⟩.

RIN/NESTA. *Open to All? Case studies of openness in research*. [S.l.], 2010. Disponível em: ⟨http://www.rin.ac.uk/our-work/data-management-and-curation/open-science-case-studies⟩.

ROBLES, G. Replicating msr: A study of the potential replicability of papers published in the mining software repositories proceedings. In: *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*. [S.l.: s.n.], 2010. p. 171–180.

RYDER, B. G. The pfort verifier. *Software: Practice and Experience*, Wiley Online Library, v. 4, n. 4, p. 359–377, 1974.

SEGAL, J.; MORRIS, C. Developing scientific software. *IEEE Software*, IEEE Computer Society, Los Alamitos, CA, USA, v. 25, p. 18–20, 2008. ISSN 0740-7459.

SILVA, F. Q. B. d. et al. Replication of Empirical Studies in Software Engineering: Preliminary Findings from a Systematic Mapping Study. In: *2011 Second International Workshop on Replication in Empirical Software Engineering Research*. [S.l.: s.n.], 2011. p. 61–70.

SMITH, A. M.; KATZ, D. S.; NIEMEYER, K. E. Software citation principles. *PeerJ Computer Science*, v. 2, p. –86, sep 2016. ISSN 2376-5992. Disponível em: ⟨https://doi.org/10.7717/peerj-cs.86⟩.

SMITH, A. M. et al. Journal of open source software (joss): design and first-year review. *ArXiv e-prints*, jul 2017.

STEVENS, S. S. On the Theory of Scales of Measurement. *Science*, v. 103, n. 2684, p. 677–680, jun. 1946. ISSN 0036-8075, 1095-9203. Disponível em: ⟨http://science.sciencemag.org/content/103/2684/677⟩.

STEWART, C. A.; ALMES, G. T.; WHEELER, B. C. Cyberinfrastructure software sustainability and reusability: Report from an nsf-funded workshop. Indiana University, 2010.

STODDEN, V. Enabling reproducible research: Open licensing for scientific innovation. *International Journal of Communications Law and Policy, Forthcoming*, v. 13, p. 1–25, 2009.

STODDEN, V. C. Reproducible research - addressing the need for data and code sharing in computational science. *Computing in Science & Engineering*, v. 12, n. 5, p. 8 – 13, 2010.

STOL, K.-J.; FITZGERALD, B. A holistic overview of software engineering research strategies. In: *3rd International Workshop on Conducting Empirical Studies in Industry*. [S.l.: s.n.], 2015. p. 8.

TERCEIRO, A. et al. Analizo: an extensible multi-language source code analysis and visualization toolkit. p. 6, 2010.

TOMAN, J.; GROSSMAN, D. Taming the static analysis beast. In: *LIPIcs-Leibniz International Proceedings in Informatics*. [S.l.: s.n.], 2017. v. 71.

VENTERS, C. C. et al. Software sustainability: The modern tower of babel. In: *CEUR Workshop Proceedings*. [S.l.: s.n.], 2014. v. 1216, p. 7–12.

VITEK, J.; KALIBERA, T. Repeatability, Reproducibility, and Rigor in Systems Research. In: *Proceedings of the Ninth ACM International Conference on Embedded Software*. New York, NY, USA: ACM, 2011. (EMSOFT '11), p. 33–38. ISBN 978-1-4503-0714-7. Disponível em: ⟨http://doi.acm.org/10.1145/2038642.2038650⟩.

WEINER, B. et al. Astronomical software wants to be free: A manifesto. In: *astro2010: The Astronomy and Astrophysics Decadal Survey*. [S.l.: s.n.], 2009. (Astronomy, v. 2010).

WICHMANN, B. et al. Industrial perspective on static analysis. *Software Engineering Journal*, Citeseer, v. 10, n. 2, p. 69, 1995.

WIKIPEDIA. *Open Science*. 2015. Open Science. [Online; acessado 13 Outubro de 2015]. Disponível em: ⟨http://en.wikipedia.org/wiki/Open_science⟩.

WILSON, G. Software carpentry: lessons learned. *F1000Research*, Faculty of 1000 Ltd, v. 3, 2014.

