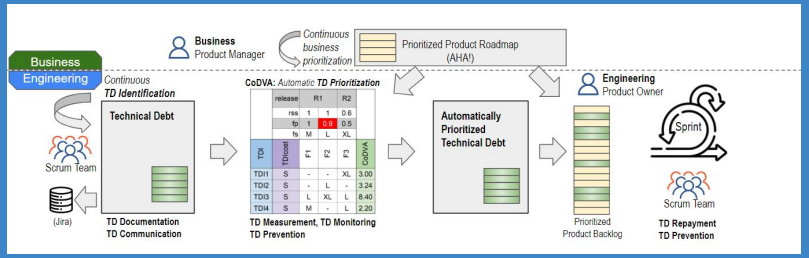


Business-Driven Technical Debt Management Using Continuous Debt Valuation Approach (CoDVA)

Author: Marek Stochel
Affiliation: Motorola Solutions Systems Polska Sp. z o.o., AGH University of Krakow

Date: Dec 1, 2023

Seminarium pod patronatem Sekcji Inżynierii Oprogramowania KI PAN

Agenda

- Introduction
- Background
- Literature review
- Continuous Debt Valuation Approach
- Summary
- Future Work

Introduction

Background: Terminology 1/3

Metaphor	Technical Debt
• **Metaphor:** *"understanding and experiencing one kind of thing in terms of another"* • Connects reason and imagination: is a matter of thinking and cognition • Trying to enforce certain behaviour we use metaphors from an area in which desirable behavior occurs	• **Technical debt metaphor** adopted as a promise to enable budgeting of commonly neglected work

• Cognitive processes are prone to individual differences (an experience is usually unique and personal)	• Individuals have different experiences and a sense of "what should have been done" - metaphor evolved, many individual interpretations appeared
	• Struggle with ambiguity begun
	  ○ The effort to reintroduce the metaphor (W. Cunningham)
	  ○ Mutually exclusive interpretations
	  ○ Avoiding its direct adoption (e.g. Scaled Agile Framework)
	  ○ Evolving interpretation in time (SonarQube)

• Metaphorical understanding changes and will continue to do
  ○ The natural process of language evolution
	• Effort to limit a metaphor applicability will likely fail
	  ○ An attempt to limit another person in expressing their imagination

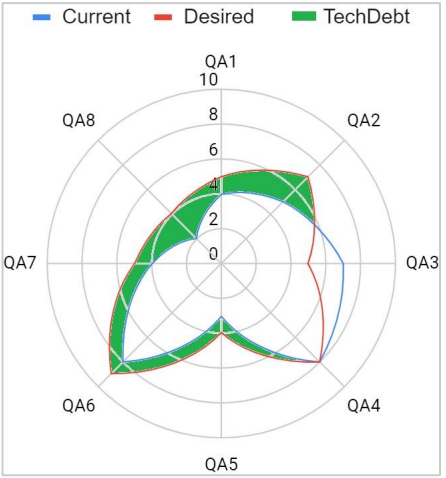
Background: Terminology 2/3

Dagstuhl

- "The *technical debt* (TD) term refers to immature **software development artifacts** which are expedient in the short-term, but introduce delayed consequences, sometimes making the future changes hard or almost impossible."
- "TD embraces a set of actionable product technical debt items (TD Items) indicating these immature artifacts and their **deviation from the desired optimal state**."

Our research

- Change of surrounding solutions (evolving technologies and their adoption by the industry) may be perceived as an act of incurring technical debt against our product, as it results in an **evolution of its desired optimal state**.
- **Context is a key:** what is considered TD in one system may not be such in another, for example when particular functionality becomes the core for one system, and is being retired in another as a result of optimizations towards serving a different customer base.



Technical Debt as compromised quality attributes of SW development artifact

Background: Terminology 3/3

- **Refactoring, repayment** or **remediation**
  - The process of restructuring software artifacts, while not changing their original functionality.
  - The act of removing the difference between the current and the desired state.

- **Contagious debt**
  - If not addressed, may cause other parts of the system to be contaminated with the same problem.
  - Makes both the cost of removing it and its effects growing exponentially.

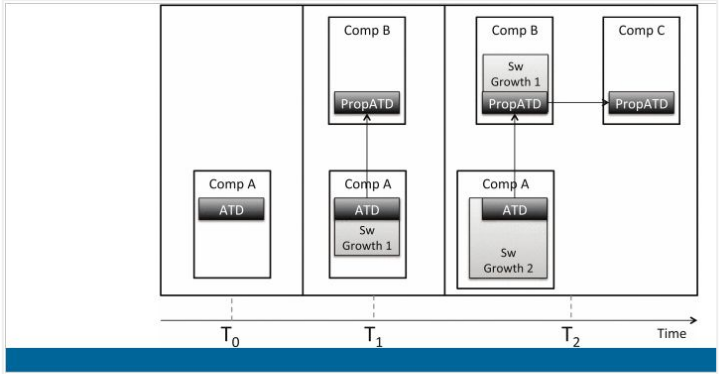


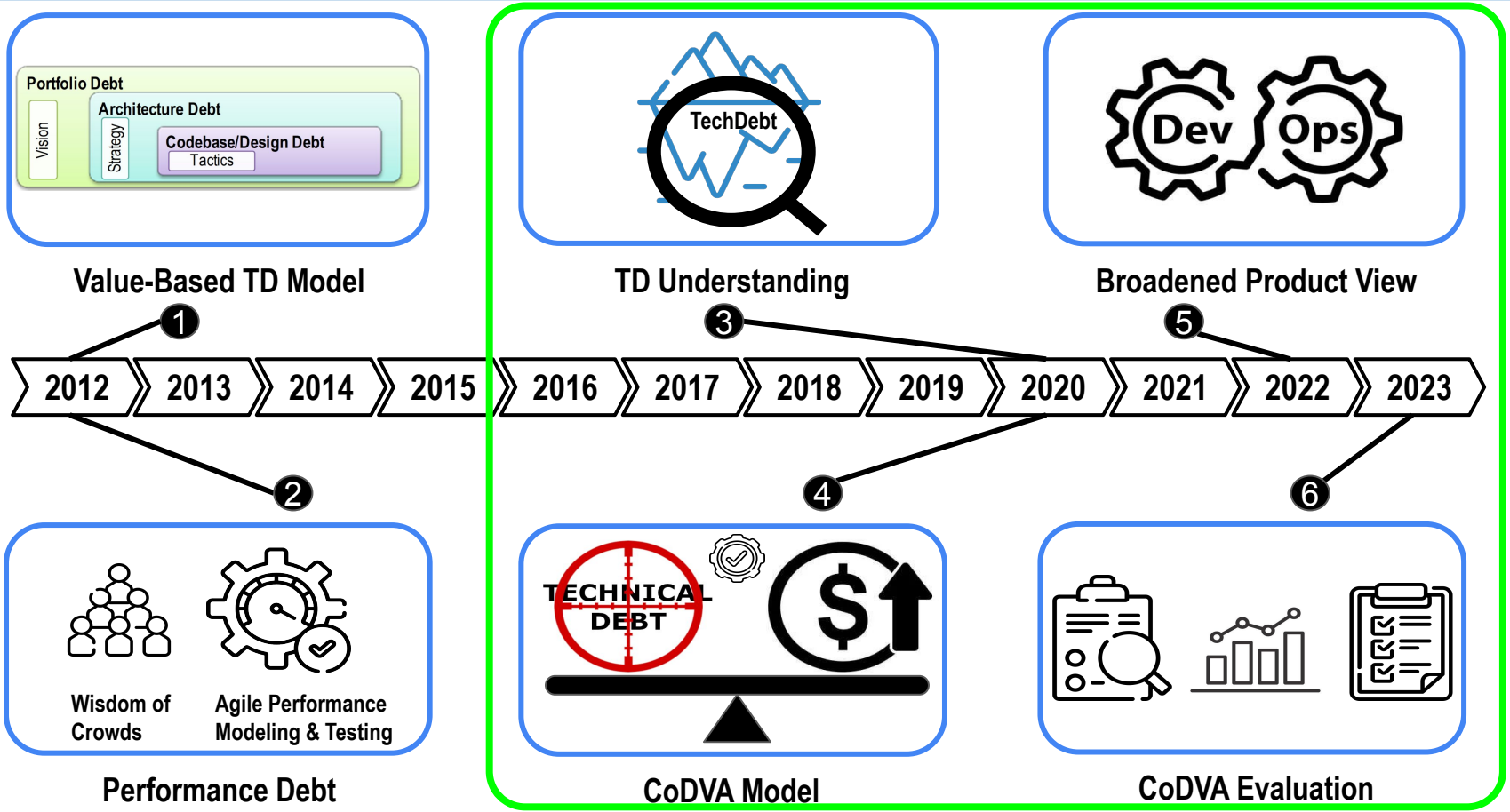
Figure 4.
Model for contagious debt accumulation at different points in time: T_1, T_2 and T_3

Introduction: Problem

How to propose a **TD management approach** that improves

- **Business value**
- **Productivity** of the engineering team
  - Efficiency
  - Effectiveness
- **Developer satisfaction**

Introduction: Contribution - 1/2



Introduction: Contribution - 2/2

Year	Paper Title and Year of Publication	Overview
2011	Marek Stochel. **Reliability of Feedback Mechanism Based on Root Cause Defect Analysis—Case Study**. Annales Universitatis Mariae Curie-Skłodowska. Sectio AI, Informatica, 11(4):21–32, 2011.	Stresses the risk of improper aggregation of data derived from engineering assessments.
2011	Marek Stochel. **Reliability and Accuracy of the Estimation Process—Wideband Delphi vs. Wisdom of Crowds**. In 2011 IEEE 35th Computer Software and Applications Conference, pages 350–359. IEEE, 2011.	Provides comparison of estimation techniques, followed by an overview of the Wisdom of Crowds approach and its applicability.
2012	Marek G. Stochel, Mariusz R. Wawrowski, and James J. Waskiel. **Adaptive Agile Performance Modeling and Testing**. In 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops, pages 446–451. IEEE, 2012.	Stresses the need for extending the TD perspective towards non-functional requirements (performance). Introduces the Wisdom of Crowds approach for TD management.
2012	Marek G. Stochel, Mariusz R. Wawrowski, and Magdalena Rabiej. **Value-Based Technical Debt Model and Its Application**. In 7th International Conference on Software Engineering Advances (ICSEA'12), pages 205–212. IARIA Press, 2012.	Embraces our initial work on technical debt, indicating the need of applying the business (value-based) perspective for effective TD management.
2020	Marek G. Stochel, Piotr Chołda, and Mariusz R. Wawrowski. **On Coherence in Technical Debt Research: Awareness of the Risks Stemming From the Metaphorical Origin and Relevant Remediation Strategies**. In 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 367–375. IEEE, 2020.	Contains the TD research assessment; provides a refined perspective on technical debt (understanding of the term), stressing the need for clarity while defining the research scope in the TD field.
2020	Marek G. Stochel, Piotr Chołda, and Mariusz R. Wawrowski. **Continuous Debt Valuation Approach (CoDVA) for Technical Debt Prioritization**. In 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 362–366, 2020.	Describes the initial version of the CoDVA methodology, which was a result of an exploratory case study.
2022	Marek G. Stochel, Piotr Chołda, and Mariusz R. Wawrowski. **Adopting DevOps Paradigm in Technical Debt Prioritization and Mitigation**. In 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 306–313. IEEE, 2022.	Stresses the need for TD management in a~broadened product context (including DevOps artifacts), as a result of the case study.
2022	Marek G. Stochel, Mariusz R. Wawrowski, and Piotr Chołda. **Technical Debt Prioritization in Telecommunication Applications: Why the Actual Refactoring Deviates From the Plan and How to Remediate It?** Applied Sciences, 12(22:11347), 2022.	Discusses the preliminary results from an introduction of the enhanced CoDVA methodology (case study), additionally provides a Wisdom of Crowds overview in the context of CoDVA.
2023	Marek G. Stochel, Tomasz Borek, Mariusz R. Wawrowski, and Piotr Chołda. **Business-Driven Technical Debt Management Using Continuous Debt Valuation Approach (CoDVA)**. Information and Software Technology, 164:107333, 2023.	Presents a comprehensive examination of the CoDVA approach, including an empirical investigation involving a case study and a survey, as well as a statistical analysis. The findings demonstrate the usefulness and advantages of the methodology.

Introduction: Research questions

Study	Research Question
TD Understanding	RQ_1: Terminology. What terms related to technical debt are used in the surveyed research, and are they consistent with Dagstuhl-TD-Definition?
	RQ_2: Metaphor. Are there any explicit signs of ambiguity in understanding the technical debt terminology (misused terms, inconsistent meaning, disambiguation strategies applied)?
	RQ_3: Model. Are the research results aligned with Dagstuhl-TD-Model?
Broadened Product View	RQ_4: Does prioritization of technical debt following the *broadened product view* (all software artifacts that are important to create value for the customer) allow us to gain faster return on investment (better quality, lower cost, faster delivery)?
CoDVA Model	RQ_5: How may the business perspective (the future feature roadmap) be applied to prioritize technical debt?
CoDVA Evaluation	RQ_6: Does the CoDVA-based TDM improve the value obtained from TD reduction when compared to pure engineering recommendations?
	RQ_7: Does the CoDVA-based TDM influence positively the productivity of the engineering team?
	RQ_8: Does the CoDVA-based TDM improve developer satisfaction?

Background

Background: Research context 1/4

1992 W. Cunningham - introduction of technical debt **metaphor**
"The debt incurred through the speeding up of software project development which results in a number of deficiencies ending up in high maintenance overheads"

2008 S. McConnell - **TD definition**
"A design or construction approach that is expedient in the short term but that creates a technical context in which the same work will cost more to do later than it would cost to do now (including increased cost over time)"

2009 M. Fowler: **Technical Debt quadrant**

2009 W. Cunningham Reintroduction of **metaphor**: https://www.youtube.com/watch?v=pqeJFYwnkjE:
TD means writing code reflecting current understanding, but not writing code poorly and make good job later

2010 Managing Technical Debt workshop → (2018) TechDebt conference; P. Kruchten, R. Nord, I. Ozkaya, N. Brown

2012 I. Ozkaya, P. Kruchten, R. Nord Technical Debt: From Metaphor to Theory and Practice:
TD Landscape: *"We hope to keep this debt metaphor useful by confining it to what is really a debt—namely, the invisible result of past decisions about software that negatively affect its future—and by not extending the concept to anything that has a cost."*

2015 A. Ampatzoglou, P. Avgeriou et al. The **financial aspect** of managing technical debt: A systematic literature review

2015 Z. Li, P. Avgeriou, P. Liang "A systematic mapping study on **technical debt and its management**"

2016 Dagstuhl seminar (P. Avgeriou, P. Kruchten, I. Ozkaya, C. Seaman): definition, conceptual model, dualistic nature (metaphor, and construct indicating immature SW development artifacts)

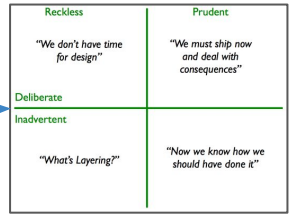
2017 ISO24765 "the deferred cost of work not done at an earlier point in the product life cycle".