WILSON, G. et al. Best practices for scientific computing. *PLOS Biology*, Public Library of Science, v. 12, n. 1, p. 1–7, 01 2014. Disponível em: ⟨https://doi.org/10.1371/journal.pbio.1001745⟩.

WILSON, G. et al. Good enough practices in scientific computing. *PLOS Computational Biology*, Public Library of Science, v. 13, n. 6, p. 1–20, 06 2017. Disponível em: ⟨https://doi.org/10.1371/journal.pcbi.1005510⟩.

WREN, J. D. et al. Use it or lose it: citations predict the continued online availability of published bioinformatics resources. *Nucleic acids research*, Oxford University Press, v. 45, n. 7, p. 3627–3633, 2017.

ZHANG, F. et al. How does context affect the distribution of software maintainability metrics? In: *ICSM*. [S.l.: s.n.], 2013. p. 350–359.

Apêndice

A

# REPRODUTIBILIDADE DO ESTUDO

Este trabalho de pesquisa, incluindo textos, imagens, dados e códigos estão disponíveis publicamente sob a licença CC BY-SA 4.0[1] (Licença Atribuição-CompartilhaIgual 4.0 Internacional Creative Commons) no seguinte endereço:

- ⟨https://github.com/joenio/dissertacao-ufba-2016⟩

A maior parte das atividades de comunicação e reuniões de orientação realizadas durante a pesquisa estão documentadas em *issues* no repositório acima ou na wiki do grupo de pesquisa aSide em:

- ⟨http://wiki.dcc.ufba.br/Aside/Orientacao2014JoenioCosta⟩

O texto foi escrito em LaTeX, códigos, scripts e templates foram desenvolvidos em Perl, os dados foram coletados em arquivos de texto plano nos formatos CSV, YAML e BibTeX. Alguns elementos do repositório serão detalhados a seguir, a Tabela A.1 apresenta uma visão geral.

## A.1 ORGANIZAÇÃO DO REPOSITÓRIO

### A.1.1 `bin/`

`bin/cache` Script utilizado para agregar todos os dados de todos os projetos num arquivo único em formato YML em `cache/dataset.yml`.

`bin/ids` Script utilizado para criar o campo `id` com valor autoincremento para cada referência no arquivo `documents/references.bib`.

`bin/merge` Combina os arquivos `acm.bib`, `ieee.bib` e `paper.bib` de cada projeto em um único arquivo no formato BibTeX removendo duplicidades dos resultados.

[1] ⟨http://creativecommons.org/licenses/by-sa/4.0⟩

**Tabela A.1** Organização de arquivos e pastas do repositório.

| Arquivo ou pasta | Descrição |
|---|---|
| `bibliografia.bib` | Arquivo BibTeX com as referências bibliográficas utilizadas no texto. |
| `bin/` | Pasta contendo os scripts desenvolvidos para coleta, transformação e análise de dados. |
| `capitulos/` | Arquivos LaTeX com o conteúdo/texto de cada capítulo da dissertação. |
| `dataset/` | Dados coletados durante os estudos apresentados nos Capítulos 3, 4 e 5. |
| `dissertacao.tex` | Arquivo LaTeX principal com inclusão dos capítulos, título e resumo do texto. |
| `documents/` | Documentos gerados pelos templates com apresentação dos dados coletados. |
| `imagens/` | Imagens, gráficos e demais elementos visuais utilizados no texto. |
| `lib/` | Diretório contendo implementação de métodos para leitura análise e transformação dos dados utilizado nos scripts. |
| `Makefile` | Conjunto de regras para execução dos scripts e compilação do código-fonte LaTeX em PDF. |
| `README.md` | Documento com descrição do repositório, informaçoes de contato, e instruções para compilação dos fontes LaTeX e execução dos scripts. |
| `templates/` | Templates (modelos) para leitura dos dados coletados e transformação em documentos. |
| `ufbathesis.cls` | Arquivo de estilo do LaTeX com definições e padrão de formatação deacordo as regras de publicação da UFBA. |

`bin/render` Este script lê o arquivo `cache/dataset.yml`, carrega os dados em memória e passa estes dados como parâmetro para os arquivos templates localizados em `templates/`.

`bin/run-analizo` Script utilizado para automatizar a execução da ferramenta *analizo metrics* para coletar o número de módulos de cada lançamento de cada software acadêmico.