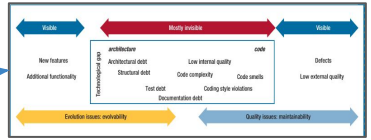
2017 SEAA/Software Engineering and Technical Debt workshop → Software Engineering and Debt Metaphors (SEaDeM) (2022) (**P. Avgeriou, A. Martini**)

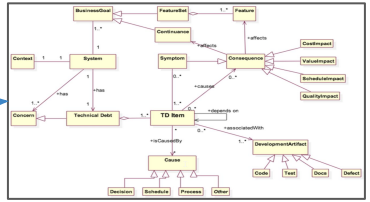
2021 V. Lenarduzzi et al. - A systematic literature review on **Technical Debt prioritization**: 3 papers covering business perspective

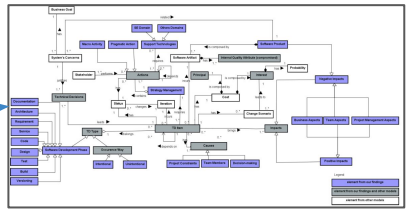
2022 J. Junior & G. Travassos "Consolidating a common perspective on Technical Debt and its Management through a Tertiary Study", limited research on TD management embracing business perspective

2023/2024??? P. Avgeriou Vision paper on Technical Debt covering shift towards value creation



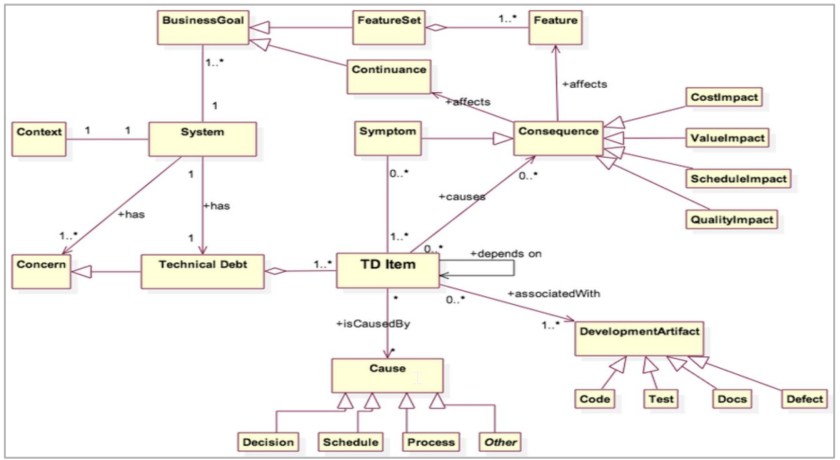






Background: Research context 2/4

- **2010** Managing Technical Debt workshop → **TechDebt conference** (2018)
- **2016** ***Dagstuhl Seminar*** *on Managing Technical Debt in Software Engineering*
  - Converge on common understanding of the technical debt term
  - Scientific rigor, requires crisply defined terms
  - Overloaded nature of the technical debt term accepted
    - A metaphor - facilitates design trade-offs discussions
    - A software development artifact - **indicates immature artifacts incurring delayed costs in the future**
- **2017** Software Engineering and Technical Debt **(SEaTeD)** → SW Engineering and Debt Metaphors workshop (2022)

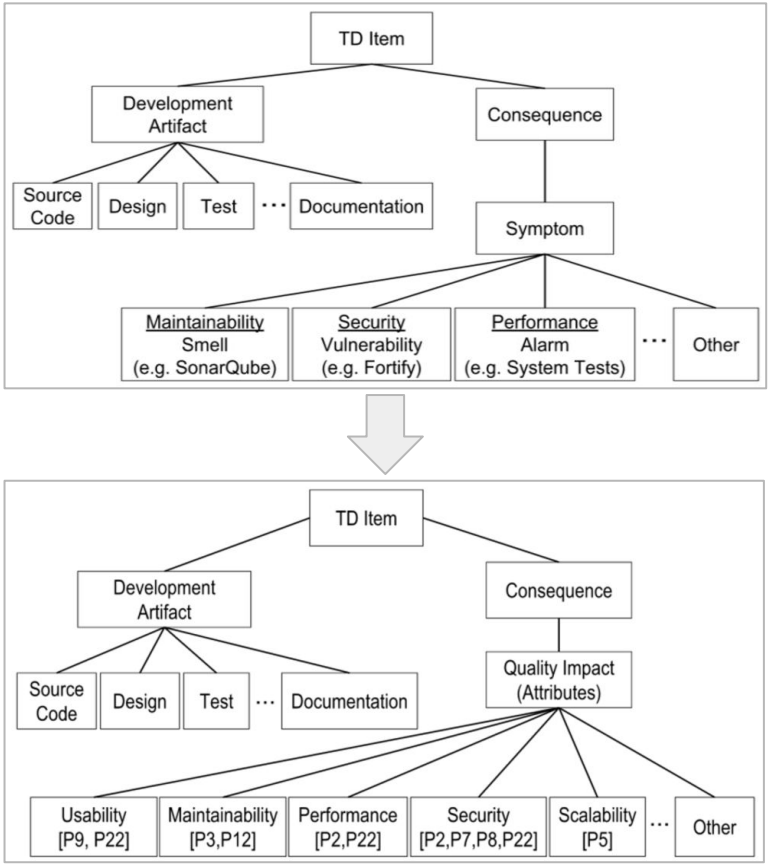


Draft Conceptual Model of Technical Debt (2016)

"In software-intensive systems, technical debt is a collection of design or implementation constructs that are expedient in the short term, but set up a technical context that can make future changes more costly or impossible.
Technical debt presents an actual or contingent liability whose impact is limited to internal system qualities, primarily maintainability and evolvability"

Definition of Technical Debt (2016)

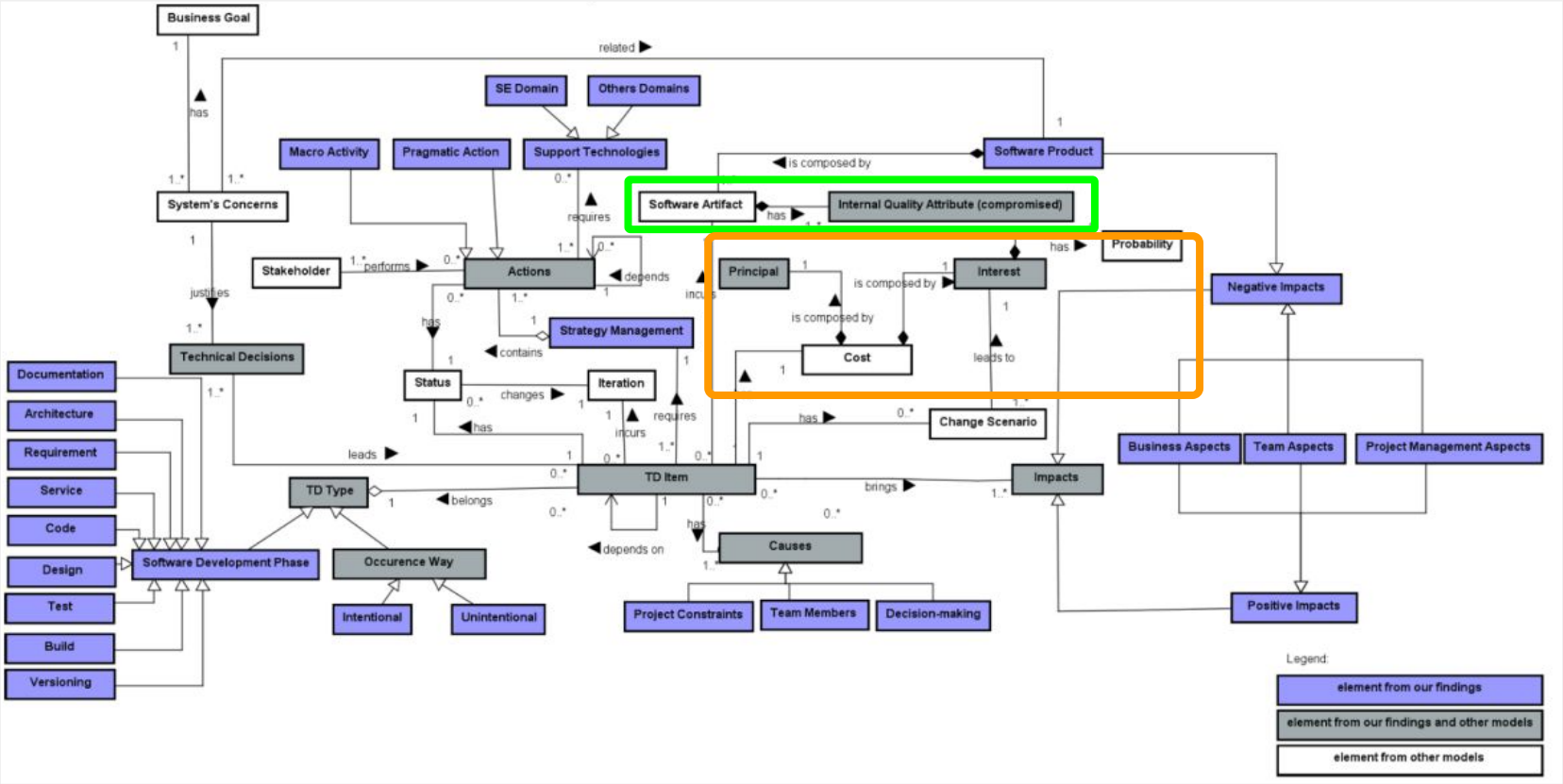
Background: Research context 3/4



Our work on coherence of TD research (2020)

- The conceptual technical debt model is still relevant
- We propose to aggregate the newly appearing technical debt concepts under *Symptom*
- Understanding technical debt referring to ***a set of development artifacts, focusing on quality impact*** may increase research consistency even further
  - Leading to determining ***Consequences*** and their ***Quality Impact***

Background: Research context 4/4



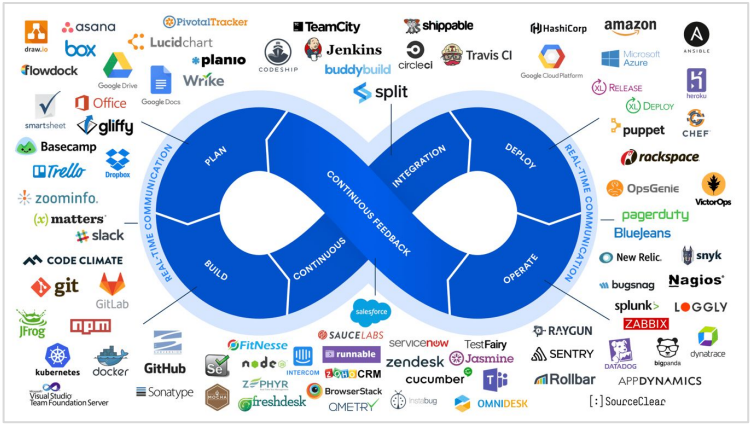
Junior & Travassos: Conjectured conceptual model of TD (2022)

Background: Product context 1/2

- What is a product?

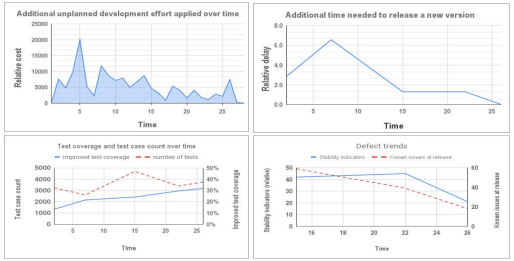
Broadened product view encompasses all development artifacts which enable value creation for a customer. As a result, tools, automated tests, testing environments, logging, and monitoring capabilities become integral components of this perspective.

- The more a product becomes a service, the more such robust perspective is required.



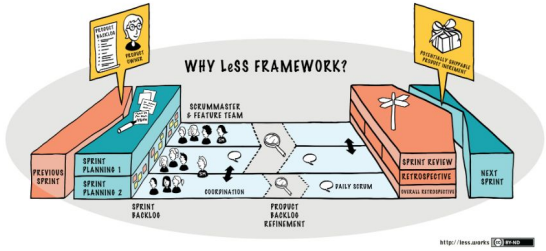
Source: SDTimes: Atlassian unveils new DevOps Marketplace and the Atlassian Stack

Background: Product context 2/2



Broadened Product View (embracing DevOps)

- Our Exploratory Case Study (2016-2018) - published in 2022
- Major steps
  - **Protection**: enhanced automated test environment
  - **Predictability** of development efforts (cost & time)
  - **Speed**: Improved development environment towards faster turnaround time for any new code change
  - Experiments with more **risky code changes**, possibly causing regression
  - **Architectural** TD remediation
- Each phase was an enabler/accelerator to the next one
- More artifacts identified (DevOps, DevSecOps, ...)



Large Scale Scrum (LeSS) Framework (2014 - 2023)

- Who are the actual end-customers and what do they consider the product to be?
- What is the original problem that the product is solving?

Bosch Software Defined Vehicle

Euromicro DSD/SEAA '22
Keynote: Dr. Arne Hamann - "Designing Reliable Distributed Systems"

Background: Value 1/2

Technical debt follows the **financial debt construct** (Ampatzoglou et al, 2015)

- **Principal** - effort required to bring SW development artifacts to the optimal state
- **Interest** - additional effort to be spent on maintaining SW development artifacts in their current suboptimal state (Li, Avgeriou, Liang et al 2013)

Mapping Study on quantifying TD interest (Arvanitou, Ampatzoglou, 2022)

- Lack of mathematical/analytical methods in this area
- Calculating TD interest is challenging
  anticipating future SW changes and quantifying the additional maintenance effort required
- Scientific problem is far from being resolved

Our proposal to apply business perspective: **TD interest → value creation**

- Business perspective ensures the **clarity (the product roadmap)** engineering team requires
- **No universal formula** of determining the exact product value exists
- We prefer *business valuation of the feature roadmap* to *engineering TD interest quantification*

Background: Value 2/2

Business Prioritization and Technical Debt Management

- **Business prioritization** guides the engineering team, led by the Product Owner, on the **order and importance of feature development**.
- Our proposed approach is **relative valuation**, which is highly relevant for effective prioritization.
- Among the various approaches discussed in the literature, only our approach offers a specific **business-driven formula** for managing technical debt.

Table 2.4: Product value: factors taken into account in TD prioritization

Year	Author (method)	Business factors
2012	Stochel et al. [183] (Value-Based TD Model)	product vision (portfolio) profitability (sales)
2015	Martini [128]	market attractiveness specific customer value lead time maintenance cost customer long-term satisfaction risk penalty volatility
2015	Ramasubbu et al. [157]	level of reliability required by the business likelihood of technological disruptions customer satisfaction needs
2016	Gupta et al. [73][a]	profitability scalability time-to-market core business, i.e. existing workflows
2018, 2019	Rebouças et al. [46, 47] (Tracy *initial*)	impacted business processes; sample metrics: – availability – cost – customer relationship – market share – revenue – sales efficiency – sales volume
2020	Stochel et al. [186] (CoDVA initial)	profitability
2021, 2022	Rebouças et al. [48], da Silva et al. [44] (Tracy)	core business value sources usage frequency risk perception time flexibility incidents delivery frequency external regulations or certifications market versus internal perspective
2023	Stochel et al. [188, 189][b] (CoDVA enhanced)	relative *business priority*, aggregates e.g.: - potential benefits - investment size - strategic aspects - time criticality

[a] Even though not explicitly stated in the paper, a few factors were revealed.
[b] The second paper covers the final version of the CoDVA methodology, which is presented in Chapter 7 of this dissertation. It was accepted for publication on Sep 17, 2023.

Technical Debt research: product value

Literature Review

Literature: Knowledge Synthesis Studies

- Limited success in searching scientific databases for relevant research
- Three approaches identified considering business perspective (Lenarduzzi, 2021)
- Snowballing technique (Wohlin, 2014) on
  - Systematic Literature Review
  - Systematic Mapping Study
  - Tertiary Study
  - Multivocal Literature Review

ID	Year	Type	Goal
Tom et al. [197]	2012	SLR	Understanding of TD implications for SW development
Tom et al. [198]	2013	multivocal SLR	Consolidated view of TD and implications for SW development
Ampatzoglou et al. [11]	2015	SLR	Financial approaches for managing TD
Li et al. [118]	2015	SMS	TD management, TD classification
Fernández-Sánchez et al. [61]	2015	SMS	TD types, TD management elements
Poliakov [152]	2015	SMS	TD definition
Alves et al. [9]	2016	SMS	TD management strategies, TD taxonomy
Ribeiro et al. [160]	2016	SMS	TD payment prioritization
Behutiye et al. [20]	2017	SLR	TD in Agile development
Fernández-Sánchez et al. [63]	2017	SMS	TD management elements
Besker et al. [23]	2018	SLR	Architectural TD management
Rios et al. [161]	2018	tertiary	TD types, management strategies
Alfayez et al. [7]	2020	SLR	TD effort
BenIdris et al. [21]	2020	SMS	TD types, indicators, and estimators
Khomyakov et al. [97]	2020	SLR	TD tools
Klimczyk and Madeyski [102]	2020	SMS	TD aware estimations
Lenarduzzi et al. [114]	2020	SLR	TD prioritization
Pina et al. [150]	2021	SMS	TD prioritization taxonomy
Arvanitou et al. [14]	2022	SMS	TD interest
Junior and Travassos [94]	2022	tertiary	TD management
Melo et al. [134]	2022	SLR	TD of requirements
Villa et al. [207]	2022	SMS	TD in microservices
Albuquerque et al. [6]	2023	SMS	TD management using intelligent techniques
Kleinwaks et al. [101]	2023	SLR	TD in systems engineering
Murillo et al. [141]	2023	SMS	TD identification and management
Perera et al. [146]	2023	SMS	Quantification of TD
Zabardast et al. [217]	2023	SLR, field study	TD in the context of quality degradation of development artifacts

Technical Debt - knowledge synthesis studies

Literature: Business-driven TD perspective 1/2

Id	Business-Driven TDM (Author, Year)	Description
S1	Net Present Value [49] (Nugroho et al., 2011)	Empirical model of TD and interest, comparing maintenance effort at a specific quality level to the optimal level
S2	SQALE [50,51] (Letouzey, 2012–16)	Software quality assessment based on life-cycle expectations, utilizes ISO25010 quality perspective to evaluate source code quality; it is implemented by several tools including SonarQube, NDepend, and Squore
S3	Architectural debt rework model [52] (Nord et al. 2012)	Metric for architectural TD to optimize development cost
S4	Cost-Benefit Analysis, Analytic Hierarchy Process, Portfolio, and Options [53] (Seaman et al. 2012)	Four decision approaches that can be used to incorporate TD information in product release planning
S5	Prioritization of architectural TD [54] (Martini and Bosch, 2015)	Recommendation on prioritization aspects needed by decision-makers to prioritize architectural TD with respect to feature development
S6	Recommendations for managing TD [55] (Ramasubbu et al., 2015)	Three-dimensional prioritization: level of reliability required by the business, customer satisfaction needs, and probability of technology disruption
S7	Pragmatic approach [56] (Gupta et al., 2016)	Repayment of code-based TD that ultimately improved product scalability
S8	TDM framework [57] (Yli-Huumo, 2016)	TDM framework developed as a result of exploratory case study; defines activities, practices/tools, stakeholders, and responsibilities of TDM on three maturity levels
S9	Tracy [58–61] (Rebouças et al., 2018–21) (da Silva et al. 2022)	Business-driven TD prioritization framework
S10	CoDVA [11,12], this paper (Stochel et al., 2020–23)	Business-driven TD management based on Continuous Debt Valuation Approach

Implemented by vFunction platform

TD management studies considering business-perspective

Literature: Business-driven TD perspective 2/2

TDM study Id	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Product perspective considered	✓	✓	✓	✓	.	.	✓	✓	✓	✓
Value quantified	✓	✓	✓	.	.	.	✓	.	✓	✓
Guidelines available (reproduction protocol)	✓	✓	✓	✓	.	.	✓	✓	✓	✓
Empirical evaluation	✓	✓	✓	.	✓	✓	✓	✓	✓	✓
Business perspective considered	.	✓	✓	.	✓	✓	✓	✓	✓	✓
Business stakeholders engaged (e.g. product management)	.	.	.	.	.	.	.	.	✓	✓
TDM process (protocol) defined	.	.	.	.	.	.	✓	✓	.	✓
Broadened product view	.	.	.	.	.	.	.	.	.	✓
Future business focus	.	.	✓	.	.	✓	.	.	.	✓
Decision-making autonomy (business/engineering)	.	.	.	.	.	.	.	.	.	✓
Statistical evaluation of approach/consequences	.	.	.	.	.	.	.	.	.	✓

Comparison of the selected TD management studies

Continuous Debt Valuation Approach (CoDVA)

Methodology: CoDVA idea

Business

Keith Eades “The New Solution Selling”

- Streamlining the selling process
- Differentiating from the competition
- Decreasing the time spent between initial *Qualifying* and a successful, profitable *Close*.

Table 7.1: Solution selling pipeline milestone chart [56]

Id	Milestone	Win Odds	Revenue [$]	Yield [$]
[T]	Territory	0%	—	—
[S]	Qualified Suspect	10%	sales-at-S	10% × sales-at-S
[D]	Qualified Sponsor	25%	sales-at-D	25% × sales-at-D
[C]	Qualified Power Sponsor	50%	sales-at-C	50% × sales-at-C
[B]	Decision Due	75%	sales-at-B	75% × sales-at-B
[A]	Pending Sale	90%	sales-at-A	90% × sales-at-A
[W]	Win	100%	sales-at-W	100% × sales-at-W

Engineering

Alignment

- Evaluating TD in relation to a predicted future company portfolio
- Relative prioritization of technical debt items
- Shift “TD interest” → “value generation”

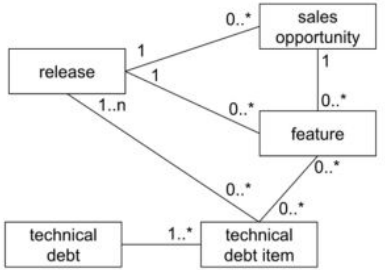


Fig. 7.1: Alignment of the business perspective with the conceptual model for TD [15]

Methodology: Initial Case Study

Initial exploratory Case Study

- Cost-benefit analysis
- Monetary-based evaluation
- Retrospective analysis

Suggestion of improvements

- TD items interdependence
- Long-term perspective
- Process: accuracy, cost and adoption

$$\Delta effort_i = \sum_{j=1}^{m} (fs_j \times effortsaved_j) - TDIc_i$$

where:

effortsaved – effort saved during feature implementation,
fs – feature size (feature cost),
TDIc – TDI remediation cost,
m – number of features.

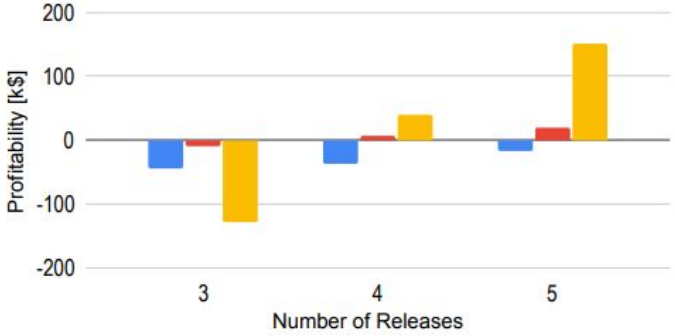


Fig. 7.7: Profitability of technical debt repayment

CoDVA - initial study

Methodology: CoDVA (final)

$$CI_i = \frac{\sum_{j=1}^{m} fp_j \times fs_j \times rss_j \times TDIe_{ij}}{TDIc_i} + \sum_{k=1,k\neq i}^{n} d_{ik} \times CI_k$$

where:

fp–feature priority,

fs–feature size,

rss–roadmap scope stability,

$TDIe$–TDI effect,

$TDIc$–TDI cost,

CI_k–$CoDVA\_index$ value for dependent TDIs,

d_{ik}–dependency coefficient between TDI_i and TDI_k,

m–number of features,

n–number of TDIs.

CoDVA_index metric (CI): a relative cost-benefit ratio reflecting investment in a given TD item

- Can be computed at any moment
- Determine the priority of allocating effort for TDI reduction

Methodology: CoDVA→Feature Priority

- Roadmap features are prioritized by the Product Manager in Aha!
  - Scope for subsequent releases
- Each ***feature*** is assigned to a particular ***future release*** and receives a ***single business prioritization value bp***
- Any ***bp*** aggregates several aspects, such as: potential benefits, investment size, strategic aspects, and time criticality
- Normalized value for an *ith* feature fp_i, on a roadmap containing n features, is taken to calculate CoDVA_index

$$fp_i = bp_i / \max\{bp_1, \ldots, bp_n\}$$

$$CI_i = \frac{\sum_{j=1}^{m} fp_j \times fs_j \times rss_j \times TDIe_{ij}}{TDIc_i} + \sum_{k=1,k\neq i}^{n} d_{ik} \times CI_k$$

where:

fp–feature priority,
fs–feature size,
rss–roadmap scope stability,
$TDIe$–TDI effect,
$TDIc$–TDI cost,
CI_k–$CoDVA\_index$ value for dependent TDIs,
d_{ik}–dependency coefficient between TDI_i and TDI_k,
m–number of features,
n–number of TDIs.

From the development perspective each feature has a single priority number fp_i reflecting the business needs.

Methodology: CoDVA→Feature Size

- Roadmap features are estimated using T-shirt sizing
  - XL - scaled to fit in a single release
  - L - greater than half of the release
  - M - 1/5 of the release
  - S - 1/10 of the release

  Hence {S; M; L; XL} correspond to coded values {0.1; 0.2; 0.6; 1}
- If a feature cannot be realized in one release, it is split and released in consecutive ones
- A half-done feature might be hidden behind a feature flag if the scope is insufficient for the customers to use

$$CI_i = \frac{\sum_{j=1}^{m} fp_j \times fs_j \times rss_j \times TDIe_{ij}}{TDIc_i} + \sum_{k=1,k\neq i}^{n} d_{ik} \times CI_k$$

where:

fp–feature priority,
fs–feature size,
rss–roadmap scope stability,
$TDIe$–TDI effect,
$TDIc$–TDI cost,
CI_k–$CoDVA\_index$ value for dependent TDIs,
d_{ik}–dependency coefficient between TDI_i and TDI_k,
m–number of features,
n–number of TDIs.

T-shirt sizing
- Used to focus on high-level project goals
- Rough estimation of the effort involved
- Involves assigning relative T-shirt size estimates {S; M; L; XL} to different tasks or features based on their perceived complexity or the required effort
- Low-cost and quick (enables evaluation of a high number of items)

Planning Poker (Scrum Poker)
- Used for short-term sprint planning
- Uses story points
- More detailed estimation

Methodology: CoDVA→Roadmap Scope Stability

- Roadmap scope stability (rss) - the prediction to what extent the originally planned scope can be achieved in a future release.
- Calculation
  - Releases have roughly constant cadence
  - Value determined retrospectively based on the completed releases, using a simple moving average (percentage of unchanged scope) from the preceding year releases.

$$rss = \frac{1}{n}\sum_{i=1}^{n} s_i$$

  - For ith release in the future we decrease rss value according to the formula:

$$rss_i = rss^{i-1}$$

For current release $rss^0 = 1$; if feature belongs to the next: rss^1, etc.

$$CI_i = \frac{\sum_{j=1}^{m} fp_j \times fs_j \times rss_j \times TDIe_{ij}}{TDIc_i} + \sum_{k=1,k\neq i}^{n} d_{ik} \times CI_k$$

where:

fp–feature priority,
fs–feature size,
rss–roadmap scope stability,
$TDIe$–TDI effect,
$TDIc$–TDI cost,
CI_k–$CoDVA\_index$ value for dependent TDIs,
d_{ik}–dependency coefficient between TDI_i and TDI_k,
m–number of features,
n–number of TDIs.

In a **continuous delivery approach**, the rss parameter can be used to gauge scope stability over a set period, e.g. a quarter. Scope is determined based on the predicted capability of the engineering team.

Methodology: CoDVA→TDI Effect

- TDI Effect - impact of a TDI on a new feature, indicating relative potential benefits:
  - Mitigation of risks
  - Potential decrease in the feature implementation effort
  - Quality
- Estimated by the development team using T-shirt sizing, serving as an input for prioritization.
- Values {S; M; L; XL} corresponds to coded values {0.1; 0.2; 0.6; 1}, following the pattern established for *feature size*

$$CI_i = \frac{\sum_{j=1}^{m} fp_j \times fs_j \times rss_j \times TDIe_{ij}}{TDIc_i} + \sum_{k=1,k\neq i}^{n} d_{ik} \times CI_k$$

where:

fp–feature priority,
fs–feature size,
rss–roadmap scope stability,
$TDIe$–TDI effect,
$TDIc$–TDI cost,
CI_k–$CoDVA\_index$ value for dependent TDIs,
d_{ik}–dependency coefficient between TDI_i and TDI_k,
m–number of features,
n–number of TDIs.

TD Items	Features								
	F1	F2	F3	F4	F5	F6	F7	F8	F9
TDI-1			XL	L					
TDI-2			L	L					
TDI-3			XL	L					
TDI-4			XL	XL					
TDI-5			S	L					
TDI-6			XL	S					
TDI-7	L	XL	XL	XL	M	M	M	L	M
TDI-8	XL	L	XL	M	S	M	M	-	S

Specific TDIs

Generic TDIs

Generic TD item - may have a positive impact on a high number of the product roadmap features. Its prioritization is minimally affected by changes in the product roadmap.

Specific TD item - may only have a positive impact on a small number of the product roadmap features. It is susceptible to prioritization changes when business priorities evolve

Methodology: CoDVA→TDI Cost

- TDI Cost - cost of TDI reduction
- Estimated using T-shirt sizing
  - Values {S; M; L; XL} corresponds to coded values {0.1; 0.2; 0.6; 1} for consistency
    - XL - scaled to fit in a single release
    - L - greater than half of the release
    - M - ⅛ of the release
    - S - □ of the release

$$CI_i = \frac{\sum_{j=1}^{m} fp_j \times fs_j \times rss_j \times TDIe_{ij}}{TDIc_i} + \sum_{k=1, k \neq i}^{n} d_{ik} \times CI_k$$

where:

fp–feature priority,
fs–feature size,
rss–roadmap scope stability,
$TDIe$–TDI effect,
$TDIc$–TDI cost,
CI_k–$CoDVA\_index$ value for dependent TDIs,
d_{ik}–dependency coefficient between TDI_i and TDI_k,
m–number of features,
n–number of TDIs.

For TDIs, our empirical experience is that they have to be split even if they are of size L, and XL. Otherwise, they would be postponed indefinitely, as the budget allocated for TD reduction is a fraction of the budget allocated for the release (usually 20% of effort).

Methodology: CoDVA→Dependent CIs

- **CoDVA_index value for dependent CIs**
  - Involves evaluating the dependencies between TDIs
  - Typical in high-complexity projects where large elements need to be decomposed to identify enablers
- **Enablers**
  - A collection of smaller TDIs forms a tree structure where independent TDIs (enablers) are located at the leaves
  - Enablers can be addressed quickly, allowing for iterative corrective actions
- **Dependency indication**
  - The coefficient d = 1 when a dependency exists, or d = 0 when there is none

$$CI_i = \frac{\sum_{j=1}^{m} fp_j \times fs_j \times rss_j \times TDIe_{ij}}{TDIc_i} + \sum_{k=1,k\neq i}^{n} d_{ik} \times CI_k$$

where:

fp–feature priority,
fs–feature size,
rss–roadmap scope stability,
$TDIe$–TDI effect,
$TDIc$–TDI cost,
CI_k–$CoDVA\_index$ value for dependent TDIs,
d_{ik}–dependency coefficient between TDI_i and TDI_k,
m–number of features,
n–number of TDIs.

Methodology: CoDVA Example

Preparation

- Two planned product releases R1, R2 that include three new features F1, F2, and F3, of size: M, L, XL.
- The Product Manager provides normalized feature priorities: fp1=1, fp2=0.9, fp3=0.5.
- Four TDIs whose impact (TDIe) on the features and cost (TDIc) is estimated by dev team.
- Roadmap scope stability, rss = 60%.

	release	R1		R2	
	rss	1	1	0.6	
	fp	1	0.9	0.5	
	fs	M	L	XL	
TDI	TDIcost	F1	F2	F3	CoDVA
TDI1	S	-	-	XL	3.00
TDI2	S	-	L	-	3.24
TDI3	S	L	XL	L	8.40
TDI4	S	M	-	L	2.20

(a) Initial state.

	release	R1		R2	
	rss	1	1	0.6	
	fp	1	0.6	0.5	
	fs	M	L	XL	
TDI	TDIcost	F1	F2	F3	CoDVA
TDI1	S	-	-	XL	3.00
TDI2	S	-	L	-	2.16
TDI3	S	L	XL	L	6.60
TDI4	S	M	-	L	2.20

(b) Priority change of $F2$.

Scenario (a) - initial

$$CI_1 = (fp_3 \times fs_3 \times rss^{2-1} \times TDIe_{f_3, TDI_1}) / TDIc_{TDI_1}$$
$$CI_1 = (0.5 \times 1 \times 0.6^1 \times 1) / 0.1 = 3.00$$

Scenario (b)
feature F2 priority value drops (fp2: 0.9→ 0.6).

CI_1 = 3.00, CI_2 = 3.24, CI_3 = 8.40, CI_4 = 2.20

Prioritisation: TDI_3, TDI_2, TDI_1, TDI_4

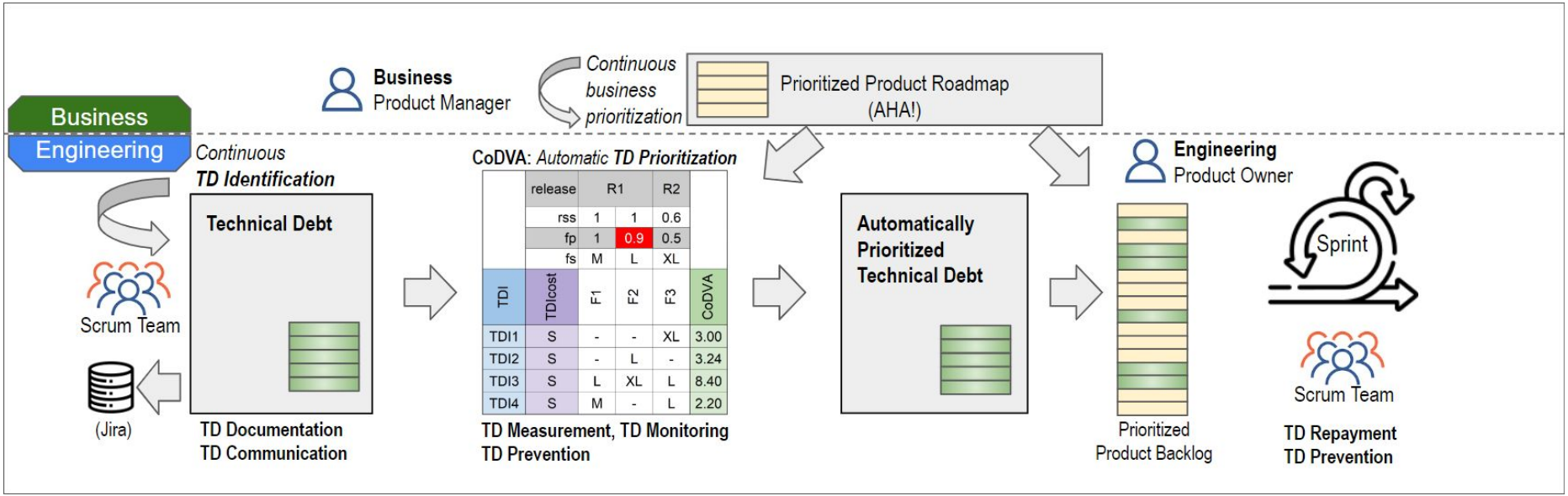
Specific TDI

Generic TDI

CI_1 = 3.00, CI_2 = 2.16, CI_3 = 6.60, CI_4 = 2.20

Prioritisation: TDI_3, TDI_1, TDI_4, TDI_2

TD Management Process



Business-driven TDM process based on CoDVA

CoDVA: Evaluation

How to evaluate effectiveness of the methodology?

- Technical debt valuation is context-dependent
- Concept of cost (principal, interest) hard to incorporate

Solution: shift towards ***value creation***, and observing the ***consequences*** on the application of the methodology

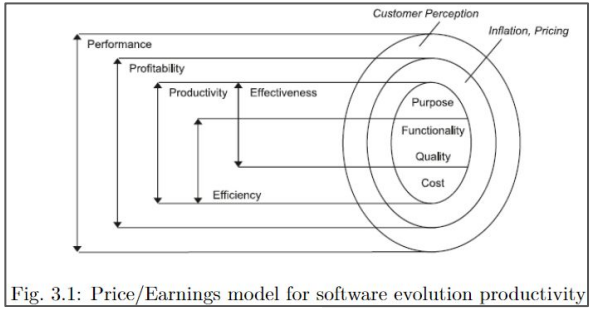


Fig. 3.1: Price/Earnings model for software evolution productivity

Wagner and Deissenboeck, 2019

Profitability *(value creation)*

- **RealizedBusinessValue** metric, represented by the CoDVA_index, a relative benefit–cost ratio reflecting investment in a given TDI. TD repayment activities, influence value generation - relative valuation derived from the business perspective of the feature roadmap.