### A.1.2 `dataset/`

`dataset/literature-review.ods` Planilha LibreOffice Calc[2] com os dados coletados no estudo do Capítulo 3 resultando na seleção de 60 projetos de software acadêmico.

Nesta planilha está documentada cada etapa da revisão estruturada, indicando em

[2] ⟨https://www.libreoffice.org⟩

cada artigo analisado qual o estado do mesmo, se foi ou não incluído na execução da atividade. Nesta planilha é possível encontrar também o nome de cada ferramenta e uma caracterização completa.

`dataset/software/` Estrutura de diretório utilizado para coleta de dados em arquivos do tipo YAML e BibTeX, cada projeto de software selecionado possui um diretório nesta estrutura.

`dataset/software/<nome>/software.yml` Arquivo contendo os dados básicos de cada software, como: nome, descrição, url, licença, disponibilidade para download, entre outros dados.

`dataset/software/<nome>/references.yml` Arquivo YAML para armazenar dados coletados sobre as menções a cada software acadêmico.

`dataset/software/<nome>/releases.yml` Arquivo para armazenar os dados de cada lançamento do software cotendo o número da versão, a data do lançamento e a url para download do software ne versão específica.

`dataset/software/<nome>/paper.bib` Arquivo BibTeX contendo os metadados do artigo inicial onde o software foi selecionado.

`dataset/software/<nome>/search/acm.bib` Metadados dos artigos obtidos como resultado da busca na base ACM.

`dataset/software/<nome>/search/ieee.bib` Metadados dos artigos obtidos como resultado da busca na base IEEE.

### A.1.3 `lib/Dissertacao.pm`

A maior parte da lógica dos scripts foi implementada no arquivo `lib/Dissertacao.pm` com objetivo de reduzir repetição de código e proporcionar reuso.

### A.1.4 `templates/`

O resultado de cada arquivo de template é armazenado num arquivo de mesmo nome no diretório `documents/` sem a extensão `.epl`.

`templates/references.csv.epl` Cria o arquivo `documents/references.csv` com todos os artigos e projetos, indicando quando há relação entre eles através de menção marcada com um `x` sempre que houver menção de um artigo para um software.

`templates/search-strings.csv.epl` Gera um documento CSV com todas as strings de busca no arquivo `documents/search-strings.csv`.

`templates/dataset.csv.epl` Este template gera o arquivo `documents/dataset.csv` com um resumo de dados coletados durante todos os estudos da pesquisa, incluindo número de lançamentos, número de menções, número de menções por tipo, métrica de código-fonte da última versão, entre outros dados para cada software estudado.

`templates/metrics.csv.epl` Este arquivo agrega as métricas de código-fonte coletadas para cada lançamento de cada software numa tabela em formato CSV.

### A.1.5 `ufbathesis.cls`

Disponível em:

- ⟨https://github.com/PGCOMP-UFBA/pgcomp-ufba-latex⟩

## A.2 DETALHES DE IMPLEMENTAÇÃO

Todos os scripts e templates foram desenvolvidos utilizando a linguagem de programação Perl[3] com o auxílio dos seguintes módulos CPAN[4]:

- Devel::CheckBin ⟨http://metacpan.org/pod/Devel::CheckBin⟩
- List::Util ⟨http://metacpan.org/pod/List::Util⟩
- Modern::Perl ⟨http://metacpan.org/pod/Modern::Perl⟩
- Mojo::Template ⟨http://metacpan.org/pod/Mojo::Template⟩
- Text::BibTeX ⟨http://metacpan.org/pod/Text::BibTeX⟩
- YAML ⟨http://metacpan.org/pod/YAML⟩
- YAML::XS ⟨http://metacpan.org/pod/YAML::XS⟩

## A.3 DADOS DO SOFTWARE ACCESSANALYSIS

Listamos aqui, a título de exemplo, os dados coletados para o projeto `s2` (AccessAnalysis), armazenados nos seguintes arquivos:

- `dataset/software/accessanalysis/paper.bib`
- `dataset/software/accessanalysis/software.yml`
- `dataset/software/accessanalysis/search.yml`
- `dataset/software/accessanalysis/references.yml`
- `dataset/software/accessanalysis/releases.yml`