Productivity

- ***Efficiency***
  - **Velocity** - the amount of work delivered by the engineering team in a sprint, measured in story points.
  - **DefectFixTime** - the active resolution time for a defect (time spent in the *In Progress* state).
- ***Effectiveness***
  - **DeliveryPredictability** - the ratio of fully successful sprints in the release, where all sprint completion criteria are consistently met, including the sprint goal.
  - **Releasability** - a leading indicator of the final product quality, indicates the effort needed to maintain an acceptable level of quality, measured by the number of full regression test runs required until the quality criteria are met.

Developer morale *(satisfaction)*

- Countermeasure for efficiency and effectiveness.

CoDVA: Evaluation

Case study

- Three-year-long case study (Jan. 2020–Mar. 2023)
- Large international company that creates, maintains and operates mission-critical telecommunication systems
- A single, embedded case study in which the case was defined as introduction of the CoDVA-based TDM by an engineering team focused on development and maintenance of software managing a fleet of devices for a video surveillance system
- The units of analyses were phases of the introduction of the method (sub-units were product releases)

Case study phases vs. product releases.

Phase	Release	Description
Forming	R1	The team members joined and learned the codebase.
Init	R2–R3	The first features were delivered and plans for TD reduction were created.
Repayment	R4–R7	Active TD reduction according to CoDVA was applied together with new feature development.
Monitor	R8–R9	No significant TD reduction happened, the prior TD reduction changes were stabilized, and the results were analyzed.

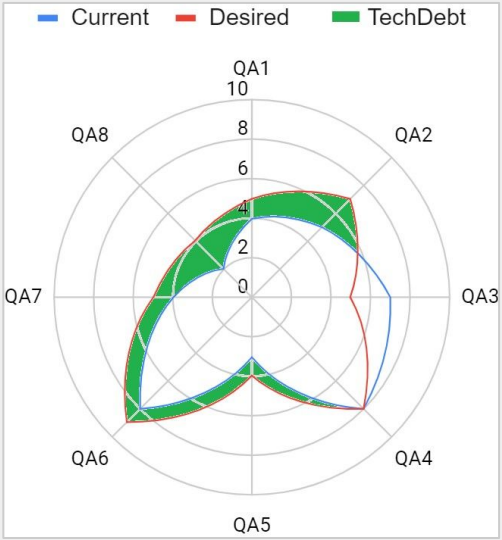
Survey

- Focused on aspects of developer satisfaction
- Built in a way to avoid response bias
  - Neutral wording
  - Question order randomly chosen
  - Used a 7-point Likert type answers (more nuanced responses, ability to discriminate)
- Pilot run executed to ensure the survey is capable to answer research questions

CoDVA: Evaluation - Identified TD items

TDI	Main Quality Characteristics
TDI01	Maintainability→Testability
TDI02	Maintainability→Testability
TDI03	Maintainability→Modifiability
TDI04	Maintainability→Testability
TDI05	Maintainability→Testability
TDI06	Maintainability→Modularity
TDI07	Security→Integrity
TDI08	Maintainability→Testability
TDI09	Maintainability→Analysability
TDI10	Portability→Adaptability
TDI11	Security→Confidentiality
TDI12	Maintainability→Modifiability
TDI13	Maintainability→Modifiability
TDI14	Maintainability→Modifiability
TDI15	Maintainability→Modifiability
TDI16	Maintainability→Analysability
TDI17	Maintainability→Testability
TDI18	Maintainability→Analysability
TDI19	Maintainability→Testability
TDI20	Maintainability→Analysability
TDI21	Reliability→Recoverability
TDI22	Security→Integrity
TDI23	Maintainability→Reusability
TDI24	Maintainability→Testability
TDI25	Portability→Installability
TDI26	Maintainability→Testability
TDI27	Maintainability→Testability
TDI28	Maintainability→Testability
TDI29	Maintainability→Analysability
TDI30	Maintainability→Analysability
TDI31	Usability→UI aesthetics[a]
TDI32	Suitability→Func. completeness[b]
TDI33	Usability→UI aesthetics[a]
TDI34	Usability→Operability

TDI	Quality Characteristics
TDI08	Maintainability→Testability Performance efficiency→Time behavior Performance efficiency→Resource utilization Performance efficiency→Capacity
TDI12	Maintainability→Modifiability Maintainability→Modularity Maintainability→Testability
TDI15	Maintainability→Modifiability Maintainability→Testability
TDI16	Maintainability→Analysability Maintainability→Testability
TDI18	Maintainability→Analysability Maintainability→Testability
TDI29	Maintainability→Analysability Performance efficiency→Resource utilization
TDI32	Suitability→Functional completeness Usability→Operability



Technical Debt as compromised quality attributes of SW development artifact

CoDVA: Evaluation (RQs)

RQ1: Does the CoDVA-based TDM improve the value obtained from TD reduction when compared to pure engineering recommendations?

- Evaluation of practical implementation of the proposed approach
- Assessment of realized business value

RQ2: Does the CoDVA-based TDM influence positively the productivity of the engineering team?

- Efficiency - velocity and defect-fixing time
- Effectiveness - delivery predictability and time-to-market while maintaining a desired quality level

RQ3: Does the CoDVA-based TDM improve developer satisfaction?

- Developer morale (affected by TD presence)

CoDVA: Evaluation

RQ1

(1) Null Hypothesis $H1_0$ assumes both approaches: *CoDVA_real* and *Eng-Choice* are equally capable of realizing business value. Whereas Alternative Hypothesis $H1_A$ assumes they differ and *CoDVA_real* results in faster accumulation of business value.

RQ2

(2) Null Hypothesis $H2_0$: assumes *Velocity* of the engineering team does not change. Whereas Alternative Hypothesis $H2_A$: assumes *Velocity* increases in the *monitor* phase after the CoDVA-based TD reduction has happened, in comparison with the *init* phase.

(3) Null Hypothesis $H3_0$: assumes the CoDVA-based TD reduction does not impact positively *DefectFixTime*. Alternative Hypothesis $H3_A$: assumes there is a positive difference (decrease) in the time to resolve a defect in the *monitor* phase in comparison with the *init* phase.

(4) Null Hypothesis $H4_0$: assumes there is no difference between the ratio of successful sprints between the *init* and *monitor* phases, meaning that the CoDVA-based TD reduction does not impact *DeliveryPredictability*. Alternative Hypothesis $H4_A$: assumes there is a difference and the percentage of successful sprints is higher in the *monitor* phase.

(5) Null Hypothesis $H5_0$: assumes there is no difference in the effort spent on the final regression tests between the *init* and *monitor* phases, meaning that the CoDVA-based TD reduction does not impact the *Releasability* value. Alternative Hypothesis $H5_A$: assumes there is a positive change (effort decrease) in the *monitor* phase.

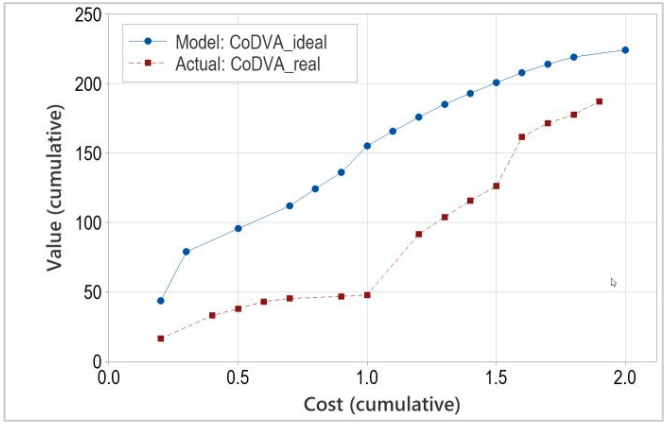
RQ3

(6) Null Hypothesis $H6_0$: assumes the cumulative developer perception of trends during their assignment to the product is neutral, meaning that the CoDVA-based TD reduction did not impact their satisfaction. Alternative Hypothesis $H6_A$: assumes there is a positive difference.

CoDVA: Evaluation

Hypothesis	Results
(1) H_A	The *CoDVA_real* approach is able to generate business value faster than pure engineering recommendations *EngChoice*. The perceived business value from TD refactorings raised by 27%.
(2) H_A	Engineering *velocity* increased by 39%.
(3) H_0	*DefectFixTime*: Even though the time spent on fixing issues grew, we concluded that the change was positive from the product quality perspective: easy-to-fix issues were resolved in early phases of the project, better test coverage prevented the issues from escaping in the added or modified code, the new code was developed with higher scrutiny, and engineers started proactively looking for hidden bugs which might be harder to investigate and resolve.
(4) H_A	*DeliveryPredictability* improved by 60%.
(5) H_A	*Releasability*: The effort spent on product stabilization decreased by 50%.
(6) H_A	*DeveloperSatisfaction*: developer satisfaction grew, the majority of the developers perceived the applied TDM strategy beneficial across the technical decisions made, ability to develop new functionality, and experience while working with the product code.

CoDVA: Evaluation



Actual implementation did not follow exactly CoDVA recommendation (reasons):

- Continuous refactoring
- TDIs interdependency
- Cost

Comparison

- **CoDVA_real** - Actual TD repayment according to CoDVA
- **EngChoice** - Expected budget allocation if the CoDVA recommendation not implemented

Observations

- The median of EngChoice (5.14) is almost equal to the first quartile of CoDVA_real (5.02), and twice smaller than the CoDVA_real median (10.11).
- In comparison with the pure engineering choice (CumulativeValue = 147), the relative business return on engineering investment grew by 27% when the CoDVA-based TDM was used (CumulativeValue = 187).
- Comparing the mean values of EngChoice (10.51) and CoDVA_real (12.46), we observe that the difference between them (1.95) is two-and-a-half smaller than the difference between the median values (4.97).

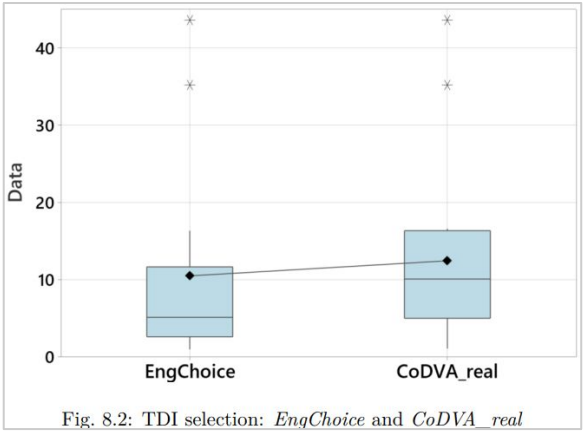


Fig. 8.2: TDI selection: *EngChoice* and *CoDVA_real*

CoDVA: Evaluation

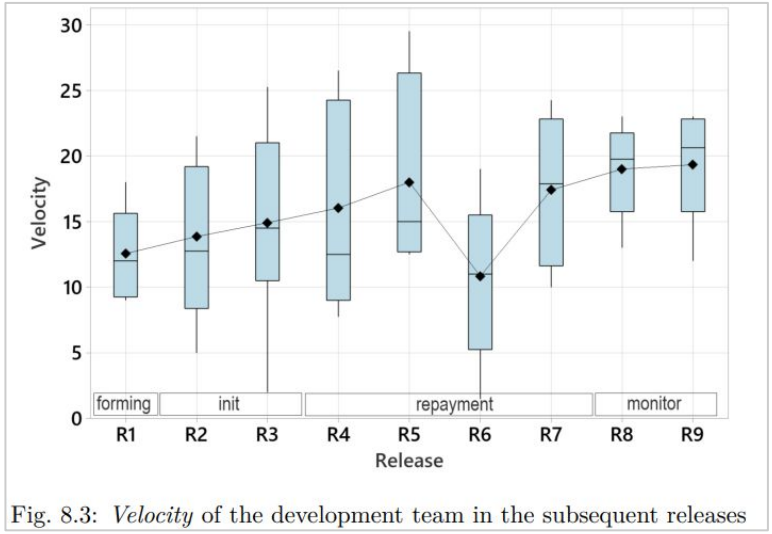


Fig. 8.3: *Velocity* of the development team in the subsequent releases

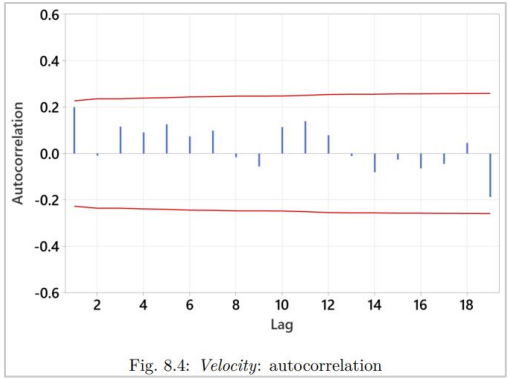


Fig. 8.4: *Velocity*: autocorrelation

Table 8.6: *Velocity*: autocorrelation table and statistics

Lag	ACF*	Z-Statistic	LBQ**	χ^{2***}
1	0.199817	1.75	3.20	3.841
2	−0.009235	−0.08	3.20	5.991
3	0.115849	0.98	4.31	7.815
4	0.090812	0.76	4.99	9.488
5	0.125637	1.04	6.33	11.070
6	0.073873	0.60	6.79	12.592
7	0.099085	0.81	7.65	14.067
8	−0.016408	−0.13	7.67	15.507
9	−0.055724	−0.45	7.95	16.919
10	0.114105	0.92	9.13	18.307
11	0.139384	1.11	10.92	19.675
12	0.078850	0.62	11.50	21.026
13	−0.010821	−0.08	11.51	22.362
14	−0.080596	−0.63	12.14	23.685
15	−0.026288	−0.20	12.21	24.996
16	−0.064063	−0.50	12.62	26.296
17	−0.044918	−0.35	12.82	27.587
18	0.045897	0.35	13.04	28.869
19	−0.186938	−1.44	16.71	30.144

* ACF: Autocorrelation Function
** LBQ: Ljung–Box statistic
*** Chi-square right-tail probability ($\geq \chi^2$) for significance level $\alpha = 0.05$.

Observations

- Velocity gradually growing, some turbulences in the time of changes
  Interpretation: some changes (process/product) need to stabilize
- Velocity data does not show the signs of autocorrelation.

CoDVA: Evaluation

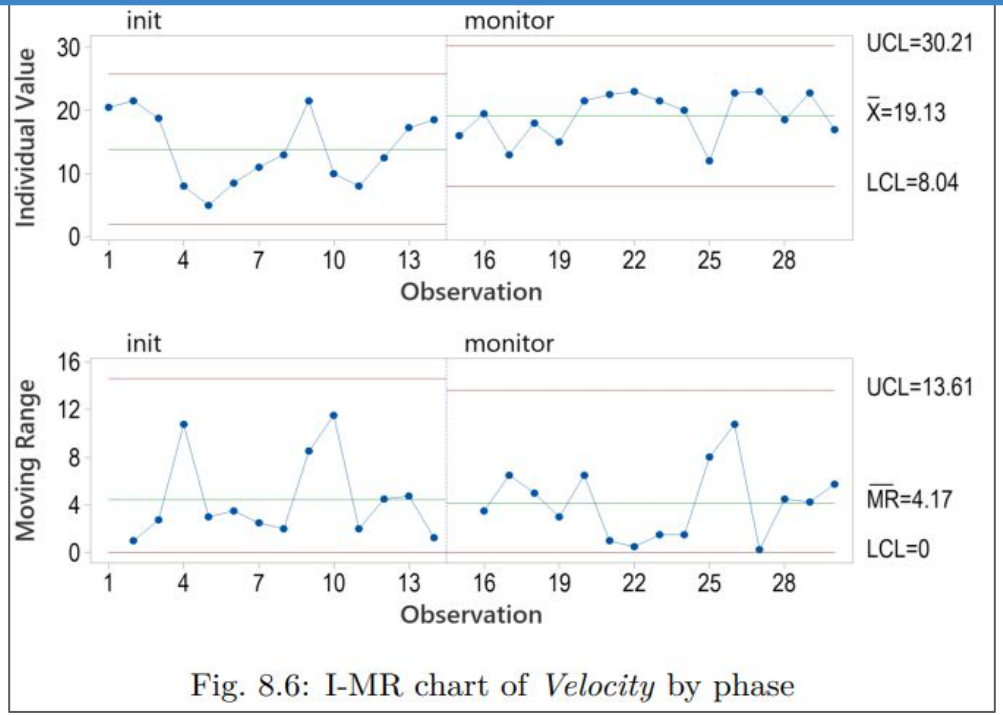


Fig. 8.6: I-MR chart of *Velocity* by phase

Observations

- Team able to deliver more (shifting the mean), 39% increase (13.8→19.3)
- Team slightly more stable sprint-to-sprint (reducing variance) - negligible

CoDVA: Evaluation

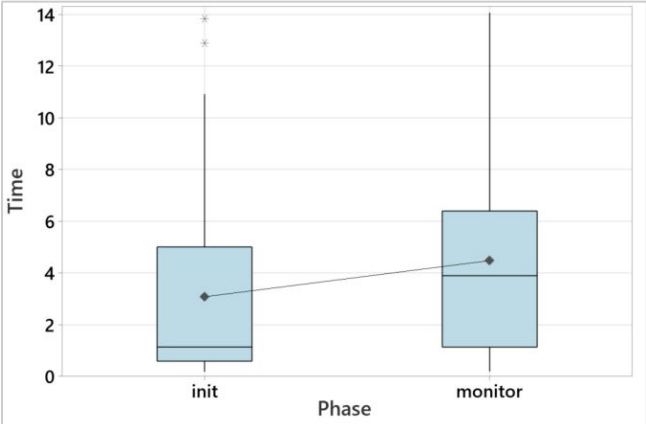


Fig. 8.7: Boxplot of *DefectFixTime* in phases: *init*, *monitor*

Table 8.7: *DefectFixTime* in phases: descriptive statistics

Phase	Mean	StDev	Q1	Median	Q3	IQR
init	3.083	3.570	0.593	1.144	5.006	4.413
monitor	4.480	3.406	1.136	3.898	6.400	5.263

Observations

- **More time needed to fix a single issue???**
- The median from an init phase is actually lower than the first quartile (Q1) for monitor phase, with unproportionally narrow both lower quartiles (Q1 and Q2)

Interpretations

- Easy-to-fix issues discovered and addressed early?

CoDVA: Evaluation

"Effort required to work on bugs increased because now we have regular meetings where we look for bugs and we find them more often. I think that we also became more strict when we're testing our PRs. So despite that effort increased, I think it's a good change."

"(...) looking at the recently added features they are covered by automated tests. Adding changes to these areas is easier than it was at the beginning"

"(...) I think that all of us are focused on improvement of the code's quality and readability. Some of the needed changes have been made, but still we need to do some more in the future."

Interpretation

- Increased focus on quality / craftsmanship
- Defect fixes embrace more complex refactorings

CoDVA: Evaluation

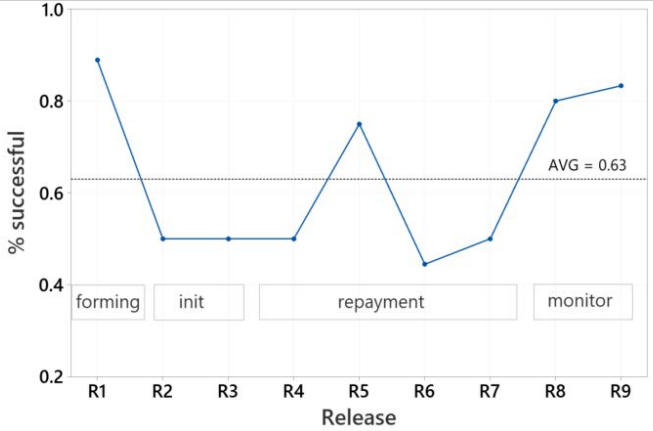


Fig. 8.8: *DeliveryPredictability*: the ratio of fully successful sprints in the subsequent releases

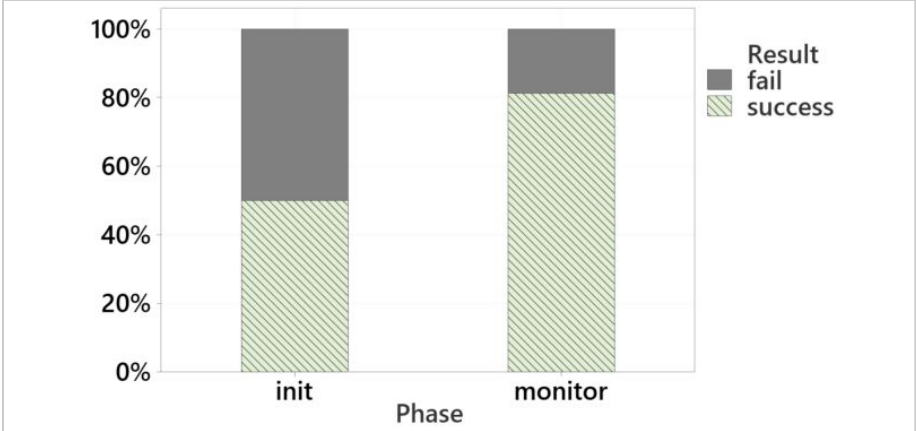


Fig. 8.9: *DeliveryPredictability* (proportions of fully successful sprints) in phases: *init* and *monitor*

Observation

- **Team more predictable** (by 60%)

CoDVA: Evaluation

Table 8.8: *Releasability*: stabilization cycles required to release a new version of the product

Stabilization Cycles per Product Release									
Phase	form	init		repayment				monitor	
Release	R1	R2	R3	R4	R5	R6	R7	R8	R9
Test runs	6	2	2	2	1	1	2	1	1
Effort [%][a]	600	100		75				50	

[a] Relative effort spent on product release stabilization against the *init* phase.

Observation

- Less regression cycles required to release, i.e. effort on product release stabilization decreased by 50%

Interpretation

- Faster time-to-market
- Lower cost, better predictability & quality

CoDVA: Evaluation

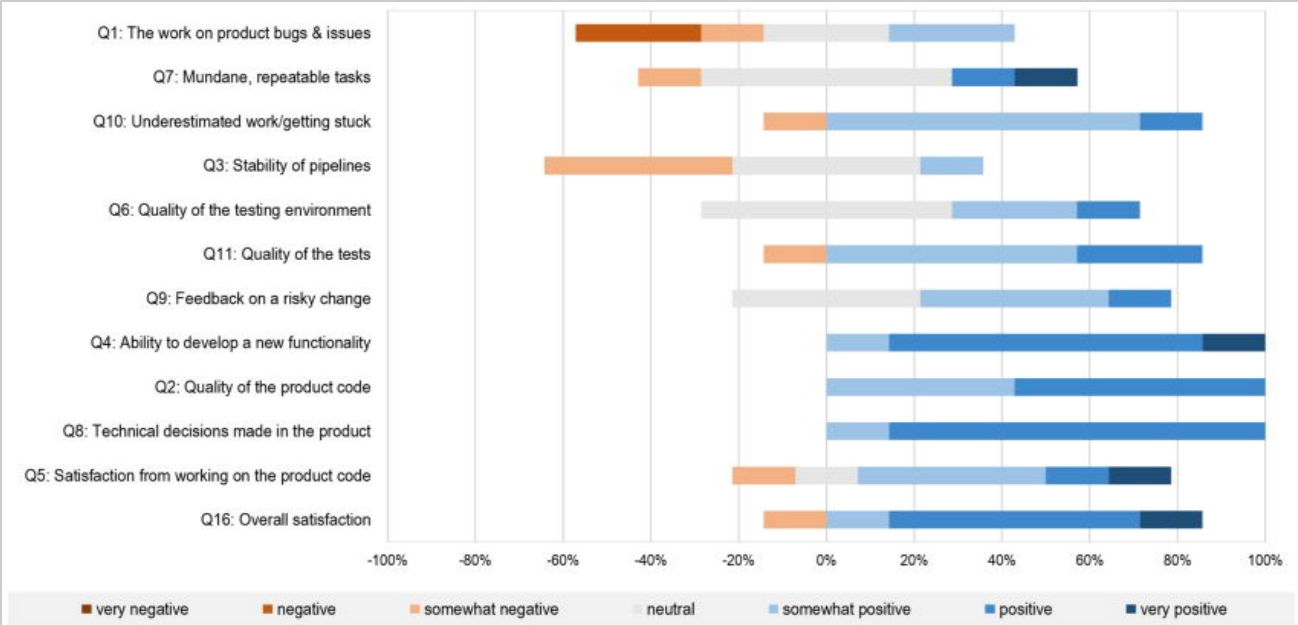


Fig. 8.10: Developer perspective on the direction of changes (trends) during assignment to the product

Observations

- **Leaned towards positive answers**, summarized by overall satisfaction
- Median neutral at least
- Favourable direction of changes: technical decisions, ability to develop new functionality, quality of the product code
- Interquartile range starting w/ negative: time spent on the work on bugs/issues

Summary

CoDVA-based TDM: Summary

- Perceived **business value** from TD refactorings raised by 27%
- **Engineering velocity** increased by 39%
- **Delivery predictability** improved by 60%
- Effort spent on **product stabilization** decreased by 50%
- In 86% of the cases overall **developer satisfaction** grew

The majority of the developers perceive the applied TDM strategy beneficial across

- Technical decisions made
- Ability to develop new functionality
- The experience while working with the product code

CoDVA: Implications for Researchers

- CoDVA-based TD management is a new methodology that has yet to gain wider adoption
- To determine the efficacy of TD management approaches, it is important to focus on the consequences of TD presence (**developer morale** and **productivity**)
- Potential ambiguity of TD repayment consequences
- TDM activities not only impact the product but also affect the behavior of the developers conducting them

CoDVA: Implications for Practitioners

- **Engineering view** (TD identification) **balanced with the business view** (TD prioritization)
- **Aligning the TDM process with Agile Scrum** increases the likelihood of the engineering team driving it
- CoDVA enhances **architectural cohesion** by strategically investing in the product areas that support new feature development
- The positive outcomes of TD repayment may not be immediately evident, but these activities can cause **temporary deterioration** in process measurements during periods of intensive changes
- The **development team is more motivated** to engage in TD management activities when they perceive the benefits of the approach

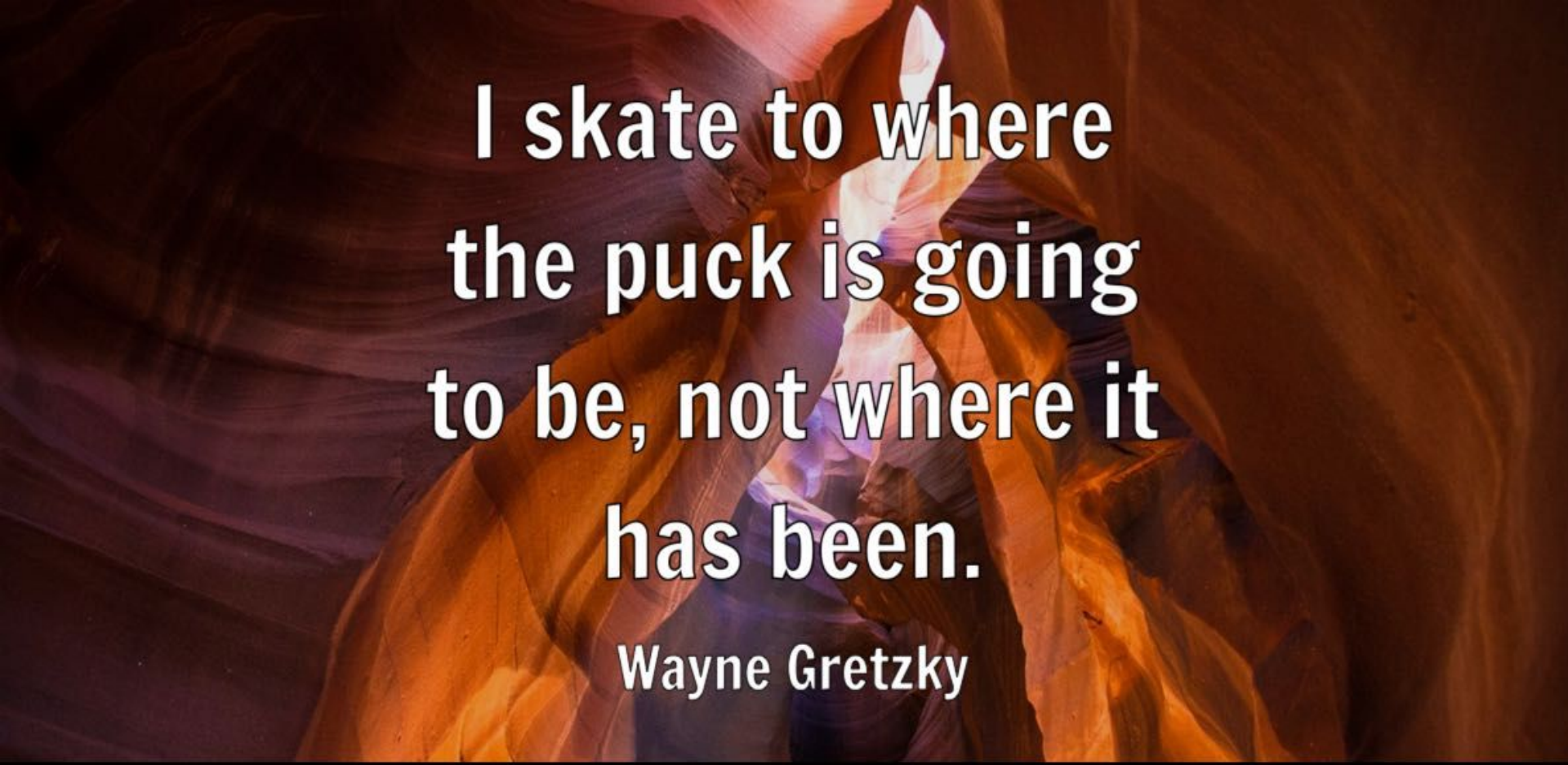
CoDVA: Generalization→Prerequisites

- Business
  - Systematic Business Process (tracked/managed)
  - Long-term prioritized feature roadmap
- Engineering
  - Product
    - Version-controlled SW artifacts
    - Broadened product view is desired
  - Process
    - Prioritization: one SW development queue
    - Communication: SW development team, PO, PdM
    - Adaptive SW development process
    - Engineering feedback elicitation and aggregation (Wisdom of Crowds)
    - The approach works well irrespectively of the setting (remote or on-site work)
- CoDVA mapping
  - Any worksheet could do

CoDVA: Generalization→Threats/Mitigation

- Threats
  - Case study related: the generalization power is limited by the selected case
  - Context-dependence: the purpose determines the evaluation measures for business prioritization

- Mitigations
  - The length of the case study reinforces the significance of the obtained results
  - Methodological triangulation: collating the data from the case studies and the survey
  - More generic approach to profitability (business value generation)
  - Used widely adopted techniques and standards: Scrum, TDM activities, ISO quality models, WoC


I skate to where the puck is going to be, not where it has been.
Wayne Gretzky
BrainyQuote®

Future Work

Future Work - ideas

- TD management tipping point
- Volatility of the desired state of the product
- Sociological aspects of TD management