[3] ⟨http://perl.org⟩
[4] ⟨http://cpan.org⟩

**Listing A.1** Arquivo paper.bib do software AccessAnalysis.

```
@INPROCEEDINGS{6392109,
  author={C. Zoller and A. Schmolitzky},
  booktitle={2012 IEEE 12th International Working Conference on Source
     Code Analysis and Manipulation},
  title={AccessAnalysis: A Tool for Measuring the Appropriateness of
     Access Modifiers in Java Systems},
  year={2012},
  pages={120-125},
  keywords={Java;application program interfaces;data encapsulation;
     program compilers;program diagnostics;user interfaces;
     AccessAnalysis;Eclipse IDE;IGAM;IGAT;Java AST API;Java DOM API;
     Java developers;Java systems;class interfaces;inappropriate
     generosity with accessibility of methods;inappropriate generosity
     with accessibility of types;information encapsulation;information
     hiding;minimal access modifiers;plug-in;programming practice;
     static source code analysis;Abstracts;Approximation methods;
     Encapsulation;Java;Measurement;Programming;Software engineering},
  doi={10.1109/SCAM.2012.23},
  month={Sept},
}
```

**Listing A.2** Arquivo software.yml do software AccessAnalysis.

```
software:
  id: s2
  name: AccessAnalysis
  description: Cálculo de métricas IGAT e IGAM (plugin Eclipse)
  distribution: foss
  license: Eclipse Public License
address:
  url: http://accessanalysis.sourceforge.net
  url_available: yes
  download_available: yes
  checked_at: Wed Aug  9 03:24:21 UTC 2017
features:
  input: Source code
  supported_languages: Java
  source_code: Java
life_cycle:
  stage: Initial development
  review: tem 4 versões entre 2010 e 2012, apenas uma mencao
     contribuindo 2012 e nada mais
```

**Listing A.3** Arquivo search.yml do software AccessAnalysis.

```
ieee:
  string: AccessAnalysis
  searched_at: Sun Jul 30 15:46:57 UTC 2017
  results: 3
acm:
  string: content.ftsec:(+AccessAnalysis +Tool +Java +Modifiers)
  searched_at: Sun Aug  6 21:45:40 UTC 2017
  results: 5
```

**Listing A.4** Arquivo references.yml do software AccessAnalysis.

```
p112:
  review: termo AccessAnalysis não encontrado
  is_software_mentioned: no
p214:
  review: artigo selecionado na revisão estruturada
  is_software_mentioned: yes
  mention_type: contribute
  contribution_weight: 1
  conference: SCAM
  step: study1
p222:
  review: são os mesmos autores do artigo selecionado na revisão
     estruturada, publicado no mesmo ano 1 mês depois
  is_software_mentioned: yes
  mention_type: cite
p473:
  review: termo AccessAnalysis faz parte de um trecho de código fonte
  is_software_mentioned: no
p544:
  review: termo AccessAnalysis não encontrado
  is_software_mentioned: no
p660:
  review: termo AccessAnalysis não encontrado
  is_software_mentioned: no
p661:
  review: termo AccessAnalysis não encontrado
  is_software_mentioned: no
p683:
  review: termo AccessAnalysis não encontrado
  is_software_mentioned: no
```

**Listing A.5** Arquivo releases.yml do software AccessAnalysis.

```
# releases: Obsolete
# este projeto foi feito do "jeito certo", origem numa tese e com contribuições posteriores
#developed at the University of Hamburg, Germany. It was initiated in a diploma
#thesis project and enhanced in further research work
# possui informações de uso no site, com ultima data de lençamento, manual e instrucoes de instalacao, etc
# principal contribuir e criador do projeto: Christian Zoller

source: https://sourceforge.net/projects/accessanalysis/files/?source=navbar
# apenas a versao 1.2 tem codigo fonte, as demais apenas .jar binarios
versions:
  - version: 0.17
    source: unavailable
    released_at: 2010-11-22
  - version: 1.0
    source: unavailable
    released_at: 2012-03-15
  - version: 1.1
    released_at: 2012-04-24
    source: unavailable
  - version: 1.2
    source: https://sourceforge.net/projects/accessanalysis/files/1.2/AccessAnalysis-1.2-src.zip/download
    released_at: 2012-05-30
```