Thank you

Q & A

Backup slides

Bibliography

Bibliography 1/6

[1] Zahra Shakeri Hossein Abad and Guenther Ruhe. Using Real Options to Manage Technical Debt in Requirements Engineering. In 2015 IEEE 23rd International Requirements Engineering Conference (RE), pages 230–235, 2015. doi: 10.1109/RE.2015.7320428.
[2] Philip Achimugu, Ali Selamat, Roliana Ibrahim, and Mohd Naz'ri Mahrin. A Systematic Literature Review of Software Requirements Prioritization Research. Information and Software Technology, 56(6):568–585, 2014. doi: 10.1016/j.infsof.2014.02.001.
[3] Aha! Labs. What is Product Value?, 2023. URL https://www.aha.io/roadmapping/guide/product-strategy/what-is-product-value. retrieved on Apr. 26, 2023.
[4] Shirin Akbarinasaji, Ayse Basar Bener, and Atakan Erdem. Measuring the Principal of Defect Debt. In 5th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), page 1–7, New York, NY, USA, 2016. ACM. ISBN 9781450341653. doi: 10.1145/2896995.2896999.
[5] Danyllo Albuquerque, Everton Tavares Guimaraes, Graziela Simone Tonin, Mirko Barbosa Perkusich, Hyggo Almeida, and Angelo Perkusich. Perceptions of Technical Debt and its Management Activities—A Survey of Software Practitioners. In 36th Brazilian Symposium on Software Engineering, pages 220–229, 2022. doi: 10.1145/3555228.3555237.
[6] Danyllo Albuquerque, Everton Guimarães, Graziela Tonin, Pilar Rodríguezs, Mirko Perkusich, Hyggo Almeida, Angelo Perkusich, and Ferdinandy Chagas. Managing Technical Debt Using Intelligent Techniques—A Systematic Mapping Study. IEEE Transactions on Software Engineering, 49(4):2202–2220, 2023. doi: 10.1109/TSE.2022.3214764.
[7] Reem Alfayez, Wesam Alwehaibi, Robert Winn, Elaine Venson, and Barry Boehm. A Systematic Literature Review of Technical Debt Prioritization. In 2020 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 1–10, New York, NY, USA, 2020. ACM. doi: 10.1145/3387906.3388630.
[8] I. Elaine Allen and Christopher A. Seaman. Likert Scales and Data Analyses.Quality Progress, 40(7):64–65, 2007.
[9] Nicolli S.R. Alves, Thiago S. Mendes, Manoel G. de Mendonça, Rodrigo O. Spínola, Forrest Shull, and Carolyn Seaman. Identification and Management of Technical Debt: A Systematic Mapping Study. Information and Software Technology, 70:100–121, 2016. doi: 10.1016/j.infsof.2015.10.008.
[10] Theodoros Amanatidis. A Study on the Evolution of Software Quality and Technical Debt in Open Source Applications. PhD thesis, University of Macedonia, Thessaloniki, Greece, 2020. URL http://hdl.handle.net/10442/hedi/47758.
[11] Areti Ampatzoglou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Paris Avgeriou. The Financial Aspect of Managing Technical Debt: A Systematic Literature Review. Information and Software Technology, 64:52–73, 2015. doi: 10.1016/j.infsof.2015.04.001.
[12] Paul Anderson, Łucja Kot, Neil Gilmore, and David Vitek. SARIF-Enabled Tooling to Encourage Gradual Technical Debt Reduction. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 71–72. IEEE, 2019. doi: 10.1109/TechDebt.2019.00024.
[13] Arno Anzenbacher. Wprowadzenie do filozofii (Introduction to Philosophy). WAM, Kraków, Poland, 4th edition, 2018.
[14] Elvira-Maria Arvanitou, Pigi Argyriadou, Georgia Koutsou, Apostolos Ampatzoglou, and Alexander Chatzigeorgiou. Quantifying TD Interest: Are We Getting Closer, or Not Even That? In 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 274–281. IEEE, 2022. doi: 10.1109/SEAA56994.2022.00050.
[15] Paris Avgeriou, Philippe Kruchten, Ipek Ozkaya, and Carolyn Seaman. Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162). Dagstuhl Reports, 6(4), 2016. doi: 10.4230/DagRep.6.4.110.
[16] Umberto Azadi, Francesca Arcelli Fontana, and Davide Taibi. Architectural Smells Detected by Tools: A Catalogue Proposal. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 88–97. IEEE, 2019. doi: 10.1109/TechDebt.2019.00027.
[17] Simon Baars and Sander Meester. CodeArena: Inspecting and Improving Code Quality Metrics in Java Using Minecraft. In 2019 International Conference on Technical Debt (Tool Demos). IEEE, 2019. doi:10.1109/TechDebt.2019.00023.
[18] Kenneth Baclawski and Bipin Indurkhya. The Notion of Inheritance in Object-Oriented Programming. Communications of the ACM, 37(9):118–119, 1994. doi: 10.1145/182987.184080.
[19] Christoph Becker, Fabian Fagerholm, Rahul Mohanani, and Alexander Chatzigeorgiou. Temporal Discounting in Technical Debt: How Do Software Practitioners Discount the Future? In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 23–32. IEEE, 2019. doi:10.1109/TechDebt.2019.00011.
[20] Woubshet Nema Behutiye, Pilar Rodríguez, Markku Oivo, and Ayşe Tosun. Analyzing the Concept of Technical Debt in the Context of Agile Software Development: A Systematic Literature Review. Information and Software Technology, 82:139–158, 2017. doi: 10.1016/j.infsof.2016.10.004.
[21] Mrwan Benldris, Dale Dzielski, and Hany Ammar. Investigate, Identify and Estimate the Technical Debt: A Systematic Mapping Study. International Journal of Software Engineering & Applications (IJSEA), 9(5), 2018. doi:10.5121/ijsea.2018.9501.
[22] Terese Besker, Antonio Martini, and Jan Bosch. A Systematic Literature Review and a Unified Model of ATD. In 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 189–197. IEEE, 2016. doi: 10.1109/SEAA.2016.42.
[23] Terese Besker, Antonio Martini, and Jan Bosch. Managing Architectural Technical Debt: A Unified Model and Systematic Literature Review. Journal of Systems and Software, 135:1–16, 2018. doi: 10.1016/j.jss.2017.09.025.
[24] Terese Besker, Antonio Martini, Rumesh Edirisooriya Lokuge, Kelly Blincoe, and Jan Bosch. Embracing Technical Debt, From a Startup Company Perspective. In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), pages 415–425. IEEE, 2018. doi: 10.1109/ICSME.2018.00051.
[25] Terese Besker, Antonio Martini, and Jan Bosch. Technical Debt Triage in Backlog Management. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 13–22. IEEE, 2019. doi:10.1109/TechDebt.2019.00010.
[26] Terese Besker, Hadi Ghanbari, Antonio Martini, and Jan Bosch. The Influence of Technical Debt on Software Developer Morale. Journal of Systems and Software, 167:110586, 2020. doi: 10.1016/j.jss.2020.110586.
[27] Kathrin Bogner and Uta Landrock. Response Biases in Standardised Surveys. GESIS Survey Guidelines. GESIS – Leibniz Institute for the Social Sciences, Mannheim, Germany, 2nd edition, 2016. doi: 10.15465/gesis-sg_en_016.
[28] Klara Borowa, Andrzej Zalewski, and Szymon Kijas. The Influence of Cognitive Biases on Architectural Technical Debt. In 2021 IEEE 18th International Conference on Software Architecture (ICSA), pages 115–125. IEEE, 2021. doi: 10.1109/ICSA51549.2021.00019.
[29] Klara Borowa, Andrzej Zalewski, and Adam Saczko. Living With Technical Debt—A Perspective From the Video Game Industry. IEEE Software, 38(06):65–70, 2021. doi: 10.1109/MS.2021.3103249.
[30] Richard Brenner. Balancing Resources and Load: Eleven Nontechnical Phenomena That Contribute to Formation or Persistence of Technical Debt. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 38–47. IEEE, 2019. doi: 10.1109/TechDebt.2019.00013.
[31] Mariya Breyter. Agile Product and Project Management: A Step-by-Step Guide to Building the Right Products Right. Apress, Berkeley, CA, USA, 1st edition, 2022. doi: 10.1007/978-1-4842-8200-7.
[32] Quentin Brook. Six Sigma and Minitab: A Complete Toolbox Guide for All Six Sigma Practitioners. QSB Consulting, 2nd edition, 2006.
[33] Frank Buschmann. To Pay or Not to Pay Technical Debt. IEEE Software, 28(6):29–31, 2011. doi: 10.1109/MS.2011.150.
[34] Yuanfang Cai and Rick Kazman. DV8: Automated Architecture Analysis Tool Suites. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 53–54. IEEE, 2019. doi: 10.1109/TechDebt.2019.00015.
[35] Marcus Ciolkowski, Valentina Lenarduzzi, and Antonio Martini. 10 Years of Technical Debt Research and Practice: Past, Present, and Future. IEEE Software, 38(6):24–29, 2021. doi: 10.1109/MS.2021.3105625.
[36] Zadia Codabux and Byron Williams. Managing Technical Debt: An Industrial Case Study. In 4th International Workshop on Managing Technical Debt (MTD), pages 8–15. IEEE, 2013. doi: 10.1109/MTD.2013.6608672.
[37] Zadia Codabux and Byron J. Williams. Technical Debt Prioritization Using Predictive Analytics. In 38th International Conference on Software Engineering Companion, pages 704–706, 2016. doi: 10.1145/2889160.2892643.
[38] Mike Cohn. Agile Estimating and Planning. Pearson Education, 1st edition, 2005.
[39] Mike Cohn. Succeeding With Agile: Software Development Using Scrum. Pearson Education, 1st edition, 2010.
[40] Hugh Coolican. Research Methods and Statistics in Psychology. Routledge, London, UK, 7th edition, 2018. doi: 10.4324/9781315201009.
[41] Alex Felipe Ferreira Costa, Anna Beatriz dos Santos Marques, Ismayle Sousa Santos, and Rossana Maria de Castro Andrade. Towards a Process to Manage Usability Technical Debts. In 2022 36th Brazilian Symposium on Software Engineering (SBES), pages 241–246, New York, NY, USA, 2022. ACM. doi:10.1145/3555228.3555266.
[42] Ward Cunningham. The WyCash Portfolio Management System. ACM SIGPLAN OOPS Messenger, 4(2):29–30, 1992. doi: 10.1145/157710.157715.
[43] Luiz Carlos da Fonseca Lage, Marcos Kalinowski, Daniela Trevisan, and Rodrigo Oliveira Spínola. Usability Technical Debt in Software Projects: A Multi-Case Study. In 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pages 1–6. IEEE, 2019. doi: 10.1109/ESEM.2019.8870180.
[44] Filipe Tabosa da Silva, Ewertton Oliveira Silva de Souza, Rodrigo Rebouças de Almeida, and Wylliams Barbosa Santos. Business-Driven Technical Debt Prioritization: A Replication Study. In 2022 17th Iberian Conference on Information Systems and Technologies (CISTI), pages 1–6. IEEE, 2022. doi:10.23919/CISTI54924.2022.9820390.

Bibliography 2/6

[45] José Diego Saraiva da Silva, José Gameleira Neto, Uirá Kulesza, Guilherme Freitas, Rodrigo Rebouças, and Roberta Coelho. Exploring Technical Debt Tools: A Systematic Mapping Study. In Enterprise Information Systems: 23rd International Conference, ICEIS 2021, Virtual Event, April 26–28, 2021, Revised Selected Papers, pages 280–303. Springer, 2022. doi: 10.1007/978-3-031-08965-7_14.
[46] Rodrigo Rebouças de Almeida, Uirá Kulesza, Christoph Treude, Aliandro Higino Guedes Lima, and D'angellys Cavalcanti Feitosa. Aligning Technical Debt Prioritization With Business Objectives: A Multiple-Case Study. In 2018 IEEE International Conference on Software Maintenance and Evolution (ICSME), pages 655–664. IEEE, 2018. doi: 10.13140/RG.2.2.16106.21441.
[47] Rodrigo Rebouças de Almeida, Christoph Treude, and Uirá Kulesza. Tracy: A Business-Driven Technical Debt Prioritization Framework. In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), pages 181–185. IEEE, 2019. doi: 10.1109/ICSME.2019.00028.
[48] Rodrigo Rebouças de Almeida, Rafael do Nascimento Ribeiro, Christoph Treude, and Uirá Kulesza. Business-Driven Technical Debt Prioritization: An Industrial Case Study. In 2021 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 74–83. IEEE, 2021. doi:10.1109/TechDebt52882.2021.00017.
[49] Brian de Haaff. Business Roadmaps vs. Product Roadmaps, 2020. URL https://www.aha.io/blog/business-roadmaps-vs-product-roadmaps, retrieved on Jul. 25, 2023.
[50] Bruno Santos de Lima, Rogerio Eduardo Garcia, and Danilo Medeiros Eler. Toward Prioritization of Self-Admitted Technical Debt: An Approach to Support Decision to Payment. Software Quality Journal, 30(3):729–755, 2022. doi: 10.1007/s11219-021-09578-7.
[51] Saulo Soares de Toledo. Managing Architectural Technical Debt in Microservices. PhD thesis, University of Oslo, Oslo, Norway, 2022. URL http://urn.nb.no/URN:NBN:no-97648.
[52] Saulo Soares de Toledo, Antonio Martini, Agata Przybyszewska, and Dag I.K. Sjøberg. Architectural Technical Debt in Microservices: A Case Study in a Large Company. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 78–87. IEEE, 2019. doi: 10.1109/TechDebt.2019.00026.
[53] Marco di Biase, Ayushi Rastogi, Magiel Bruntink, and Arie van Deursen. The Delta Maintainability Model: Measuring Maintainability of Fine-Grained Code Changes. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 113–122. IEEE, 2019. doi:10.1109/TechDebt.2019.00030.
[54] Danny Dig. Lessons From the Exponential Growth of Refactoring Research in the Last Decade. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 62–63. IEEE, 2019. doi:10.1109/TechDebt.2019.00020.
[55] George Digkas, Alexander Chatzigeorgiou, Apostolos Ampatzoglou, and Paris Avgeriou. Can Clean New Code Reduce Technical Debt Density? IEEE Transactions on Software Engineering, 48(5):1705–1721, 2022. doi: 10.1109/TSE.2020.3032557.
[56] Keith Eades. The New Solution Selling. McGraw-Hill New York, 1st edition, 2003.
[57] Floris M.A. Erich, Chintan Amrit, and Maya Daneva. A Qualitative Study of DevOps Usage in Practice. Journal of Software: Evolution and Process, 29(6):e1885, 2017. doi: 10.1002/smr.1885.
[58] Morgan Ericsson and Anna Wingkvist. TDMentions: A Dataset of Technical Debt Mentions in Online Posts. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 123–124. IEEE, 2019. doi: 10.1109/TechDebt.2019.00031.
[59] Neil A Ernst, Stephany Bellomo, Ipek Ozkaya, Robert L. Nord, and Ian Gorton. Measure It? Manage It? Ignore It? Software Practitioners and Technical Debt. In 2015 10th Joint Meeting on Foundations of Software Engineering, pages 50–60, 2015. doi: 10.1145/2786805.2786848.
[60] Davide Falessi, Michele A. Shaw, Forrest Shull, Kathleen Mullen, and Mark Stein Keymind. Practical Considerations, Challenges, and Requirements of Tool-Support for Managing Technical Debt. In 4th International Workshop on Managing Technical Debt (MTD), pages 16–19, 2013. doi: 10.1109/MTD.2013.6608673.
[61] Carlos Fernández-Sánchez, Juan Garbajosa, Carlos Vidal, and Agustín Yagüe. An Analysis of Techniques and Methods for Technical Debt Management: A Reflection From the Architecture Perspective. In 2015 IEEE/ACM 2nd International Workshop on Software Architecture and Metrics, pages 22–28, 2015. doi: 10.1109/SAM.2015.11.
[62] Carlos Fernández-Sánchez, Juan Garbajosa, and Agustín Yagüe. A Framework to Aid in Decision Making for Technical Debt Management. In 7th International Workshop on Managing Technical Debt (MTD), pages 69–76, 2015. doi: 10.1109/MTD.2015.7332628.
[63] Carlos Fernández-Sánchez, Juan Garbajosa, Agustín Yagüe, and Jennifer Perez. Identification and Analysis of the Elements Required to Manage Technical Debt by Means of a Systematic Mapping Study. Journal of Systems and Software, 124:22–38, 2017. doi: 10.1016/j.jss.2016.10.018.
[64] Martin Fowler. CodeSmell, 2006. URL https://martinfowler.com/bliki/CodeSmell.html. retrieved on Apr. 24, 2023.
[65] Martin Fowler. Technical Debt Quadrant, 2009. URL http://www.martinfowler.com/bliki/TechnicalDebtQuadrant.html. retrieved on Apr. 24, 2023.
[66] Sávio Freire. Organizing the State of Practice on Technical Debt Prevention, Monitoring, and Payment in Software Projects. PhD thesis, Federal University of Bahia, Salvador, Brazil, 2023. URL https://repositorio.ufba.br/handle/ri/37027.
[67] Vahid Garousi, Michael Felderer, and Mika V. Mäntylä. The Need for Multivocal Literature Reviews in Software Engineering: Complementing Systematic Literature Reviews With Grey Literature. In 20th International Conference on Evaluation and Assessment in Software Engineering, pages 1–6, 2016. doi: 10.1145/2915970.2916008.
[68] Felipe Gomes, Eder Pereira dos Santos, Sávio Freire, Manoel Mendonça, Thiago Souto Mendes, and Rodrigo O. Spínola. Investigating the Point of View of Project Management Practitioners on Technical Debt—A Preliminary Study on Stack Exchange. In 2022 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 31–40, 2022. doi: 10.1145/3524843.3528095.
[69] Daniel Graziotin, Fabian Fagerholm, Xiaofeng Wang, and Pekka Abrahamsson. On the Unhappiness of Software Developers. In 21st International Conference on Evaluation and Assessment in Software Engineering (EASE), pages 324–333, 2017. doi: 10.1145/3084226.3084242.
[70] Valentin Guerlesquin. How (Not) to Remove Technical Debt in Testing Environments. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 1–2. IEEE, 2019. doi: 10.1109/TechDebt.2019.00008.
[71] Yuepu Guo. Measuring and Monitoring Technical Debt. PhD thesis, University of Maryland, College Park, MD, USA, 2016.
[72] Yuepu Guo, Rodrigo Oliveira Spínola, and Carolyn Seaman. Exploring the Costs of Technical Debt Management—A Case Study. Empirical Software Engineering, 21:159–182, 2016. doi: 10.1007/s10664-014-9351-7.
[73] Rajeev Kumar Gupta, Prabhulinga Manikreddy, Sandesh Naik, and Kavi Arya. Pragmatic Approach for Managing Technical Debt in Legacy Software Project. In 9th India Software Engineering Conference, pages 170–176, 2016. doi: 10.1145/2856636.2856655.
[74] Roman Haas, Rainer Niedermayr, and Elmar Juergens. Teamscale: Tackle Technical Debt and Control the Quality of Your Software. In 2019
IEEE/ACM International Conference on Technical Debt (TechDebt), pages 55–56. IEEE, 2019. doi: 10.1109/TechDebt.2019.00016.
[75] Geir Kjetil Hanssen, Gunnar Brataas, and Antonio Martini. Identifying Scalability Debt in Open Systems. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 48–52. IEEE, 2019. doi: 10.1109/TechDebt.2019.00014.
[76] Jim Highsmith and Alistair Cockburn. Agile Software Development: The Business of Innovation. Computer, 34(9):120–127, 2001. doi: 10.1109/2.947100.
[77] Johannes Holvitie, Ville Leppänen, and Sami Hyrynsalmi. Technical Debt and the Effect of Agile Software Development Practices on It—An Industry Practitioner Survey. In 2014 Sixth International Workshop on Managing Technical Debt, pages 35–42. IEEE, 2014. doi: 10.1016/j.infsof.2017.11.015.
[78] Rob Hyndman, Anne B. Koehler, J. Keith Ord, and Ralph D. Snyder. Forecasting With Exponential Smoothing: The State Space Approach. Springer, 1st edition, 2008.
[79] IEEE. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. IEEE Std 610, pages 1–217, 1991. doi: 10.1109/IEEESTD.1991.106963.
[80] Bipin Indurkhya. Some Philosophical Observations on the Nature of Software
and Their Implications for Requirement Engineering, 2017. URL https://www.academia.edu/7817075. retrieved on May 16, 2023.

Bibliography 3/6

[81] ISO/IEC. 9126-1:2001, Software Engineering—Product Quality—Part 1:Quality Model. System and Software Quality Models Geneva, 2001.
[82] ISO/IEC. 25010:2011(E) Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE)—System and Software Quality Models. System and Software Quality Models Geneva, 2011.
[83] ISO/IEC. 25052-1:2022 Systems and Software Engineering—Systems and Software Quality Requirements and Evaluation (SQuaRE): Cloud Services—Part 1: Quality Model. System and Software Quality Models Geneva, 2022.
[84] ISO/IEC/IEEE. 24765:2017(E) Systems and Software Engineering—Vocabulary. IEEE, 2nd edition, 2017. doi: 10.1109/IEEESTD.2017.8016712.
[85] ISO/IEC/IEEE. 29148:2018(E) Systems and Software Engineering—Life Cycle Processes—Requirements Engineering. IEEE, 2nd edition, 2018. doi:10.1109/IEEESTD.2018.8559686.
[86] Clemente Izurieta and Mary Prouty. Leveraging SecDevOps to Tackle the Technical Debt Associated With Cybersecurity Attack Tactics. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 33–37. IEEE, 2019. doi: 10.1109/TechDebt.2019.00012.
[87] Clemente Izurieta, Ipek Ozkaya, Carolyn Seaman, Philippe Kruchten, Robert Nord, Will Snipes, and Paris Avgeriou. Perspectives on Managing Technical Debt: A Transition Point and Roadmap From Dagstuhl. In Lichter Horst, Fögen Konrad, Sunetnanta Thanwadee, Anwar Toni, Yamashita Aiko, Moonen Leon, Mens Tom, Tahir Amjed, and Sureka Ashish, editors, 2016
1st International Workshop on Technical Debt Analytics (TDA), volume 1771 of CEUR Workshop Proceedings, pages 84–87. CEUR-WS.org, 2016.
URL https://ceur-ws.org/Vol-1771/paper15.pdf.
[88] Clemente Izurieta, Isaac Griffith, and Chris Huvaere. An Industry Perspective to Comparing the SQALE and Quamoco Software Quality Models. In 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pages 287–296. IEEE, 2017. doi:10.1109/ESEM.2017.42.
[89] Clemente Izurieta, Kali Kimball, David Rice, and Tessa Valentien. A Position Study to Investigate Technical Debt Associated With Security Weaknesses. In 2018 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 138–142. IEEE, 2018. doi: 10.1145/3194164.3194167.
[90] Ciera Jaspan and Collin Green. Defining, Measuring, and Managing Technical Debt. IEEE Software, 40(3):15–19, 2023. doi: 10.1109/MS.2023.3242137.
[91] Ciera Jaspan and Caitlin Sadowski. No Single Metric Captures Productivity. In Caitlin Sadowski and Thomas Zimmermann, editors, Rethinking Productivity in Software Engineering, pages 13–20. Springer Nature, 2019. doi: 10.1007/978-1-4842-4221-6_2.
[92] Ankur Joshi, Saket Kale, Satish Chandel, and D. Kumar Pal. Likert Scale: Explored and Explained. British Journal of Applied Science & Technology, 7(4):396, 2015. doi: 10.9734/BJAST/2015/14975.
[93] Benjamin Jowett. The Allegory of the Cave by Plato. P & L Publication Brea, 2010.
[94] Helvio Jeronimo Junior and Guilherme Horta Travassos. Consolidating a Common Perspective on Technical Debt and Its Management Through a Tertiary Study. Information and Software Technology, 149:106964, 2022. doi: 10.1016/j.infsof.2022.106964.
[95] Steven H. Kan. Metrics and Models in Software Quality Engineering. Addison-Wesley Professional, 2nd edition, 2002.
[96] Andrej Katin, Valentina Lenarduzzi, Davide Taibi, and Vladimir Mandić. On the Technical Debt Prioritization and Cost Estimation With SonarQube Tool. In 18th International Conference on Industrial Systems (IS'20): Industrial Innovation in Digital Age, pages 302–309. Springer, 2022. doi: 10.1007/978-3-030-97947-8_40.
[97] Ilya Khomyakov, Zufar Makhmutov, Ruzilya Mirgalimova, and Alberto Sillitti. An Analysis of Automated Technical Debt Measurement. In Enterprise Information Systems: 21st International Conference, ICEIS 2019, Heraklion, Crete, Greece, May 3–5, 2019, Revised Selected Papers 21, pages 250–273. Springer, 2020. doi: 10.1007/978-3-030-40783-4_12.
[98] Gene Kim, Jez Humble, Patrick Debois, John Willis, and Nicole Forsgren.The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations. IT Revolution, 1st edition, 2021.
[99] Daniel Kindström. Towards a Service-Based Business Model—Key Aspects for Future Competitive Advantage. European Management Journal, 28(6): 479–490, 2010. doi: https://doi.org/10.1016/j.emj.2010.07.002.
[100] Barbara Kitchenham, Stuart Charters, et al. Guidelines for Performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE 2007-01, Keele University and Durham University Joint Report, 2007.
[101] Howard Kleinwaks, Ann Batchelor, and Thomas H. Bradley. Technical Debt in Systems Engineering—A Systematic Literature Review. Systems Engineering (Early View), pages 1–13, 2023. doi: 10.1002/sys.21681.
[102] Paweł Klimczyk and Lech Madeyski. Technical Debt Aware Estimations in Software Engineering: A Systematic Mapping Study. e-Informatica Software Engineering Journal, 14(1):61–76, 2020. doi: 10.37190/e-Inf200102.
[103] Boris Kontsevoi, Elizabeth Soroka, and Sergei Terekhov. TETRA, as a Set of Techniques and Tools for Calculating Technical Debt Principal and Interest. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 64–65. IEEE, 2019. doi: 10.1109/TechDebt.2019.00021.
[104] Philippe Kruchten. Contextualizing Agile Software Development. Journal of Software Maintenance and Evolution: Research and Practice, 25, 04 2013. doi: 10.1002/smr.572.
[105] Philippe Kruchten. The End of Agile as We Know It. In 2019 IEEE/ACM International Conference on Software and System Processes (ICSSP), pages 104–104. IEEE, 2019. doi: 10.1109/ICSSP.2019.00033.
[106] Philippe Kruchten, Robert L. Nord, and Ipek Ozkaya. Technical Debt: From Metaphor to Theory and Practice. IEEE Software, 29(6):18–21, 2012. doi: 10.1109/MS.2012.167.
[107] Philippe Kruchten, Robert Nord, and Ipek Ozkaya. Managing Technical Debt: Reducing Friction in Software Development. Addison-Wesley Professional, 1st edition, 2019.
[108] Eetu Kupiainen, Mika V. Mäntylä, and Juha Itkonen. Using Metrics in Agile and Lean Software Development—A Systematic Literature Review of Industrial Studies. Information and Software Technology, 62:143–163, 2015. doi: 10.1016/j.infsof.2015.02.005.
[109] George Lakoff and Mark Johnson. Metaphors We Live By. University of Chicago Press, 1st edition, 2003.
[110] Craig Larman and Bas Vodde. Large-Scale Scrum: More With LeSS. Addison-Wesley Professional, 1st edition, 2016.
[111] Jason Lefever, Yuanfang Cai, Humberto Cervantes, Rick Kazman, and Hongzhou Fang. On the Lack of Consensus Among Technical Debt Detection Tools. In 2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pages 121–130, 2021.doi: 10.1109/ICSE-SEIP52600.2021.00021.
[112] Valentina Lenarduzzi, Nyyti Saarimaki, and Davide Taibi. On the Diffuseness of Code Technical Debt in Java Projects of the Apache Ecosystem. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 98–107. IEEE, 2019. doi: 10.1109/TechDebt.2019.00028.
[113] Valentina Lenarduzzi, Jeremy Daly, Antonio Martini, Sebastiano Panichella, and Damian Andrew Tamburri. Toward a Technical Debt Conceptualization for Serverless Computing. IEEE Software, 38(1):40–47, 2020. doi:10.1109/MS.2020.3030786.
[114] Valentina Lenarduzzi, Terese Besker, Davide Taibi, Antonio Martini, and Francesca Arcelli Fontana. A Systematic Literature Review on Technical Debt Prioritization: Strategies, Processes, Factors, and Tools. Journal of Systems and Software, 171:110827, 2021. doi: 10.1016/j.jss.2020.110827.
[115] Aversano Lerina and Laura Nardi. Investigating on the Impact of Software Clones on Technical Debt. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 108–112. IEEE, 2019. doi: 10.1109/TechDebt.2019.00029.
[116] Jean-Louis Letouzey. The SQALE Method for Evaluating Technical Debt. In 3rd International Workshop on Managing Technical Debt (MTD), pages 31–36. IEEE, 2012. doi: 10.1109/MTD.2012.6225997.
[117] Jean-Louis Letouzey. The SQALE Method for Managing Technical Debt, 2016. URL http://sqale.org/download. Definition Document, version 1.1, retrieved on Jun. 29, 2023.
[118] Zengyang Li, Paris Avgeriou, and Peng Liang. A Systematic Mapping Study on Technical Debt and Its Management. Journal of Systems and Software, 101:193–220, 2015. doi: 10.1016/j.jss.2014.12.027.
[119] Francesco Lomio, Zadia Codabux, Dale Birtch, Dale Hopkins, and Davide Taibi. On the Benefits of the Accelerate Metrics: An Industrial Survey at Vendasta. In 2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), pages 46–50. IEEE, 2022. doi: 10.1109/SANER53432.2022.00017.
[120] Jeremy Ludwig and Devin Cline. CBR Insight: Measure and Visualize Source Code Quality. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 57–58. IEEE, 2019. doi: 10.1109/TechDebt.2019.00017.
[121] Bjørn Arild Lunde and Ricardo Colomo-Palacios. Continuous Practices and Technical Debt: A Systematic Literature Review. In 2020 20th International Conference on Computational Science and Its Applications (ICCSA), pages 40–44. IEEE, 2020. doi: 10.1109/ICCSA50381.2020.00018.

Bibliography 4/6

[122] Lucy Ellen Lwakatare, Terhi Kilamo, Teemu Karvonen, Tanja Sauvola, Ville Heikkilä, Juha Itkonen, Pasi Kuvaja, Tommi Mikkonen, Markku Oivo, and Casper Lassenius. DevOps in Practice: A Multiple Case Study of Five Companies. Information and Software Technology, 114:217–230, 2019. doi: 10.1016/j.infsof.2019.06.010.
[123] Johan Magnusson, Carlos Juiz, Beatriz Gómez, and Belén Bermejo. Governing Technology Debt: Beyond Technical Debt. In 2018 International Conference on Technical Debt (TechDebt), pages 76–84. IEEE, 2018. doi: 10.1145/3194164.3194169.
[124] Netta Mäki, Esko Penttinen, and Tapani Rinta-Kahila. A Domino Effect: Interdependencies among Different Types of Technical Debt. In 56th Hawaii International Conference on System Sciences, page 5949, 2023. URL https://hdl.handle.net/10125/103356.
[125] Mika V. Mäntylä, Bram Adams, Foutse Khomh, Emelie Engström, and Kai Petersen. On Rapid Releases and Software Testing: A Case Study and a Semi-Systematic Literature Review. Empirical Software Engineering, 20:1384–1425, 2015. doi: 10.1007/s10664-014-9338-4.
[126] Diego Marcilio, Rodrigo Bonifácio, Eduardo Monteiro, Edna Canedo, Welder Luz, and Gustavo Pinto. Are Static Analysis Violations Really Fixed? A Closer Look at Realistic Usage of SonarQube. In 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC), pages 209–219. IEEE, 2019. doi: 10.1109/ICPC.2019.00040.
[127] Antonio Martini and Jan Bosch. The Danger of Architectural Technical Debt: Contagious Debt and Vicious Circles. In 2015 12th Working IEEE/IFIP Conference on Software Architecture, pages 1–10. IEEE, 2015. doi: 10.1109/WICSA.2015.31.
[128] Antonio Martini and Jan Bosch. Towards Prioritizing Architecture Technical Debt: Information Needs of Architects and Product Owners. In 2015 41st Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 422–429. IEEE, 2015. doi: 10.1109/SEAA.2015.78.
[129] Antonio Martini, Jan Bosch, and Michel Chaudron. Architecture Technical Debt: Understanding Causes and a Qualitative Model. In 2014 40th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 85–92, 2014. doi: 10.1109/SEAA.2014.65.
[130] Antonio Martini, Jan Bosch, and Michel Chaudron. Investigating Architectural Technical Debt Accumulation and Refactoring Over Time: A Multiple-Case Study. Information and Software Technology, 67:237–253, 2015. doi:10.1016/j.infsof.2015.07.005.
[131] Antonio Martini, Terese Besker, and Jan Bosch. The Introduction of Technical Debt Tracking in Large Companies. In 2016 23rd Asia-Pacific Software Engineering Conference (APSEC), pages 161–168, 2016. doi: 10.1109/APSEC.2016.032.
[132] Maria Mathioudaki, Dimitrios Tsoukalas, Miltiadis Siavvas, and Dionysios Kehagias. Comparing Univariate and Multivariate Time Series Models for Technical Debt Forecasting. In Computational Science and Its Applications – ICCSA 2022 Workshops, pages 62–78. Springer, 2022. doi: 10.1007/978-3-031-10542-5_5.
[133] Steve McConnell. Managing Technical Debt. White Paper, 2008. URL http://www.construx.com/uploadedfiles/resources/whitepapers/Managing%20Technical%20Debt.pdf. white paper, retrieved on Apr. 24, 2023.
[134] Ana Melo, Roberta Fagundes, Valentina Lenarduzzi, and Wylliams Santos.Identification and Measurement of Requirements Technical Debt in Software Development: A Systematic Literature Review. Journal of Systems and Software, 194:111483, 2022. doi: 10.1016/j.jss.2022.111483.
[135] Carlos Mera-Gómez, Rami Bahsoon, and Rajkumar Buyya. Elasticity Debt: A Debt-Aware Approach to Reason About Elasticity Decisions in the Cloud. In 2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC), pages 79–88, 2016. doi:10.1145/2996890.2996904.
[136] Sharan B. Merriam and Elizabeth J. Tisdell. Qualitative Research: A Guide to Design and Implementation. John Wiley & Sons, 4th edition, 2015.
[137] Microsoft. What is DevOps?, 2023. URL https://azure.microsoft.com/overview/what-is-devops. retrieved on Apr. 22, 2023.
[138] George A. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. In C. R. Evans and Robertson A. D. J., editors, Brain Physiology and Psychology, pages 175–202. University of California Press, Berkeley, 1966. doi: 10.1525/9780520318267-011.
[139] Vaishnavi Mohan and Lotfi Ben Othmane. SecDevOps: Is It a Marketing Buzzword?—Mapping Research on Security in DevOps. In 2016 11th International Conference on Availability, Reliability and Security (ARES), pages 542–547. IEEE, 2016. doi: 10.1109/ARES.2016.92.
[140] Douglas C. Montgomery, Cheryl L. Jennings, and Murat Kulahci. Introduction to Time Series Analysis and Forecasting. John Wiley & Sons, 2nd edition, 2015.
[141] Maria Murillo, Gustavo López, Rodrigo Spínola, Julio Guzman, Nicolli Rios, and Alexia Pacheco. Identification and Management of Technical Debt: A Systematic Mapping Study Update. Journal of Software Engineering Research and Development, 11, 08 2023. doi: 10.5753/jserd.2023.2671.
[142] Håvard Myrbakken and Ricardo Colomo-Palacios. DevSecOps: A Multivocal Literature Review. In Software Process Improvement and Capability Determination: 17th International Conference, SPICE 2017, Palma de Mallorca, Spain, October 4–5, 2017, Proceedings, pages 17–29. Springer, 2017.
[143] Robert L. Nord, Ipek Ozkaya, Philippe Kruchten, and Marco Gonzalez-Rojas. In Search of a Metric for Managing Architectural Technical Debt. In 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture, pages 91–100, 2012. doi:10.1109/WICSA-ECSA.212.17.
[144] Ariadi Nugroho, Joost Visser, and Tobias Kuipers. An Empirical Model of Technical Debt and Interest. In 2nd International Workshop on Managing Technical Debt (MTD), pages 1–8. IEEE, 2011. doi:10.1145/1985362.1985364.
[145] Luka Pavlič, Tilen Hliš, Marjan Heričko, and Tina Beranič. The Gap Between the Admitted and the Measured Technical Debt: An Empirical Study. Applied Sciences, 12(15):7482, 2022. doi: 10.3390/app12157482.
[146] Judith Perera, Ewan Tempero, Yu-Cheng Tu, and Kelly Blincoe. Quantifying Technical Debt: A Systematic Mapping Study and a Conceptual Model. arXiv preprint arXiv:2303.06535, 2023. doi: 10.48550/arXiv.2303.06535.
[147] Boris Pérez, Darío Correal, and Hernán Astudillo. A Proposed Model-Driven Approach to Manage Architectural Technical Debt Life Cycle. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 73–77. IEEE, 2019. doi: 10.1109/TechDebt.2019.00025.
[148] Ralph Peters and Andy Zaidman. Evaluating the Lifespan of Code Smells Using Software Repository Mining. In 2012 16th European Conference on Software Maintenance and Reengineering, pages 411–416. IEEE, 2012. doi: 10.1109/CSMR.2012.79
[149] Christopher Peterson, Nansook Park, and Patrick J. Sweeney. Group Well-Being: Morale From a Positive Psychology Perspective. Applied Psychology, 57:19–36, 2008. doi: 10.1111/j.1464-0597.2008.00352.x.
[150] Diogo Pina, Alfredo Goldman, and Graziela Tonin. Technical Debt Prioritization: Taxonomy, Methods Results, and Practical Characteristics. In 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 206–213. IEEE, 2021. doi:10.1109/SEAA53835.2021.00034.
[151] Diogo Pina, Carolyn Seaman, and Alfredo Goldman. Technical Debt Prioritization: A Developer's Perspective. In 2022 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 46–55, 2022. doi:10.1145/3524843.3528096.
[152] Poliakov, Dmitrii. A Systematic Mapping Study on Technical Debt Definition. Lappeenranta University of Technology, 2015. MSc Thesis.
[153] Akond Rahman, Rezvan Mahdavi-Hezaveh, and Laurie Williams. A Systematic Mapping Study of Infrastructure as Code Research. Information and Software Technology, 108:65–77, 2019. doi:https://doi.org/10.1016/j.infsof.2018.12.004.
[154] Paul Ralph and Ewan Tempero. Construct Validity in Software Engineering Research and Software Metrics. In 2018 22nd International Conference on Evaluation and Assessment in Software Engineering (EASE), pages 13–23, 2018. doi: 10.1145/3210459.3210461.
[155] Narayan Ramasubbu and Chris F. Kemerer. Managing Technical Debt in Enterprise Software Packages. IEEE Transactions on Software Engineering, 40(8):758–772, 2014. doi: 10.1109/TSE.2014.2327027.
[156] Narayan Ramasubbu and Chris F. Kemerer. Integrating Technical Debt Management and Software Quality Management Processes: A Normative Framework and Field Tests. IEEE Transactions on Software Engineering, 45(3):285–300, 2017. doi: 10.1109/TSE.2017.2774832.
[157] Narayan Ramasubbu, Chris F. Kemerer, and C. Jason Woodard. Managing Technical Debt: Insights From Recent Empirical Evidence. IEEE Software, 32(2):22–25, 2015. doi: 10.1109/MS.2015.45.
[158] Cesario Ramos. Product Definition in Large Scale Scrum (LeSS), 2020. URL https://less.works/blog/2020/07/07/product-definition-in-less-1.html. retrieved on Apr. 22, 2023.
[159] Xiaoxue Ren, Zhenchang Xing, Xin Xia, David Lo, Xinyu Wang, and John Grundy. Neural Network-Based Detection of Self-Admitted Technical Debt: From Performance to Explainability. ACM Transactions on Software Engineering and Methodology (TOSEM), 28(3):1–45, 2019. doi:10.1145/3324916.

Bibliography 5/6

[160] Leilane Ferreira Ribeiro, Mário André de Freitas Farias, Manoel G. Mendonça, and Rodrigo Oliveira Spínola. Decision Criteria for the Payment of Technical Debt in Software Projects: A Systematic Mapping Study. In 18th International Conference on Enterprise Information Systems (ICEIS), volume 1, pages 572–579, 2016. doi: 10.5220/0005914605720579.
[161] Nicolli Rios, Manoel Gomes de Mendonça Neto, and Rodrigo Oliveira Spínola. A Tertiary Study on Technical Debt: Types, Management Strategies, Research Trends, and Base Information for Practitioners. Information and Software Technology, 102:117–145, 2018. doi:10.1016/j.infsof.2018.05.010.
[162] Nicolli Rios, Rodrigo Oliveira Spínola, Manoel Mendonça, and Carolyn Seaman. The Most Common Causes and Effects of Technical Debt: First Results From a Global Family of Industrial Surveys. In 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pages 1–10, 2018. doi: 10.1145/3239235.3268917.
[163] Nicolli Rios, Rodrigo Oliveira Spínola, Manoel Mendonça, and Carolyn Seaman. Supporting Analysis of Technical Debt Causes and Effects With Cross-Company Probabilistic Cause-Effect Diagrams. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 3–12. IEEE, 2019.doi: 10.1109/TechDebt.2019.00009.
[164] Colin Robson and Kieran McCartan. Real World Research. John Wiley & Sons, 4th edition, 2016.
[165] Kenneth S. Rubin. Essential Scrum: A Practical Guide to the Most Popular Agile Process. Addison-Wesley, 1st edition, 2012.
[166] Per Runeson, Martin Host, Austen Rainer, and Bjorn Regnell. Case Study Research in Software Engineering: Guidelines and Examples. John Wiley & Sons, 1st edition, 2012.
[167] Caitlin Sadowski, Margaret-Anne Storey, and Robert Feldt. A Software Development Productivity Framework. In Caitlin Sadowski and Thomas Zimmermann, editors, Rethinking Productivity in Software Engineering, pages 39–47. Springer Nature, 2019. doi: 10.1007/978-1-4842-4221-6_5.
[168] Darius Sas and Paris Avgeriou. An Architectural Technical Debt Index Based on Machine Learning and Architectural Smells. IEEE Transactions on Software Engineering, 49(8):4169–4195, 2023. doi: 10.1109/TSE.2023.3286179.
[169] Scaled Agile, Inc. Architectural Runway, 2023. URL https://scaledagileframework.com/architectural-runway/. retrieved on Aug. 6, 2023.
[170] Scaled Agile, Inc. Scaled Agile Framework (SAFe), 2023. URL https://www.scaledagileframework.com. retrieved on Apr. 24, 2023.
[171] Klaus Schmid. On the Limits of the Technical Debt Metaphor. Some Guidance on Going Beyond. In 4th International Workshop on Managing Technical Debt (MTD), pages 63–66. IEEE, 2013. doi: 10.1109/MTD.2013.6608681.
[172] Ken Schwaber and Jeff Sutherland. Scrum Guide, 2020. URL https://scrumguides.org. retrieved on Feb. 4, 2023.
[173] Ezequiel Scott, Khaled Nimr Charkie, and Dietmar Pfahl. Productivity, Turnover, and Team Stability of Agile Teams in Open-Source Software Projects. In 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 124–131. IEEE, 2020. doi:10.1109/SEAA51224.2020.00029.
[174] Carolyn Seaman and Yuepu Guo. Chapter 2 – Measuring and Monitoring Technical Debt. In Marvin V. Zelkowitz, editor, Advances in Computers, volume 82, pages 25–46. Elsevier, 2011. doi: https://doi.org/10.1016/B978-0-12-385512-1.00002-5.
[175] Carolyn Seaman, Yuepu Guo, Nico Zazworka, Forrest Shull, Clemente Izurieta, Yuanfang Cai, and Antonio Vetrò. Using Technical Debt Data in Decision Making: Potential Decision Approaches. In 3rd International Workshop on Managing Technical Debt (MTD), pages 45–48. IEEE, 2012. doi: 10.1109/MTD.2012.6225999.
[176] Tushar Sharma. How Deep Is the Mud: Fathoming Architecture Technical Debt Using Designite. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 59–60. IEEE, 2019. doi:10.1109/TechDebt.2019.00018.
[177] Miltiadis Siavvas, Dimitrios Tsoukalas, Marija Jankovic, Dionysios Kehagias, and Dimitrios Tzovaras. Technical Debt as an Indicator of Software Security Risk: A Machine Learning Approach for Software Development Enterprises. Enterprise Information Systems, 16(5):1824017, 2022. doi: 10.1080/17517575.2020.1824017.
[178] SonarSource. SonarQube: Metric Definitions, 2023. URL https://docs.sonarqube.org/latest/user-guide/metric-definitions/. retrieved on Apr. 22, 2023.
[179] Robert E. Stake. The Art of Case Study Research. SAGE Publications, 1st edition, 1995.
[180] Philip B. Stark. Glossary of Statistical Terms. In SticiGui. University of California, Berkeley, CA, USA, 2019. URL https://www.stat.berkeley.edu/~stark/SticiGui/Text/gloss.htm. retrieved on Aug. 15, 2023.
[181] Marek G. Stochel. Reliability of Feedback Mechanism Based on Root Cause Defect Analysis—Case Study. Annales Universitatis Mariae Curie-Skłodowska. Sectio AI, Informatica, 11(4):21–32, 2011. URL https://bibliotekanauki.pl/articles/764245.
[182] Marek G. Stochel. Reliability and Accuracy of the Estimation Process—Wideband Delphi vs. Wisdom of Crowds. In 2011 IEEE 35th Computer Software and Applications Conference, pages 350–359. IEEE, 2011. doi:10.1109/COMPSAC.2011.53.
[183] Marek G. Stochel, Mariusz R. Wawrowski, and Magdalena Rabiej. Value-Based Technical Debt Model and Its Application. In 7th International Conference on Software Engineering Advances (ICSEA'12), pages 205–212. IARIA Press, 2012. URL https://www.thinkmind.org/index.php?view=article&articleid=icsea_2012_8_10_10039.
[184] Marek G. Stochel, Mariusz R. Wawrowski, and James J. Waskiel. Adaptive Agile Performance Modeling and Testing. In 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops, pages 446–451. IEEE, 2012. doi: 10.1109/COMPSACW.2012.85.
[185] Marek G. Stochel, Piotr Chołda, and Mariusz R. Wawrowski. On Coherence in Technical Debt Research: Awareness of the Risks Stemming From the Metaphorical Origin and Relevant Remediation Strategies. In 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 367–375. IEEE, 2020. doi: 10.1109/SEAA51224.2020.00067.
[186] Marek G. Stochel, Piotr Chołda, and Mariusz R. Wawrowski. Continuous Debt Valuation Approach (CoDVA) for Technical Debt Prioritization. In 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 362–366, 2020. doi:10.1109/SEAA51224.2020.00066.
[187] Marek G. Stochel, Piotr Chołda, and Mariusz R. Wawrowski. Adopting DevOps Paradigm in Technical Debt Prioritization and Mitigation. In 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pages 306–313. IEEE, 2022. doi: 10.1109/SEAA56994.2022.00055.
[188] Marek G. Stochel, Mariusz R. Wawrowski, and Piotr Chołda. Technical Debt Prioritization in Telecommunication Applications: Why the Actual Refactoring Deviates From the Plan and How to Remediate It? Applied Sciences, 12(22:11347), 2022. doi: 10.3390/app122211347.
[189] Marek G. Stochel, Tomasz Borek, Mariusz R. Wawrowski, and Piotr Chołda. Business-Driven Technical Debt Management Using Continuous Debt Valuation Approach (CoDVA). Information and Software Technology, 164:107333, 2023. doi: 10.1016/j.infsof.2023.107333.
[190] Peter Strečanský, Stanislav Chren, and Bruno Rossi. Comparing Maintainability Index, SIG Method, and SQALE for Technical Debt Identification. In 35th Annual ACM Symposium on Applied Computing (SAC), pages 121–124, New York, NY, USA, 2020. ACM. doi:10.1145/3341105.3374079.
[191] Dan Sturtevant. Silverthread CodeMRI Care. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 61–61. IEEE, 2019. doi: 10.1109/TechDebt.2019.00019.
[192] James Surowiecki. Wisdom of Crowds. Why the Many Are Smarter Than the Few. Abacus, London, UK, 1st edition, 2005.
[193] Girish Suryanarayana, Ganesh Samarthyam, and Tushar Sharma. Refactoring for Software Design Smells: Managing Technical Debt. Morgan Kaufmann, 1st edition, 2014.
[194] Damian A. Tamburri, Philippe Kruchten, Patricia Lago, and Hans van Vliet. What Is Social Debt in Software Engineering? In 2013 6th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), pages 93–96. IEEE, 2013. doi:10.1109/CHASE.2013.6614739.
[195] TechDebt. International Conference on Technical Debt, 2019. URL https://2019.techdebtconf.org. retrieved on May. 13, 2023.
[196] The LeSS Company B.V. Large-Scale Scrum Framework (LeSS), 2023. URL https://less.works/. retrieved on Aug. 15, 2023.
[197] Edith Tom, Aybüke Aurum, and Richard Vidgen. A Consolidated Understanding of Technical Debt. In 20th European Conference on Information Systems (ECIS 2012), pages 1–12, 2012.
[198] Edith Tom, Aybüke Aurum, and Richard Vidgen. An Exploration of Technical Debt. Journal of Systems and Software, 86(6):1498–1516, 2013. doi: 10.1016/j.jss.2012.12.052.

Bibliography 6/6

[199] Graziela Simone Tonin. Technical Debt Management in the Context of Agile Methods in Software Development. PhD thesis, University of São Paulo, São Paulo, Brazil, 2018.
[200] Adam Tornhill and Markus Borg. Code Red: The Business Impact of Code Quality—A Quantitative Study of 39 Proprietary Production Codebases. In 2022 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 11–20, 2022. doi: 10.1145/3524843.3528091.
[201] António Trigo, João Varajão, and Leandro Sousa. DevOps Adoption: Insights From a Large European Telco. Cogent Engineering, 9(1):2083474, 2022. doi: 10.1080/23311916.2022.2083474.
[202] Dimitrios Tsoukalas, Dionysios Kehagias, Miltiadis Siavvas, and Alexander Chatzigeorgiou. Technical Debt Forecasting: An Empirical Study on Open-Source Repositories. Journal of Systems and Software, 170:110777, 2020. doi: 10.1016/j.jss.2020.110777.
[203] Dimitrios Tsoukalas, Nikolaos Mittas, Alexander Chatzigeorgiou, Dionysios Kehagias, Apostolos Ampatzoglou, Theodoros Amanatidis, and Lefteris Angelis. Machine Learning for Technical Debt Identification. IEEE Transactions on Software Engineering, 48(12):4892–4906, 2022. doi: 10.1109/TSE.2021.3129355.
[204] Mark Velasquez and Patrick T. Hester. An Analysis of Multi-Criteria Decision Making Methods. International Journal of Operations Research, 10(2):56–66, 2013.
[205] vFunction. Technical Debt. A Guide for Frustrated Software Architects and CIOs, 2022. URL https://vfunction.com/resources/whitepaper-technical-debt-guide/. white paper, retrieved on Aug. 2, 2023.
[206] Santiago Vidal, Hernan Vazquez, J. Andres Diaz-Pace, Claudia Marcos, Alessandro Garcia, and Willian Oizumi. JSpIRIT: A Flexible Tool for the Analysis of Code Smells. In 2015 34th International Conference of the Chilean Computer Science Society (SCCC), pages 1–6, 2015. doi: 10.1109/SCCC.2015.7416572.
[207] Arturo Villa, Jorge Octavio Ocharán-Hernández, Juan Carlos Pérez-Arriaga, and Xavier Limón. A Systematic Mapping Study on Technical Debt in Microservices. In 2022 10th International Conference in Software Engineering Research and Innovation (CONISOFT), pages 182–191. IEEE, 2022. doi: 10.1109/CONISOFT55708.2022.00032.
[208] Alexander von Zitzewitz. Mitigating Technical and Architectural Debt With Sonargraph. In 2019 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 66–67. IEEE, 2019. doi:10.1109/TechDebt.2019.00022.
[209] Stefan Wagner and Florian Deissenboeck. Defining Productivity in Software Engineering. In Caitlin Sadowski and Thomas Zimmermann, editors, Rethinking Productivity in Software Engineering, pages 29–38. Springer Nature, 2019. doi: 10.1007/978-1-4842-4221-6_4.
[210] Anna Wiedemann, Nicole Forsgren, Manuel Wiesche, Heiko Gewald, and Helmut Krcmar. The DevOps Phenomenon: An Executive Crash Course. Queue, 17(2):93–112, 2019. doi: 10.1145/3329781.3338532.
[211] Marion Wiese and Klara Borowa. IT Managers' Perspective on Technical Debt Management. Journal of Systems and Software, 202:111700, 2023. doi: 10.1016/j.jss.2023.111700.
[212] Marion Wiese, Matthias Riebisch, and Julian Schwarze. Preventing Technical Debt by Technical Debt Aware Project Management. In 2021 IEEE/ACM International Conference on Technical Debt (TechDebt), pages 84–93, 2021. doi: 10.1109/TechDebt52882.2021.00018.
[213] Claes Wohlin. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In 18th International Conference on Evaluation and Assessment in Software Engineering, pages 1–10, 2014. doi: 10.1145/2601248.2601268.
[214] Daniele Wolfart, Wesley Klewerton Guez Assunção, and Jabier Martinez. Variability Debt: Characterization, Causes and Consequences. In 20th Brazilian Symposium on Software Quality (SBQS), New York, NY, USA, 2021. ACM. doi: 10.1145/3493244.3493250.
[215] Robert K. Yin. Case Study Research and Applications: Design and Methods. SAGE Publications, Thousand Oaks, CA, USA, 6th edition, 2018.
[216] Jesse Yli-Huumo, Andrey Maglyas, and Kari Smolander. How Do Software Development Teams Manage Technical Debt?—An Empirical Study. Journal of Systems and Software, 120:195–218, 2016. doi: 10.1016/j.jss.2016.05.018.
[217] Ehsan Zabardast, Javier Gonzalez-Huerta, Tony Gorschek, Darja Šmite, Emil Alégroth, and Fabian Fagerholm. A Taxonomy of Assets for the Development of Software-Intensive Products and Services. Journal of Systems and Software, 202:111701, 2023. doi: 10.1016/j.jss.2023.111701.
[218] Ehsan Zabardast, Javier Gonzalez-Huerta, Francis Palma, and Panagiota Chatzipetrou. The Impact of Ownership and Contribution Alignment on Code Technical Debt Accumulation. arXiv preprint arXiv:2304.02140, 2023. doi: 10.48550/arXiv.2304.02140.
[219] Nico Zazworka, Carolyn Seaman, and Forrest Shull. Prioritizing Design Debt Investment Opportunities. In 2nd Workshop on Managing Technical Debt, pages 39–42, 2011. doi: 10.1145/1985362.1985372.
[220] Nico Zazworka, Rodrigo Oliveira Spínola, Antonio Vetro', Forrest Shull, and Carolyn Seaman. A Case Study on Effectively Identifying Technical Debt. In 2013 17th International Conference on Evaluation and Assessment in Software Engineering (EASE), pages 42–47, 2013. doi:10.1145/2460999.2461005.
[221] Hongyu Zhang, Liang Gong, and Steve Versteeg. Predicting Bug-Fixing Time: An Empirical Study of Commercial Software Projects. In 2013 35th International Conference on Software Engineering (ICSE), pages 1042–1051, 2013. doi: 10.1109/ICSE.2013.6606654.
[222] Chenxing Zhong, Huang Huang, He Zhang, and Shanshan Li. Impacts, Causes, and Solutions of Architectural Smells in Microservices: An Industrial Investigation. Software: Practice and Experience, 52, 08 2022. doi:10.1002/spe.3138.